

CNN-Based Camera Pose Estimation and Localisation of Scan Images for Aircraft Visual Inspection

Xueyan Oh¹, Leonard Loh¹, Shaohui Foong¹, Zhong Bao Andy Koh², Kow Leong Ng², Poh Kang Tan², Pei Lin Pearlin Toh², and U-Xuan Tan¹

Abstract—General Visual Inspection is a manual inspection process regularly used to detect and localise obvious damage on the exterior of commercial aircraft. There has been increasing demand to perform this process at the boarding gate to minimise the downtime of the aircraft and automating this process is desired to reduce the reliance on human labour. Automating this typically requires estimating a camera’s pose with respect to the aircraft for initialisation but most existing localisation methods require infrastructure, which is very challenging in uncontrolled outdoor environments and within the limited turnover time (approximately 2 hours) on an airport tarmac. Additionally, many airlines and airports do not allow contact with the aircraft’s surface or using UAVs for inspection between flights, and restrict access to commercial aircraft. Hence, this paper proposes an on-site method that is infrastructure-free and easy to deploy for estimating a pan-tilt-zoom camera’s pose and localising scan images. This method initialises using the same pan-tilt-zoom camera used for the inspection task by utilising a Deep Convolutional Neural Network fine-tuned on only synthetic images to predict its own pose. We apply domain randomisation to generate the dataset for fine-tuning the network and modify its loss function by leveraging aircraft geometry to improve accuracy. We also propose a workflow for initialisation, scan path planning, and precise localisation of images captured from a pan-tilt-zoom camera. We evaluate and demonstrate our approach through experiments with real aircraft, achieving root-mean-square camera pose estimation errors of less than 0.24 m and 2° for all real scenes.

Index Terms—Localisation, inspection, aircraft maintenance.

I. INTRODUCTION

General Visual Inspection (GVI) is a widely used technique as part of regular inspections of aircraft such as during pre-flight inspections on an airport tarmac or during maintenance usually performed in a hanger. This process involves visual examinations of the aircraft’s exterior for noticeable damage or irregularities and provides a means for early detection of typical air-frame defects [2]. Currently, this is manually performed by well-trained personnel, which is labour intensive and have high error rates [2, 3]. To address this, research has shown that automating this process can increase the speed of inspection, and increase the accuracy of detecting defects among other benefits [4, 5].

Many studies have explored automating the task of detecting defects on the surface of aircraft. Mumtaz *et al.* [6] propose a visual approach using directional energies

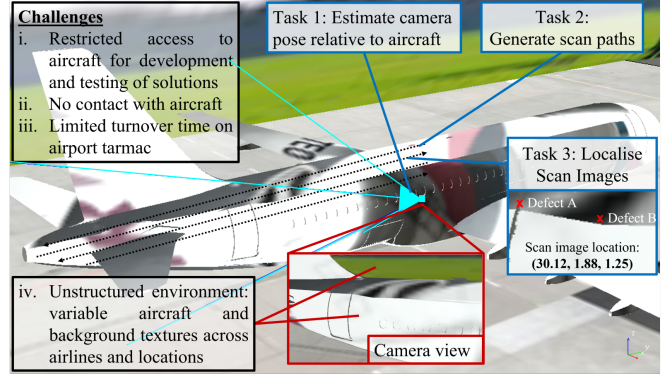


Fig. 1: We propose an infrastructure-free and contact-less method to estimate a PTZ camera’s pose with respect to an aircraft and generate a scan path to obtain scan images labelled with their location on the aircraft’s surface.

of textures to detect and distinguish between cracks and scratches. Malekzadeh *et al.* [7] train deep neural networks to detect defects within images of an aircraft fuselage. Miranda *et al.* [8] focus on detecting and inspecting exterior screws on aircraft by training a deep convolutional neural network (DCNN) to extract screws from images and apply algorithms to evaluate the state of each screw. Miranda *et al.* [9] address the issue of data imbalance between different defect categories in datasets used to train defect classification networks by proposing a hybrid machine learning method that improves defect classification performance for the under-represented category. Doğru *et al.* [10] tackle the issue of small datasets for aircraft defects by using a DCNN combined with data augmentation techniques to improve the performance of detecting dents on aircraft surfaces within images. While the above studies show the potential of automating defect detection within images, localisation of the defects with respect to the aircraft is still missing.

On top of detecting abnormalities, other studies have suggested automation of the tedious data collection process with the use of ground-based robotic systems. Several works [4, 11] study the use of a Pan-Tilt-Zoom (PTZ) camera mounted on an autonomous mobile ground robot that relies on navigating to pre-defined checkpoints with the help of laser range finders or image data. However, these systems are only designed to inspect larger-scale features from the ground, such as an air-inlet vent or an oxygen bay handle, and do not require high camera viewpoint accuracy. Jovančević *et al.* [5] propose the use of a similar autonomous mobile ground robot that moves along a pre-defined path around an A320 within a hanger to collect 3D point cloud data using a 3D scanner. This work focuses on the analysis of point cloud data for defect detection and classification and not on localising them. These solutions are also designed

¹X. Oh, L. Loh, S. Foong and U-X. Tan are with the Pillar of Engineering Product Development, Singapore University of Technology and Design, Singapore, xueyan_oh@mymail.sutd.edu.sg, uxuan_tan@sutd.edu.sg.

²Z. B. A. Koh, K. L. Ng, P. K. Tan and P. L. P. Toh are with ST Engineering Aerospace Ltd., Singapore

A preliminary version [1] of this paper has been published in ICRA 2021.

to be implemented in a hangar during maintenance, which provides a controlled environment to set up infrastructure for establishing checkpoints around the aircraft. Hence, they are not suitable to be performed outdoors on the airport tarmac and with limited turnover time (approximately 2 hours) between flights.

The use of Unmanned Aerial Vehicles (UAV) have also been explored for aircraft inspection. Malandrakis *et al.* [12] and Tzitzilonis *et al.* [13] use a commercially available UAV attached with a UV LED torch and a camera for autonomous inspection of an aircraft wing. Both works suggest that the UAV autonomously navigate between fixed waypoints along the wing using optical flow and capture images. However, the initial pose of the UAV is still required. Umberto and Salvatore [2] also use a UAV to automate defect detection and maintains a fixed vertical distance from the aircraft surface using ultrasonic sensors but is piloted by a human operator and relative pose to the aircraft is not known. Cazzato *et al.* [14] estimate the pose of a UAV relative to an aircraft by comparing the features of known landmarks between the real image and a reference 3D model. However, the landmarks are specific logos and flags painted on the aircraft and assumed to be present on the reference model. This is not feasible in reality as the paintwork of an aircraft changes from time to time and differs between airlines. Lastly, Bugaj *et al.* [15] present a methodology and the benefits for using semi-autonomous UAVs in aircraft pre-flight inspections in a safe manner. While these studies show the potential of using UAVs in aircraft inspections, none of them explore an end-to-end solution that can be quickly deployed in uncontrolled environment and capable of precisely localising detected defects, and the use of UAVs is also prohibited in many airports due to safety and security concerns.

It is difficult to localise a detected defect just by looking at an optically zoomed in image as there are often no location-specific features within an image. Additionally, many airlines and airports do not allow contact with the aircraft's surface between flights, causing pose estimation methods that require physically attaching sensors or markers onto the surface of aircraft to be unusable. To obtain an image's location on the aircraft's surface, it is required to first determine the camera's initial pose relative to the aircraft which can be referred to as Camera Pose Estimation (CPE). The recent increase in the use of Deep Convolutional Neural Networks (DCNNs) for monocular CPE has shown its potential to out-perform classical 3D structure-based methods in several aspects, including shorter inference times, smaller memory [16], and robustness to uncontrolled environments [17]. This shows the potential viability of applying DCNNs for CPE as an infrastructure-free and contact-less method to initialise aircraft inspection systems on airport tarmacs, under limitations such as prohibited contact with aircraft, time constraints and variations in background and lighting.

Our main contributions in this paper are as follows:

- We propose an end-to-end workflow for obtaining scan images labelled with their location on an aircraft's

surface using an easily deployable PTZ camera system.

- We propose a method to initialise and estimate the pose of the camera used for inspection without requiring physical contact with or prior access to a real aircraft by using a DCNN fine-tuned on only synthetic images of a 3D aircraft model to estimate the camera's pose given a real image as input.
- We improve the network's performance by adding a new component to the loss function. This additional loss component leverages on known geometry of an aircraft to provide a geometric relationship between the predicted position and orientation of the camera.

In this work, we build on the preliminary version [1] of this paper and the key differences include: 1) proposed approach of end-to-end workflow from initialisation of the inspection camera to localising scan images; 2) experimental validation of the proposed algorithm with aircraft in different scenes with more comprehensive evaluation of our pose estimation method; 3) a scanning strategy for a PTZ camera to scan and label images of the upper-half surface of an aircraft with their 3D coordinates; and 4) demonstration of the complete workflow on a real aircraft.

II. RELATED WORK

A. CNN-Based Camera Pose Estimation

CPE can be described as using an input image to output an estimate of the pose – position and orientation – of the camera [16]. In most cases, the pose is obtained with respect to a predefined global reference. PoseNet [18] and other similar deep architectures that predict a camera pose [19]–[21] share a common process, where images from a database are used as input into a DCNN for training, with the aim of minimising the error between the predicted and ground truth pose. PoseNet [18] is the pioneer to introduce the use of a DCNN – based on a modified GoogLeNet [22] – to directly regress a camera pose, and many improved methods based on deep learning architectures have since been proposed [16]. These deep pose estimation methods have been fine-tuned and tested on publicly available indoor and outdoor datasets that have been collected using hand-held camera devices and using software to automate the retrieval of camera poses [18]. However, models fine-tuned on these datasets may not accurately represent their effectiveness in CPE for our application due to the difficulties in obtaining a real dataset to train on in the first place, as well as the need to accommodate scene changes such as different appearance of the same aircraft model and its background.

B. Using Synthetic Images to Fine-Tune DCNNs

Several research [23]–[26] explore fine-tuning DCNNs without the need for real images by using synthetic scenes for tasks related to camera pose estimation. Among these, Sadeghi and Levine [23] propose a learning method for a drone to autonomously navigate an indoor environment

without collision, by training a network through reinforcement learning using only 3D CAD models. Only RGB images rendered from a 3D indoor environment with random textures, object positions and lighting are used to train the CNN to output velocity commands, and they achieve autonomous drone navigation and obstacle avoidance. While they explore the ability of a network trained on synthetic images to generalise to the real world, their objective is to avoid collision as opposed to CPE.

Acharya *et al.* [24] propose a solution for indoor CPE by fine-tuning PoseNet [18] using synthetic images rendered from a low-detail 3D indoor environment, modelled with reference to a Building Information Model (BIM). This work explores different methods of rendering, from cartoon-like to photo-realistic and textured to rendering only edges within each scene, achieving CPE from real images with an accuracy of about 2 m. Another study [25] proposes to use synthetic image sequences to fine-tune a recurrent neural network for CPE and show that it improves CPE accuracy when testing on real image sequences, achieving CPE accuracy of about 1.6 m. More recently, Acharya *et al.* [26] show that combining edge maps and semantic segmentation can close the domain gap between synthetic images for training the CNN and real images and improve sim-to-real camera pose estimation, achieving an accuracy of 1.12 m and 6.06° in indoor scenes. However, these methods have only been tested in environments with substantial changes in scenes and viewpoints as the camera relocates within the environment. This is as opposed to differentiating the camera pose between images that are captured with slight changes in viewpoint in the context of aircraft GVI. Moreover, the accuracy reported in most of these works are insufficient for our application.

C. Generating Synthetic Images via Domain Randomisation

Tobin *et al.* [27] investigate the use of domain randomisation to bridge the gap between simulation and reality. They argue that models fine-tuned with only synthetic scenes can generalise to real scenes if the scenes are diverse enough. The authors generate their dataset by uniformly randomising many aspects of their domain, including size, shape, position and colour of objects in each scene, and successfully teach a robotic arm to pick objects within a real, crowded indoor environment using only “low-fidelity” rendered images. Following this, others [28]–[30] also use domain randomisation for deep pose estimation tasks without training on real data. Despite being robust to object distractors, these models only apply to object pose estimation and often have other unchanging major objects such as a table where objects are placed on or a robot gripper which provides useful information of each object’s pose relative to these major objects in the scene. We also use domain randomisation in our work, but focus on CPE with respect to an aircraft without any other known objects in the scene.

In summary, it is challenging to develop deep learning based solutions for CPE with respect to an aircraft on an

airport tarmac due to the limited access to real aircraft and solutions need to be robust to large variations in environment and aircraft texture across airlines. We propose to address this challenge by eliminating the need to obtain real images for fine-tuning and use only synthetic images of the aircraft’s 3D model in scenes varied using domain randomisation. We also leverage on known geometry of an aircraft’s surface from its Structural Repair Manual (SRM) to explore a geometric relationship between the camera’s position and orientation within the network’s loss function and improve the pose estimation accuracy. To complete the workflow, we propose a path planning method for automated scanning of an aircraft with a PTZ camera that uses the output of CPE as well as information from the 3D model of the aircraft and label every captured image with the estimated location of its centre pixel on the aircraft surface.

III. WORKFLOW FOR SCANNING AND LOCALISING IMAGES FOR AIRCRAFT VISUAL INSPECTION

Performing automated visual inspection of aircraft between flights demands minimal infrastructure due to the short turn-over time, and physical contact with the aircraft is prohibited by some airlines and airports. We propose a workflow, illustrated in Fig. 2, that achieves this using only the same PTZ camera for inspection mounted onto an extendable mast. The workflow consists of two phases, the first is called the *Preparation Phase* and is only required when preparing the system for a new aircraft model or to improve the performance of the system (such as to fine-tune the DCNN). The second phase is called the *Deployment Phase* which can be repeated on aircraft of the same model as long as the *Preparation Phase* has been completed for that model.

The *Preparation Phase* starts by sectioning the chosen aircraft model (we choose an Airbus A320 in this work) into four quadrants and identifying their boundaries (both position and orientation) in which operators can reasonably position the PTZ camera (on its mast) within using aircraft features as visual guides. Second, a fully synthetic dataset is separately generated for each quadrant from a simulator and used to fine-tune a DCNN to predict the pose of a camera relative to the aircraft given a single image. Third, the 3D aircraft model’s point cloud is interpolated to prepare for scan path generation during the *Deployment Phase*.

In the *Deployment Phase*, a PTZ camera of known FOV is set up within the proposed boundary of a chosen quadrant to be scanned. The PTZ camera is first initialised with CNN-based CPE using an image captured by the PTZ camera at full FOV. Next, points in the interpolated 3D aircraft model point cloud from the *Preparation Phase* is converted to their respective PTZ camera pan-tilt values, given the estimated pose of the camera with respect to the aircraft. A scan path is then generated by selecting pan-tilt pairs based on the desired zoom level for the scan. Lastly, the scan is performed by commanding the PTZ camera to each of the selected pan-tilt pairs at the desired zoom level to capture images and label their centre pixels with their respective Cartesian coordinates.

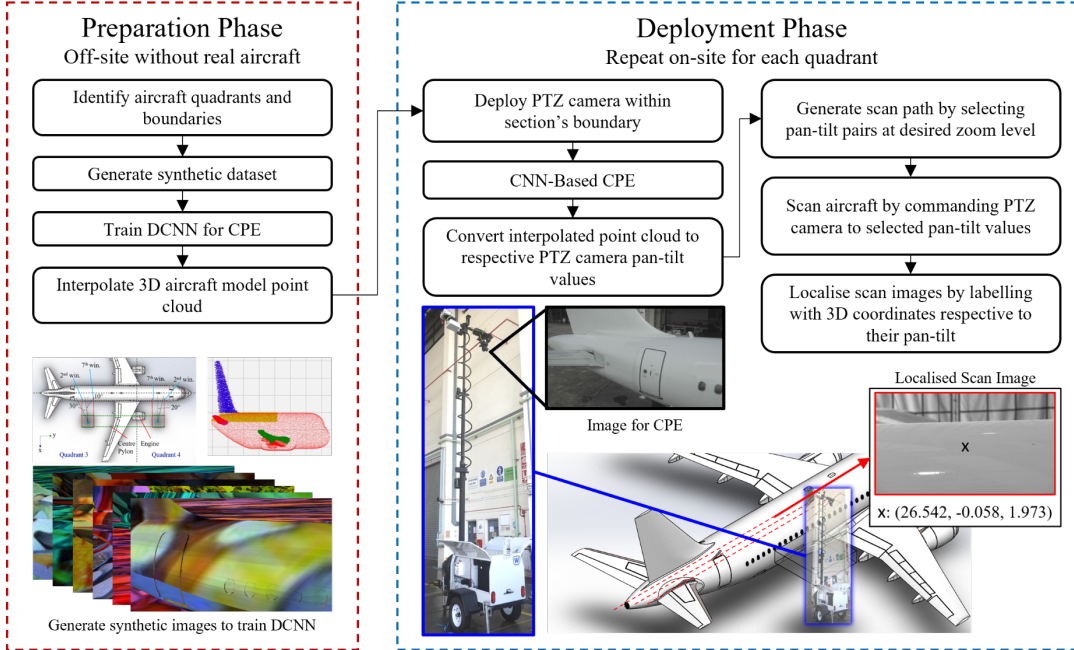


Fig. 2: Overview of our proposed workflow for scanning and localising images for aircraft visual inspection on an airport tarmac between flights. Our method is easy to deploy and does not require additional infrastructure or attaching sensors or markers onto the aircraft for camera pose estimation (CPE).

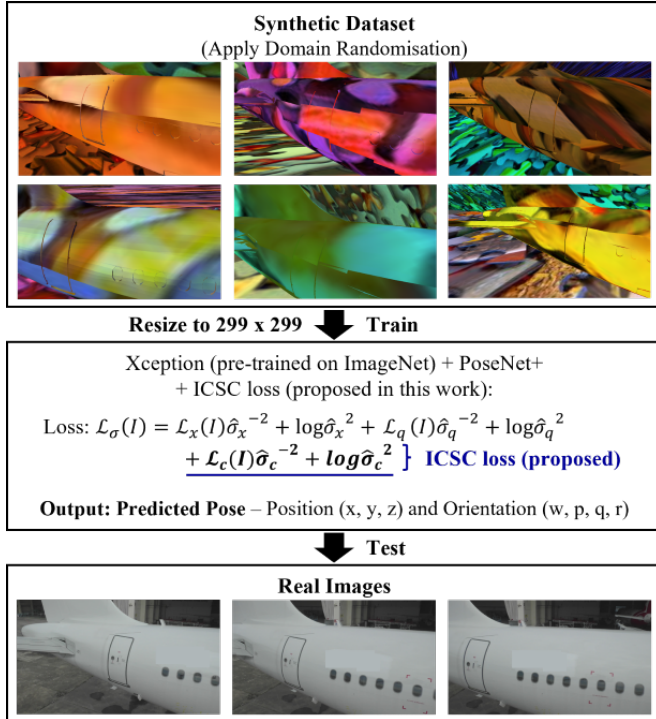


Fig. 3: Summary of deep learning approach.

IV. CNN-BASED CAMERA POSE ESTIMATION

We first design a setup with realistic constraints and assumptions for how a PTZ camera of known specifications can be easily deployed next to an aircraft for the purpose of inspecting the upper surface of the aircraft. More information of the setup is mentioned in IV-A. Based on this setup, we use a virtual camera within a virtual 3D environment to capture images while applying domain randomisation to generate our synthetic dataset. We use this synthetic dataset to fine-tune

a network that regresses the camera's pose from an input image. We base our network on an existing PoseNet variant with learnable weights [21] (we refer to as PoseNet+). In addition, we propose modifying its loss function by introducing an additional component that provides a geometric relationship between the camera's position and orientation. Fig. 3 summarises our deep learning approach.

A. Proposed Setup with PTZ Camera

We split an aircraft into four quadrants (refer to Fig. 4) and use only one side, quadrants 3 and 4, of an Airbus A320 (A320 in short) to illustrate our method and assume that the other half of the aircraft (quadrants 1 and 2) is similar and the method can be easily mirrored. It is assumed that the PTZ camera's specifications are known and we use its maximum FOV (at 1x zoom) for initialisation. It is also assumed that the PTZ camera's base can be easily levelled (no roll and pitch relative to the ground) with the use of a gimbal or level gauge, reducing the problem to 4 Degrees of Freedom - position and yaw. The following steps are proposed for deployment:

- 1) Position the PTZ camera within a reasonable region as illustrated in Fig. 4 and Fig. 5, which includes an area of 3 m by 3 m in each quadrant, and can be easily performed using visual check.
- 2) Raise the PTZ camera to a height of 6.25 m to 7.25 m from the ground via equipment such as an electronic mast or a boom lift and easily approximated with the use of accessible equipment such as a range finder.
- 3) Rotate the camera's base about the z-axis to face the aircraft perpendicularly (within $\pm 10^\circ$ yaw error).
- 4) From its home pan-tilt orientation, pan the camera 20° towards the aircraft's tail (quadrant 3), or 10° towards

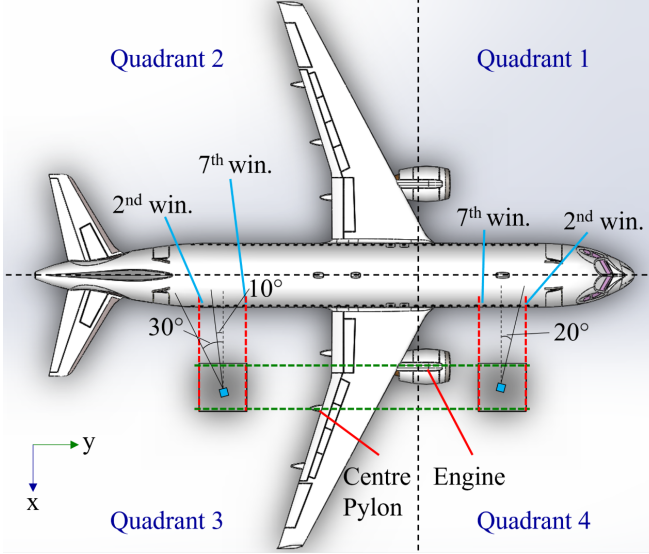


Fig. 4: Aircraft quadrants and proposed boundary for PTZ camera position and axes direction. Windows and pylons of an A320 are used as a visual guide to position the PTZ camera within the proposed boundary.

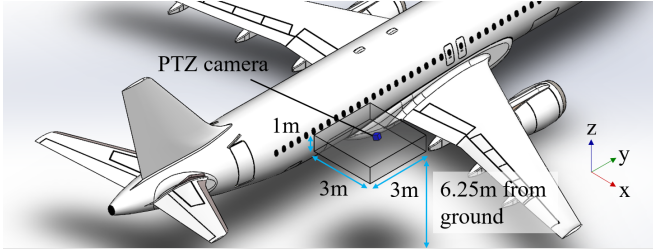


Fig. 5: Proposed 3D boundary for PTZ camera position in quadrant 3 with direction of axes.

the aircraft’s nose (quadrant 4), and tilt 18° towards the ground via software commands and capture an image for initialisation.

Fig. 4 shows the features (windows, pylon, and engine) of an A320 to use as visual guides for the quick manual positioning of the mast, while Fig. 5 illustrates the proposed permissible 3D space where the PTZ camera can be set up using quadrant 3 as an example. The allowable yaw error (referred to as α) of $\pm 10^\circ$ due to manual orientation towards the aircraft suggests that the camera is oriented to between $+10^\circ$ and $+30^\circ$ about the z-axis for quadrant 3, and between 0° and -20° about the z-axis for quadrant 4.

B. Virtual Environment and Synthetic Dataset

We obtain the 3D model of an A320 from a GrabCAD contribution [31]. Minor modifications are made to match general features and overall dimensions of a real A320, based on details obtained from an A320’s SRM. Our virtual setup is shown in Fig. 6. To create this 3D environment, we place our 3D model into a scene in robot simulator CoppeliaSim [32]. A large wall is added on one side of the aircraft model as background. A virtual camera is placed beside the aircraft and its FOV is set to match the real PTZ camera at 1x zoom.

We apply domain randomisation [27] when generating our synthetic dataset as it has been demonstrated to be capable

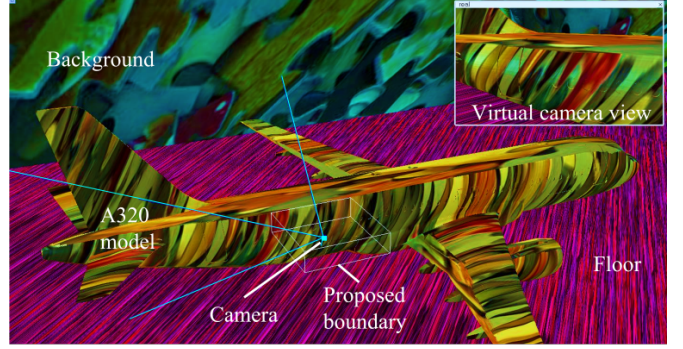


Fig. 6: Virtual setup with an instance of domain randomisation.

of generalising to real-world data, given sufficient simulated variability. We use a free stock image of a randomly scattered puzzle to apply as texture for the ground, aircraft model, and background. We randomise the following aspects when capturing each image for our dataset and use quadrant 3 to illustrate our approach:

- The PTZ camera’s position within the proposed 3 m x 3 m x 1 m boundary in quadrant 3;
- The PTZ camera’s pan between $+10^\circ$ and $+30^\circ$;
- The PTZ camera’s tilt between -17.5° to -18.5° . (slight tolerance of $\pm 0.5^\circ$ from the proposed 18° tilt);
- Colour – RGB values of both ambient and specular components for the texture of every object; and
- The position, orientation, as well as horizontal and vertical scaling factors of textures applied onto all surfaces.

For each camera with a different FOV at 1x zoom, we generate 4000 synthetic images for fine-tuning, 700 for validation and 300 for testing.

C. Deep Learning Approach

We use an improved variant [21] (which we refer to as PoseNet+) of PoseNet [18] for its ability to directly regress for CPE given a single input image. We apply PoseNet+’s approach onto a more recent deep architecture, Xception [33], as it results in substantially better performance than GoogLeNet (Inception v1) originally used in PoseNet+. We modify the Xception network in a similar fashion to PoseNet+, by replacing the softmax layer with regression layers that output position (x, y, z) and orientation vectors (quaternions – w, p, q, r). We resize every input image to match the network’s 299 x 299 pixel input size without a centre crop as we find this to improve performance and attribute this to the increase in features and other spatial information that may be present in the whole image despite the distortion from resizing.

PoseNet+’s loss function learns a weighting between the position and orientation components and is formulated using the concept of homoscedastic uncertainty - a measure of uncertainty of the task and is independent of input data [34]. It is defined as:

$$\mathcal{L}_\sigma(I) = \mathcal{L}_x(I)\hat{\sigma}_x^{-2} + \log \hat{\sigma}_x^2 + \mathcal{L}_q(I)\hat{\sigma}_q^{-2} + \log \hat{\sigma}_q^2 \quad (1)$$

where $\mathcal{L}_x(I) = \|x - \hat{x}\|_2$ and $\mathcal{L}_q(I) = \|q - \frac{\hat{q}}{\|\hat{q}\|}\|_2$, with $\hat{x}$ and $\hat{q}$ representing the predicted position and orientation vectors respectively while x and q represent ground truth pose. $\hat{\sigma}_x^2$ and $\hat{\sigma}_q^2$ represent the homoscedastic uncertainties and are optimised with respect to the loss function through back propagation, and the reason for their usage is described in the paper introducing PoseNet+ [21].

Kendall *et al.* [21] also explore removing the need for balancing positional and rotational loss weights by directly learning from geometric reprojection loss but find that this loss was unable to converge without first pretraining on their originally proposed loss function (1). Inspired by their use of scene geometry, we propose to modify loss function (1) by introducing an additional loss component c that uses the scene coordinate of each image's centre pixel instead of reprojection. This is obtained by finding the point of intersection between the equation describing the camera's optical axis (as a function of x and q) and the aircraft's surface. With our proposed setup, we find this point of intersection is always on the upper half of the fuselage and propose to model the aircraft's surface as the equation of a cylinder. With c as the Cartesian coordinates of any point on the cylinder's surface, the equation of the surface is given by:

$$c_x^2 + (c_z - h_0)^2 = r_0^2 \quad (2)$$

Where c_x, c_y (any value along the cylinder's length) and c_z are the coordinates of a point on the cylinder's surface, h_0 is the displacement of the cylinder's cross-sectional centre from the scene's origin, and r_0 is the aircraft's fuselage radius. The line representing the camera's viewpoint is formulated as:

$$\vec{l} = \vec{x} + t\vec{v} \quad (3)$$

Where $\vec{l}$ represents the camera's viewpoint, $\vec{x}$ is the camera's position, $\vec{v}$ is obtained by rotating the camera's default direction vector by quaternion q , and t is a variable that determines the position of any point along line $\vec{l}$.

For every pair of camera position and orientation, we use equations (2) and (3) to solve for t where $c = \vec{l}$ to determine the point of intersection between line $\vec{l}$ and the surface of the cylinder. Since a line may intersect the surface of a cylinder at up to two points, only the point nearest to the camera's position, x , is kept. Fig. 7 illustrates how the aircraft fuselage's surface is modelled as the surface of a cylinder, as well as how $\mathcal{L}_x$ and $\mathcal{L}_q$ can be related by $\mathcal{L}_c$. We combine our proposed loss component with (1) to result in:

$$\begin{aligned} \mathcal{L}_\sigma(I) = & \mathcal{L}_x(I)\hat{\sigma}_x^{-2} + \log \hat{\sigma}_x^2 + \mathcal{L}_q(I)\hat{\sigma}_q^{-2} + \log \hat{\sigma}_q^2 \\ & + \mathcal{L}_c(I)\hat{\sigma}_c^{-2} + \log \hat{\sigma}_c^2 \end{aligned} \quad (4)$$

Where $\mathcal{L}_c(I) = \|c - \hat{c}\|_2$, and $c - \hat{c}$ represent the difference between the true and predicted point of intersection coordinates. While the variance σ^2 is learnt, the logarithmic regularisation term prevents the network from learning an infinite variance to achieve zero loss. Hence, $\hat{s} := \log \hat{\sigma}^2$ is

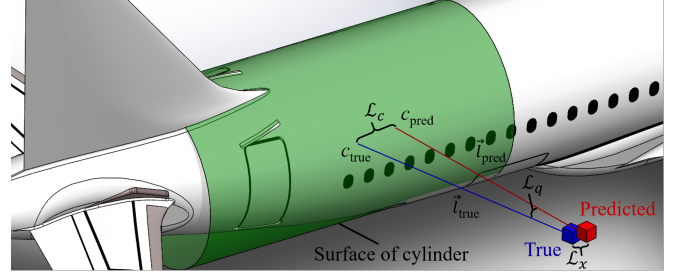


Fig. 7: Visualisation of how the top half of an aircraft's fuselage is modelled as the surface of a cylinder (green). Losses $\mathcal{L}_x$, $\mathcal{L}_q$ and $\mathcal{L}_c$ can also be visualised as the difference between their respective true and predicted components.

learnt during implementation as it avoids a potential division by zero, resulting in the following function:

$$\begin{aligned} \mathcal{L}_\sigma(I) = & \mathcal{L}_x(I) \exp(-\hat{s}_x) + \hat{s}_x + \mathcal{L}_q(I) \exp(-\hat{s}_q) + \hat{s}_q \\ & + \mathcal{L}_c(I) \exp(-\hat{s}_c) + \hat{s}_c \end{aligned} \quad (5)$$

Where $\hat{s}_x$, $\hat{s}_q$ and $\hat{s}_c$ are learnt and we arbitrarily initialise all of them to zero. We refer to our proposed additional loss component as the Image Centre Scene Coordinate (ICSC) loss and the network with loss function (5) as ICSC-PoseNet.

V. SCAN PATH GENERATION AND IMAGE LOCALISATION

After estimating the PTZ camera's pose, we propose three steps to generate a path in pan-tilt values for the camera to follow to scan the upper half of the aircraft. The first step extracts relevant scan sections from within the point cloud of the 3D aircraft model and performing a linear interpolation on each sections separately. Secondly, the interpolated surface points are converted from Cartesian coordinates into an array of pan-tilt values relative to the estimated PTZ camera's pose. Lastly, an algorithm extracts scan points from the array of pan-tilt values based on the desired FOV and amount of overlap between captured scan images.

A. Interpolation of Aircraft Model Point Cloud

Interpolation of the point cloud (from the 3D aircraft model) is performed separately on different features (i.e. the fuselage, tail, wing and horizontal stabiliser) as their major surfaces lie on different planes. We define the relevant scan sections for a camera positioned in the back half (quadrants 2 and 3) of the aircraft as the tail, top half of the wing, the stabiliser and the back half of the fuselage, while a camera in the front half (quadrants 1 and 4) will scan the remaining front half of the fuselage. Only the upper-half of the fuselage (above the windows) are included in the scan. Fig. 8 shows the side and top view of the aircraft model's point cloud with different scan sections relevant to quadrant 3 coloured for visualisation. For points on the fuselage, stabiliser and wing, z-coordinates are interpolated over x and y while for points on the tail, their x-coordinates are interpolated over y and z. All interpolation are performed at 5 cm intervals using SciPy's griddata function [35] to form an array for each

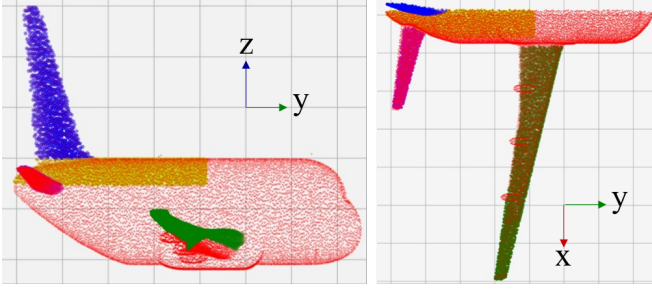


Fig. 8: Side view (left) and top view (right) of point cloud with direction of axes. Example of sectioning point cloud to fuselage (yellow), tail (blue), wing (green) and stabiliser (purple) for scanning by a camera in quadrant 3 are shown.

section where each row i has the same x -coordinate (except for the tail where each row has the same z -coordinate) and each column j has the same y -coordinate as such:

$$\begin{bmatrix} (x_{11}, y_{11}, z_{11}) & (x_{12}, y_{12}, z_{12}) & \dots & (x_{1j}, y_{1j}, z_{1j}) \\ (x_{21}, y_{21}, z_{21}) & (x_{22}, y_{22}, z_{22}) & \dots & (x_{2j}, y_{2j}, z_{2j}) \\ \vdots & \vdots & \ddots & \vdots \\ (x_{i1}, y_{i1}, z_{i1}) & (x_{i2}, y_{i2}, z_{i2}) & \dots & (x_{ij}, y_{ij}, z_{ij}) \end{bmatrix} \quad (6)$$

B. Point Cloud to Camera Pan-tilt

To convert the points in the interpolated point cloud into camera pan-tilt values, the camera is first placed at its estimated position with respect to the aircraft in the Cartesian space. Next, every point in the point cloud is converted into their respective pan-tilt values with the assumption that both the camera's and aircraft's x - y plane are parallel to each other. This is achieved by relating the camera's pan and tilt angles to the same angles (φ and θ) of a spherical coordinate system and can be converted from the Cartesian coordinate system in a similar manner. The equations for converting every point into camera pan and tilt are:

$$\varphi = \arctan \frac{y_c}{x_c} - \alpha \quad (7)$$

$$\theta = \arctan \frac{\sqrt{(x_c^2 + y_c^2)}}{z_c} \quad (8)$$

Where φ and θ are pan and tilt respectively, while x_c , y_c , and z_c are the displacements of each point in the interpolated point cloud with respect to the PTZ camera in the x , y , and z axis respectively. α represents the yaw error which is expected to be within $\pm 10^\circ$ as described in section IV-A, and can be obtained using:

$$\alpha = \gamma - \beta_q \quad (9)$$

Where γ is the camera's estimated angle in the yaw direction (rotation about the z -axis) extracted from the CNN's predicted quaternion output, with 0° being the yaw angle if the camera is facing exactly perpendicular to the aircraft's fuselage. β_q is the pre-defined angle rotated by the camera

for initialisation depending on the quadrant, q , it is placed in: $\beta_1 = +10^\circ$; $\beta_2 = -20^\circ$; $\beta_3 = +20^\circ$; $\beta_4 = -10^\circ$.

Equations (7), (8) and (9) are used to compute the respective pan-tilt values for every valid element in the array, (6) of each section, forming another array, U , with a similar organisation:

$$U = \begin{bmatrix} (p_{11}, t_{11}) & (p_{12}, t_{12}) & \dots & (p_{1j}, t_{1j}) \\ (p_{21}, t_{21}) & (p_{22}, t_{22}) & \dots & (p_{2j}, t_{2j}) \\ \vdots & \vdots & \ddots & \vdots \\ (p_{i1}, t_{i1}) & (p_{i2}, t_{i2}) & \dots & (p_{ij}, t_{ij}) \end{bmatrix} \quad (10)$$

Where p_{ij} and t_{ij} are the computed camera pan and tilt values, respective to each element with coordinates (x_{ij}, y_{ij}, z_{ij}) in (6).

C. Scan Path Generation and Image Localisation

To reduce the time taken to scan the aircraft, we generate an efficient scan path that does not capture images with unnecessary amount of overlap. This is achieved by taking into account both the desired Horizontal FOV (HFOV) and Vertical FOV (VFOV) of the camera during the scan, as well as an overlap factor, μ , which represents the desired minimum ratio of overlap between two consecutive images in terms of their HFOV and VFOV. We choose HFOV and VFOV to be 6.15° and 3.46° respectively (13x zoom) for our scans, and μ to be 0.15 as we observe that it results in sufficient overlaps between images.

We propose algorithm 1 (for fuselage and tail sections) to generate a list of scan points, V , given array U (10) for each section and a pre-defined overlap factor μ . For the fuselage and tail, the separation between rows in array U

Algorithm 1 Generate PTZ Camera Scan Path

- 1: **Input:** Array U
 - 2: **Output:** List V containing appended scan points
 - 3: Define $\lambda = 1 - \mu$ (overlap factor)
 - 4: $m_{last} = \text{median}(t_{1,all}) + \lambda \times VFOV$
 - 5: **for** row i in U **do**
 - 6: $m_{next} = \text{median}(t_{i,all})$
 - 7: $n_{last} = p_{i1} + \lambda \times HFOV$
 - 8: **if** $|m_{last} - m_{next}| \geq (\lambda \times VFOV)$ **or** i is last row of U **and** $|m_{last} - m_{next}| > VFOV/2$ **then**
 - 9: $m_{last} = m_{next}$
 - 10: **for** col j in row i **do**
 - 11: $n_{next} = p_{ij}$
 - 12: **if** $|n_{last} - n_{next}| \geq (\lambda \times HFOV)$ **or** j is last col of i **and** $|n_{last} - n_{next}| > HFOV/2$ **then**
 - 13: Append (p_{ij}, t_{ij}) to V
 - 14: $n_{last} = n_{next}$
 - 15: **end if**
 - 16: **end for**
 - 17: **end if**
 - 18: **end for**
-



Fig. 9: (Left) prototype: a 3D LiDAR and a PTZ camera mounted onto a passive 2-axis gimbal and secured onto a mounting stand during lab tests. (Right) the same prototype gimbal mounted to the top of a mast extended to about 7 m from the ground.

is related to the variation in camera tilt and VFOV. On the other hand, the separation between rows for array U of the wing and stabiliser is related to the camera pan and HFOV. Hence, the algorithm for the wing and stabiliser is similar except that all p_{ij} are replaced by t_{ij} and vice versa, and all VFOV are replaced by HFOV and vice versa. The eventual scan path for each section is in the order of the pan-tilt values in list V . Each scan image, captured from (p_{ij}, t_{ij}) , is labelled with its Cartesian coordinate by retrieving its respective coordinate (x_{ij}, y_{ij}, z_{ij}) from (6). This coordinate represents the estimated location of the image's centre pixel on the surface of the aircraft.

VI. EXPERIMENTAL STUDY

A. Experimental Setup

To evaluate our approach, we requested and obtained special access to an A320 to gather images and pose-related data for our real test dataset. We built a prototype consisting of a Panasonic AW-UE150 PTZ Camera and a Velodyne VLP-16 3D LiDAR mounted onto a passive 2-axis gimbal. The prototype was secured onto the top of a vertical mast that is extendable up to 8 m from the ground and brought within the proposed boundary. Fig. 9 shows the prototype in the lab and when mounted onto the mast. Images for the entire setup next to the aircraft are not shown as they are deemed sensitive by the venue and airline. The 3D LiDAR was only used to obtain ground truth, requiring multiple minor manual adjustments to ensure the desired features are observed within the point cloud, and is not used in the proposed methodology.

B. Real Dataset and Aircraft Scan Images

Real images with their ground truth poses were gathered from three scenes, each containing an A320 with different

paintwork, background and lighting to evaluate our method. We include results from our previous work [1] (Real Scene 1 - captured at 60° HFOV using a different PTZ camera) as they are relevant and gathered new real data captured at 72.5° HFOV. All images were obtained from 16 different positions in quadrant 3 by first following steps 1 and 2 in section IV-A to deploy the camera (with some intentionally nearer to the border to simulate less-careful positioning). To obtain variations of camera base orientation, steps 3 and 4 of section IV-A were then repeated four to six times while keeping the camera base's position unchanged and an image was captured after each repetition. A total of 73 real images were captured from unique camera poses. To illustrate the ability of our scan path planning method described in section V, we used one of the camera poses predicted by ICSC-PoseNet using a real image captured from quadrant 3 as input to generate its scan path and capture scan images after initialisation.

C. Network Implementation and Experiments

We implemented two networks (PoseNet+ [21] as baseline and our ICSC-PoseNet) using TensorFlow, supported by a NVIDIA RTX Turbo 2080Ti GPU. We fine-tuned the networks pre-trained on ImageNet [36] to leverage on transfer learning and used Xception [33] (a state-of-the-art architecture) as their base architecture. All input images are normalised such that all pixel intensities range from -1 to 1. We optimised both networks with ADAM [37] using default parameters at a learn rate of 10^{-4} and a batch size of 25. We adapted the implementation [38] of multi-task loss introduced by Kendall *et al.* [39] for PoseNet+'s and our proposed loss function. To compare performance of the two networks, each network was separately fine-tuned using synthetic images generated at 60° (Syn60) or 72.5° (Syn725) HFOV and tested on synthetic and real images obtained with the same camera HFOV. To evaluate fine-tuning on a small set of real images, we also fine-tuned each network on real images extracted from the real dataset and tested on the remaining real images not used during fine-tuning. Each network was fine-tuned (from the pre-trained weights) several times between 50-200 epochs on the synthetic datasets, and 200 epochs on the real dataset. We use the best performance of each network for evaluation and present all results in Table I with their experimental details.

VII. RESULTS AND DISCUSSION

We discuss the performance of our proposed method in five aspects: (a) deploying a PTZ camera within proposed boundaries; (b) sim-to-real camera pose estimation; (c) improvements due to geometric loss component; (d) large synthetic dataset vs small real dataset; and (e) scan path planning and localisation of scan images to demonstrate our proposed workflow. Table I shows the results from our proposed network when fine-tuned and tested on both synthetic and real datasets, with a comparison with PoseNet+ [21].

Expt. (HFOV)	Fine-tune on (images)	Test on (images)	Error	PoseNet+ [21]	ICSC-PoseNet (Ours)
1 (60°)	Syn60 (4000)	Syn60 (300)	Median	0.067m, 0.60°	0.063m, 0.50°
			RMSE	0.10m, 0.75°	0.085m, 0.63°
2 (72.5°)	Syn725 (4000)	Syn725 (300)	Median	0.12m, 0.56°	0.050m, 0.36°
			RMSE	0.12m, 0.63°	0.065m, 0.39°
3 (60°)	Syn60 (4000)	Real Scene 1 (28)	Median	0.292m, 1.25°	0.217m, 0.73°
			RMSE	0.312m, 1.44°	0.237m, 0.88°
4 (72.5°)	Syn725 (4000)	Real Scene 2 (15)	Median	0.20m, 2.51°	0.14m, 1.78°
			RMSE	0.267m, 2.74°	0.16m, 1.94°
5 (72.5°)	Syn725 (4000)	Real Scene 3 (30)	Median	0.16m, 1.26°	0.18m, 1.09°
			RMSE	0.19m, 1.40°	0.20m, 1.13°

TABLE I: Comparison of our proposed ICSC-PoseNet against PoseNet+ [21] for all scenes when fine-tuned and tested on different synthetic (Syn60, Syn725) or real (Real Scene 1,2,3) images, and with different camera HFOV (60°, 72.5°). Each Real Scene contains an A320 with different paintwork, background and lighting. Their performance are evaluated by median and root-mean-square error (RMSE) of position and orientation with the lowest error for each scene in bold and blue. Our proposed method achieves lower error when tested on all synthetic and real scenes, except for Real Scene 3 where the difference in position error is marginal. For reference, the distance between numbered frames of an A320, used by operators to locate defects on the fuselage, is about 0.53 m and improving localisation error helps reduce the chance that detected defects are wrongly tagged to an adjacent frame.

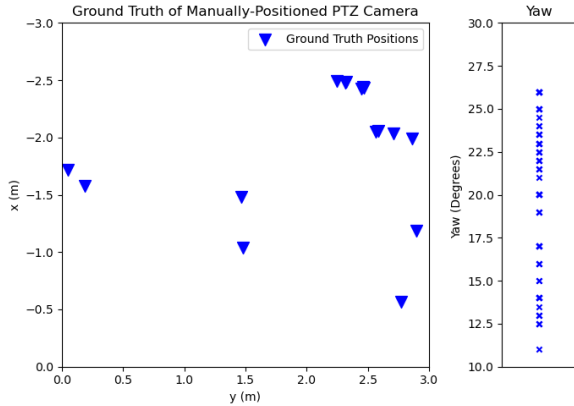


Fig. 10: Spread of ground truth 2D positions (left) and orientations (right) of real images captured by the PTZ camera at a height (z) of between 6.25-7.25 m, following steps proposed in section IV-A. All ground truth lie within our proposed boundaries (3 m x 3 m and 10°-30°), demonstrating that the proposed use of aircraft features as landmarks to guide the initial manual set up of the PTZ camera within these boundaries is feasible. A total of 73 images with unique combinations of position and orientation are collected and used for evaluating localisation accuracy.

A. Deploying a PTZ Camera Within Proposed Boundaries

All real images are obtained from manually setting up the PTZ camera using the steps proposed in section IV-A. Fig. 10 shows the spread of ground truth of all real images obtained and we find that they all lie within our proposed boundary, suggesting that the proposed use of aircraft features (windows and pylons) as landmarks to guide the initial manual set up of the PTZ camera's position and orientation is feasible. This is an important step since the use of DCNN for CPE performs best within a pre-defined range of predictions (for both position and orientation) included in the fine-tuning dataset, as deep pose estimators underperform in the task of generalising to unseen scenes [16]. However, this is only useful under the assumption that consistent deployment of the PTZ camera within the proposed boundary is feasible in the first place. We show that we can manually position and orientate our PTZ camera within the same proposed boundary and pan range used to generate our synthetic images.

B. Sim-to-real Camera Pose Estimation

Our network is able to estimate a PTZ camera's pose relative to an Airbus A320 using a single image as input

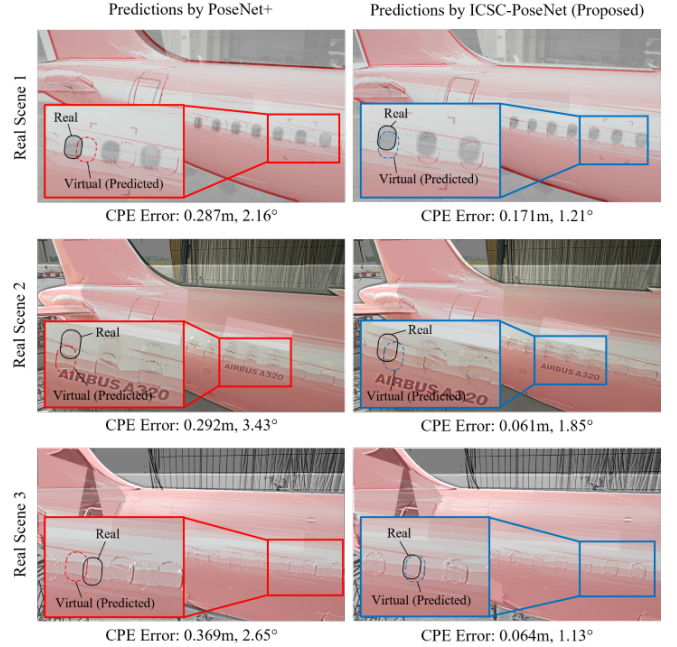


Fig. 11: Example predictions extracted from different scenes, each with an A320 with different paintwork, background and lighting. Features that are sensitive in nature are covered to protect the airlines' privacy. Virtual images captured from predicted camera poses are overlaid in red onto their respective real input image. ICSC-PoseNet (proposed) achieves lower camera pose estimation (CPE) error and results in better overlap compared to those by PoseNet+ [21] (larger offsets, more visible in the enlarged regions).

and without fine-tuning on any real images. This is achieved without any knowledge of the scene other than the aircraft's 3D model, and without any additional infrastructure. Our network (ICSC-PoseNet), when fine-tuned on only synthetic images, obtains median and RMSE prediction errors of less than 0.24 m and 2° across all real scenes (Table I) which is sufficient for initialisation given the scale of the aircraft. Interestingly, fine-tuning PoseNet+ using a limited set of 36 real images from Real Scene 1 & 2 results in higher median and RMSE error, both 0.50m and about 7°, when tested on 9 unseen real images from the same scenes. This further supports the use of synthetic images over real images in this application, especially given the difficulty of obtaining real images in the first place. Fig. 11 shows sample synthetic images captured from their predicted camera poses overlaid onto their respective real input images and we observe

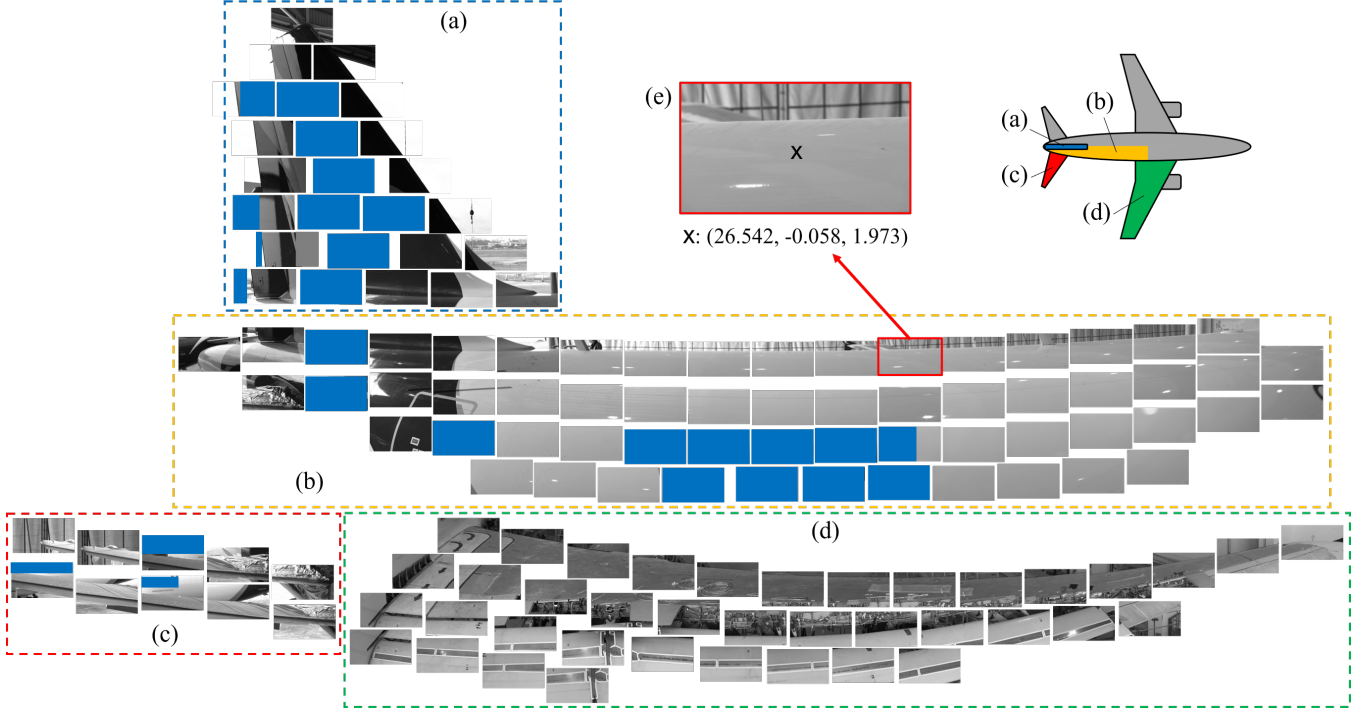


Fig. 12: A full set of scan images captured from quadrant 3 at 10X zoom using our proposed workflow, after initialisation using camera pose estimation via ICSC-PoseNet. All images have been converted to greyscale and certain parts of the image have been covered in blue to protect the airline’s privacy. The images are organised into a montage and grouped into four sections: (a) tail, (b) fuselage, (c) stabiliser, and (d) wing (in a partially-opened state for maintenance). All scan images are localised as part of our workflow and labelled with their centre-pixel’s Cartesian coordinates w.r.t. an aircraft coordinate system. (e) is an example of a scan image with its position label.

generally good overlap of aircraft outline. Our results show the network’s ability to extract relevant aircraft features from the randomised textures in the synthetic dataset and match their scale and position with the real input images to regress a camera pose, demonstrating successful sim-to-real transfer for this task.

C. Improvements Due to Geometric Loss Component

With reference to Table I, our network achieves lower median errors and RMSE as compared to PoseNet+ for both Experiments 1 & 2 that test on only synthetic images at different HFOV. As the synthetic images used for fine-tuning and testing belong to the same domain, the results show that our proposed loss function improves camera pose estimation when there is no domain gap. Our network also out-performs PoseNet+ when tested on all real scenes, except in Real Scene 3 where the difference in their position errors is marginal (about 10 percent) and argue that it is likely due to sim-to-real differences of that scene. Example predictions in Fig. 11 demonstrate that ICSC-PoseNet improves CPE accuracy and achieves smaller offsets between real and predicted positions of aircraft features. For comparison, an A320’s window-to-window distance is about 0.53 m, which is also the distance between frame numbers used by operators to locate features and defects on the fuselage. Any reduction in localisation error is beneficial for defect detection systems as it reduces the chance that a detected defect is wrongly tagged to an adjacent frame. We conclude that our proposed additional component in the loss function which geometrically relates

the predicted position and orientation improves camera pose estimation accuracy in our task.

D. Scan Path Planning and Localisation of Scan Images

To demonstrate the result of our scan and the successful completion of our proposed workflow, a total of 134 scan images are captured at 10x zoom from one instance of CNN-based initialisation within quadrant 3. We group the scan images into their respective sections (i.e. tail, fuselage, stabiliser, and wing) and organise them into a montage shown in Fig. 12. As intended by our algorithm, we observe a small overlap between images to reduce the chance of missing areas while keeping the total scan images low for efficiency. Every image was captured from a computed set of camera pan and tilt values and labelled with their respective Cartesian coordinates on the aircraft’s surface (retrieved from their respective elements (x_{ij}, y_{ij}, z_{ij}) in (6)). This achieves localisation of every scan image and can be used to localise and label surface defects during visual inspection tasks.

VIII. CONCLUSION

We demonstrate camera pose estimation with respect to an aircraft, as well as a workflow for scan path generation and image localisation. Unlike existing methods, this is easy to deploy and achieved without additional infrastructure, physical contact with or prior access to a real aircraft. We are successful in sim-to-real transfer for our CPE task, by fine-tuning a DCNN with synthetic images generated with domain

randomisation. We also show that providing a geometric relationship between the predicted position and orientation as an additional component in the loss function can improve pose estimation performance and the localisation of aircraft features in images. Future work can explore improving the initial pose estimate by using multiple input images captured from the same PTZ camera to benefit from spatio-temporal information, as well as sensor fusion with other sensor data such as from a LiDAR.

ACKNOWLEDGMENT

This research is supported by ST Engineering Aerospace as part of a project with the Civil Aviation Authority of Singapore to develop a GVI system for detecting damage to the exterior of aircraft due to lightning strikes.

REFERENCES

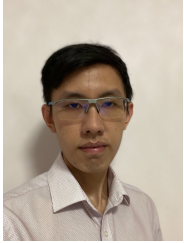
- [1] X. Oh *et al.*, "Initialisation of autonomous aircraft visual inspection systems via cnn-based camera pose estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, Conference Proceedings, pp. 11 047–11 053. [Online]. Available: <https://ieeexplore.ieee.org/document/9561575/>
- [2] P. Umberto and P. Salvatore, "Preliminary design of an unmanned aircraft system for aircraft general visual inspection," *Electronics*, vol. 7, no. 12, p. 435, 2018.
- [3] J. Gu, C. Wang, and X. Wu, "Self-adjusted adsorption strategy for an aircraft skin inspection robot," *Journal of Mechanical Science and Technology*, vol. 32, no. 6, pp. 2867–2875, 2018.
- [4] I. Jovančević, S. Larnier, J.-J. Orteu, and T. Sentenac, "Automated exterior inspection of an aircraft with a pan-tilt-zoom camera mounted on a mobile robot," *Journal of Electronic Imaging*, vol. 24, no. 6, p. 061110, 2015.
- [5] I. Jovančević *et al.*, "3d point cloud analysis for detection and characterization of defects on airplane exterior surface," *Journal of Nondestructive Evaluation*, vol. 36, no. 4, pp. 1–17, 2017.
- [6] M. Mumtaz, A. B. Mansoor, and H. Masood, "A new approach to aircraft surface inspection based on directional energies of texture," in *2010 20th International Conference on Pattern Recognition*, 2010, Conference Proceedings, pp. 4404–4407.
- [7] T. Malekzadeh, M. Abdollahzadeh, H. Nejati, and N.-M. Cheung, "Aircraft fuselage defect detection using deep neural networks," *2017 IEEE GlobalSIP*, 2017.
- [8] J. Miranda, S. Larnier, A. Herbulot, and M. Devy, "Uav-based inspection of airplane exterior screws with computer vision," in *14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2019, Conference Proceedings.
- [9] J. Miranda, J. Veith, S. Larnier, A. Herbulot, and M. Devy, "Machine learning approaches for defect classification on aircraft fuselage images acquired by an uav," vol. 11172. SPIE, 2019, Conference Proceedings, pp. 1 117 208–1–8.
- [10] A. Doğru, S. Bouarfâ, R. Arizar, and R. Aydoğan, "Using convolutional neural networks to automate aircraft maintenance visual inspection," *Aerospace*, vol. 7, no. 12, p. 171, 2020.
- [11] J. R. Leiva, T. Villemot, G. Dangoumeau, M. Bauda, and S. Larnier, "Automatic visual detection and verification of exterior aircraft elements," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, 2017, Conference Proceedings, pp. 1–5.
- [12] K. Malandrakis *et al.*, "Inspection of aircraft wing panels using unmanned aerial vehicles," in *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 2018, Conference Proceedings, pp. 56–61.
- [13] V. Tzitzilonis *et al.*, "Inspection of aircraft wing panels using unmanned aerial vehicles," *Sensors (Basel)*, vol. 19, no. 8, p. 1824, 2019.
- [14] D. Cazzato, M. A. Olivares-Mendez, J. L. Sanchez-Lopez, and H. Voos, "Vision-based aircraft pose estimation for uavs autonomous inspection without fiducial markers," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, Conference Proceedings, pp. 5642–5648.
- [15] M. Bugaj, A. Novák, A. Stelmach, and T. Lusiak, "Unmanned aerial vehicles and their use for aircraft inspection," in *2020 New Trends in Civil Aviation (NTCA)*, 2020, Conference Proceedings, pp. 45–50.
- [16] Y. Shavit and R. Ferens, "Introduction to camera pose estimation with deep learning," *CVPR*, 2019.
- [17] Y. H. Ma *et al.*, *A Method to Build Multi-Scene Datasets for CNN for Camera Pose Regression*, ser. 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality. New York: IEEE, 2018.
- [18] A. Kendall, M. Grimes, and R. Cipolla, *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*, ser. IEEE International Conference on Computer Vision, 2015, pp. 2938–2946.
- [19] S. Brahmabhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [20] N. Radwan, A. Valada, and W. Burgard, "Vlocnet++: Deep multitask learning for semantic visual localization and odometry," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4407–4414, 2018.
- [21] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," *CVPR 2017*, 2017.
- [22] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [23] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *Robotics: Science and Systems Conference (RSS)*, 2017.
- [24] D. Acharya, K. Khoshelham, and S. Winter, "Bim-posenet: Indoor camera localisation using a 3d indoor model and deep learning from synthetic images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 245–258, 2019.
- [25] D. Acharya, S. Singha Roy, K. Khoshelham, and S. Winter, "A recurrent deep network for estimating the pose of real indoor images from synthetic image sequences," *Sensors*, vol. 20, no. 19, p. 5492, 2020.
- [26] D. Acharya, R. Tennakoon, S. Muthu, K. Khoshelham, R. Hosein-nezhad, and A. Bab-Hadiashar, "Single-image localisation using 3d models: Combining hierarchical edge maps and semantic segmentation for domain adaptation," *Automation in Construction*, vol. 136, p. 104152, 2022.
- [27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.
- [28] X. Ren *et al.*, "Domain randomization for active pose estimation," *2019 International Conference on Robotics and Automation (ICRA)*, 2019.
- [29] J. Tobin *et al.*, "Domain randomization and generative models for robotic grasping," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [30] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *Robotics: Science and Systems (RSS)*, June 2018.
- [31] S. Roy, "Airbus a320neo," 2020. [Online]. Available: https://grabcad.com/library/airbus-a320neo-1/details?folder_id=7882341
- [32] C. Robotics, "Coppeliasim," 2019.
- [33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [34] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *NIPS*, 2017.
- [35] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [36] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, *ImageNet: A Large-Scale Hierarchical Image Database*, ser. IEEE Conference on Computer Vision and Pattern Recognition. New York: IEEE, 2009, pp. 248–255.
- [37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.
- [38] Y. Gal, "Multi-task learning example notebook," <https://github.com/yaringal/multi-task-learning-example/blob/master/multi-task-learning-example.ipynb>, Jun 2017.
- [39] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.



Xueyan Oh received the B.Eng. (Hons.) degree in engineering from the Singapore University of Technology and Design (SUTD), Singapore, in 2016. He is currently working toward the Ph.D. degree in Engineering Product Development (EPD) with the SUTD Engineering Product Development (EPD) Pillar, Singapore. His research interests include using deep learning for easily deployable vision-based localisation methods.



Poh Kang Tan received the B.Eng. degree in Computer Engineering from Nanyang Technological University, Singapore. He is currently working as a Software Engineer in ST Engineering.



Leonard Loh received the M.Tech. in Intelligent Systems from the National University of Singapore, Singapore, in 2022. He is a Research Assistant with the Singapore University of Technology and Design. His work involves software development & integration for various projects, such as an automated defect detection system. He is also working on developing a map merging algorithm for multiple robot exploration.



Shaohui Foong is an Associate Professor in the Engineering Product Development (EPD) Pillar at the Singapore University of Technology and Design (SUTD) and Senior Visiting Academician at the Changi General Hospital, Singapore. He received his B.S., M.S. and Ph.D. degrees in Mechanical Engineering from the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, USA. In 2011, he was a Visiting Assistant Professor at the Massachusetts Institute of Technology, Cambridge,

USA. His research interests include system dynamics & control, nature-inspired robotics, magnetic localization, medical devices and design education & pedagogy.



Pei Lin Pearlín Toh received the B.Eng. degree in mechanical and production engineering from Nanyang Technological University, Singapore, in 2003. She is a PMP certified Programme Manager with ST Engineering Aerospace Ltd, specialising in Unmanned Air Systems solutions.



Zhong Bao Andy Koh received the B.Eng. degree in Aerospace Engineering from the Nanyang Technological University, Singapore in 2014. He is currently working on UAV systems in ST Engineering Aerospace Ltd.



U-Xuan Tan (Member, IEEE) received the B.Eng. and Ph.D. degrees from Nanyang Technological University, Singapore, in 2005 and 2010, respectively. From 2009 to 2011, he was a Postdoctoral Fellow with the University of Maryland, College Park, MD, USA. From 2012 to 2014, he was a Lecturer with the Singapore University of Technology and Design, Singapore, where he took up a research intensive role in 2014 and has been promoted to Associate Professor since 2021. He is also holding a Senior Visiting Academician position at

Changi General Hospital. His research interests include mechatronics, on-site robotics algorithm, sensing and control, sensing and control technologies for human-robot interaction, and interdisciplinary teaching.



Kow Leong Ng received the B.Sc. degree in Computing and Information Systems from the University of London, London, United Kingdom. He is currently a Software Engineer with ST Engineering Aerospace Ltd, Singapore.