ISSN 1648-7788 / eISSN 1822-4180

VILNIUS TECH
Vilnius Gediminas Technical University

AVIATION

2023
Volume 27
Issue 4
Pages 248–258
https://doi.org/10.3846/aviation.2023.20588

DEVELOPING A COMPUTER VISION BASED SYSTEM FOR AUTONOMOUS TAXIING OF AIRCRAFT

Prashant GAIKWAD, Abhishek MUKHOPADHYAY ✉, Anujith MURALEEDHARAN, Mukund MITRA, Pradipta BISWAS

Indian Institute of Science, Bengaluru, Karnataka, India

Article History:
received 7 November 2023
accepted 21 November 2023

Abstract. Authors of this paper propose a computer vision based autonomous system for the taxiing of an aircraft in the real world. The system integrates both lane detection and collision detection and avoidance models. The lane detection component employs a segmentation model consisting of two parallel architectures. An airport dataset is proposed, and the collision detection model is evaluated with it to avoid collision with any ground vehicle. The lane detection model identifies the aircraft's path and transmits control signals to the steer-control algorithm. The *steer-control* algorithm, in turn, utilizes a controller to guide the aircraft along the central line with 0.013 cm resolution. To determine the most effective controller, a comparative analysis is conducted, ultimately highlighting the Linear Quadratic Regulator (LQR) as the superior choice, boasting an average deviation of 0.26 cm from the central line. In parallel, the collision detection model is also compared with other state-of-the-art models on the same dataset and proved its superiority. A detailed study is conducted in different lighting conditions to prove the efficacy of the proposed system. It is observed that lane detection and collision avoidance modules achieve true positive rates of 92.59% and 85.19%, respectively.

Keywords: autonomous taxi, lane detection, lane navigation, object detection, collision avoidance, airport dataset.

✉Corresponding author. E-mail: *abhishekmukh@iisc.ac.in*

Introduction

In the current aviation system, aircraft rely on autopilot systems during flight, while the Instrument Landing System (ILS) guides approaching and landing on runways (Acharya, 2014). However, the taxi phase heavily relies on manual operations, inspections, and communication between pilots and air traffic controllers (ATCs). Many issues can be observed in the current taxiing system, like low visibility conditions drastically reducing ground control capacity (Hakkeling-Mesland et al., 2010), a large amount of fuel consumption due to prolonged taxi operations (Daidzic, 2017), and so on. To overcome these issues, autonomy in taxiing can be an option. Understanding the significance of automation in this phase can be aided by relevant statistics. According to Boeing, the taxi phase ranks the second highest risk for accidents during flight phases (Boeing, 2007). In 2017, a report by the aviation safety network revealed that ground operations resulted in 39 fatalities despite being the safest year for commercial airlines (BBC News, 2018). Airbus predicts that air traffic will double over the next 20 years, leading to increased accident risks and a higher workload for crew members (Airbus Aircraft, 2022). Implementing an automatic aircraft control system would reduce crew workload, thus minimizing the risk of accidents and improving surface traffic control. This paper proposes a machine vision-based autonomous taxiing system following Federal Aviation Administration (FAA) rules. According to FAA guidelines (Zhang et al., 2020), the following essential measures should be taken into consideration:

1. Pilots confirm the route sent by Air Traffic Control (ATC) and stop the aircraft if they find any unwanted objects in the same taxiway.
2. The pilot should ensure the cockpit is over the center line.
3. The pilot should obey the speed limit and stop the aircraft before entering a runway.
4. Detection of the lane and other entities to be accomplished by the system.
5. Reliable sensing devices should be installed to avoid unwanted situations.

This paper addresses these problems and criteria by introducing a computer vision-based end-to-end autonomous taxiing system (Figure 1). This system includes lane detection followed by a lane navigation algorithm to keep the aircraft over the center line. The collision avoidance module is used to stop the aircraft to avoid collision with any ground vehicles detected in line with the aircraft. The proposed autonomous aircraft taxiing system aligns with

FAA guidelines by autonomously confirming routes and halting upon detecting obstacles, ensuring aircraft alignment and adherence to speed limits via lane detection, fulfilling the requirement of entity detection, and employing reliable computer vision-based models validated across various lighting conditions for enhanced safety during taxiing. In this context, TurtleBot3 (2023) is used as a replacement for actual aircraft. The contributions of this work are as follows.

- A lane navigation algorithm to manoeuvre an aircraft using the lane and light system present in the airport.
- An airport dataset containing real-world and synthetic data consisting of a novel class of road users found in airport scenarios.
- The working of the proposed system in a prototype taxiway (Figure 2) between 94 Lux and 2500 Lux values was experimented to test.

This paper is structured as follows. Section 1 explains the literature review on different parts of the proposed system. Section 2 explains the workings of the proposed system, followed by datasets prepared for object detection, which are explained in detail. The experimental setup and results are discussed in detail in Section 3, followed by a discussion and conclusions in Sections 4 and the last section, respectively.

1. Literature survey

1.1. Aircraft taxi

Various classical techniques are currently used for taxi guidance, including methods like Follow-me trucks, docking guidance systems, and batsmen. In 2001, Dow (2003) proposed an airport ground navigation system that uses aircraft tugs and a system for centralized positive control of all aircraft ground movements. The physical position of each tug and its associated aircraft is precisely tracked and relayed to a central control system. The central control system handles routing, schedules, and movement within the airport. Similarly, LeBlanc (2001) proposed an aircraft towing vehicle system where a towing tractor vehicle facilitates the aircraft's movement without requiring the aircraft's jet engines. The towing tractor can then be controlled by the pilot or remotely. Both of the previous approaches work on the assumption that the airport will be well equipped with the necessary equipment like pushback tugs, towing tractors, etc. However, necessary installed equipment might only be the case for some airports. A better approach to achieving automatic taxing would be to use the onboard systems already present on the aircraft. One such approach was studied by Cheng et al. (2001), using a state-of-the-art nonlinear control system based on feedback linearization designed for a detailed B-737 aircraft taxi model. This study focuses on improving aircraft surface navigation through automation. Other studies, like the one done by Zammit and Zammmit-Mangion (2014), focused on creating a controller for the precision navigation of aircraft during its taxing phase. Zhang et al. (2020) proposed a multi-layer architecture for auto taxiing. They mainly focused on path planning and control algorithms. They tested the working of the algorithm using an X-plane flight simulator. Hakkeling-Mesland et al. (2010) proposed a concept of autonomous taxiing in low visibility conditions, keeping cabin crew in the loop. They proposed different taxi display systems and alert Levels to evaluate the concept. Liu and Ferrari (2019) proposed the taxiing problem as a hybrid model combining a vision-based approach, obstacle avoidance, and feedback control theory. They combined this information with airport maps and ATC commands for computing motion plans.

1.2. Lane detection

An essential part of autonomous taxiing is the machine's perception of the environment, and lane detection is one of its key components. There is a plethora of work for detecting lanes. Various lane detection models have been developed based on classical computer vision and deep learning. Sun et al. (2006) proposed a method using the HSI color model for lane-marking detection, HSILMD, among classical computer vision techniques. In HSILMD, full-color images are converted to HSI color representation, and then thresholds of intensity and saturation are selected accordingly. Another classical computer vision model was presented by Kang et al. (1996), where they presented a method by combining the Hough transform and the "active line method" (ALM). The Hough transform is used to obtain an initial position estimation of the lane boundaries on the road. The lined snake – ALM – then improves the initial approximation to an accurate configuration of the lane boundaries. Deep learning-based lane detection models can broadly be categorized as anchor-based, row-wise, parametric-based, and segmentation-based. Among all these techniques, semantic segmentation-based approaches achieved impressive performance in accuracy and efficiency. Pizzati et al. (2020) took the deep learning approach. They presented an end-to-end system for lane boundary identification, clustering, and classification based on two cascaded neural networks that run in real time. Out of the two cascaded neural networks, the first is used to segment out the lanes, whereas the second is used to classify the detected lane type.

1.3. Object detection

Object detection is another essential part of an autonomous vehicle that must be integrated with lane detection for a safer driving experience. Traditional object detection techniques like Histogram Of Oriented Gradients (HOG) (Dalal & Triggs, 2005) and deformable Part-based Model (DPM) (Felzenszwalb et al., 2009) showed robust results for person detection. More recently, deep learning-based

approaches outperformed the previous models regarding accuracy and latency. One of these approaches is Regions with CNN features (RCNN) (Girshick et al., 2015). RCNN is a two-stage detector where the first stage uses techniques like Selective Search (Uijlings et al., 2013) to give proposals of where an object might be in the image, and the second stage uses CNN to classify the object. Since classification happens on each of the proposals given by the first stage, RCNN is slow in detection. Variants of RCNN like Fast RCNN (Girshick, 2015) and Faster RCNN (Ren et al., 2015) were developed to overcome this drawback. These variations are significantly faster than RCNN but still unsuitable for real-time applications. Another popular deep learning-based approach is "You Only Look Once" (YOLO) (Redmon et al., 2016), a single-stage detector that is extremely fast and can reach a speed of up to 155 fps. However, YOLO needs better localization accuracy, especially for small objects. Subsequent versions of YOLO solve this problem to some extent. Semantic segmentation has many applications in medical imaging for detecting organs, vessels, or cells and in autonomous driving for detecting traffic signs, pedestrians, or lane detection. SegNet (Badrinarayanan et al., 2017) is the first-of-its-kind segmentation model used for road scene understanding.

In summary, existing literature suggests that most systems involve crew intervention or follow-me vehicles. Zhang et al. (2020) introduced a software architecture for auto-taxiing, reliant on inputs like ATC-provided routes and airport maps, which becomes challenging during low visibility scenarios where ATC input might be unavailable. The proposed system navigates under challenging conditions using a lane navigation algorithm independent of prior map information or ATC inputs. Mukhopadhyay et al. (2023) presented a lane navigation algorithm for semi-autonomous vehicles using PD controllers but needed more clarification on controller selection. This work conducted a study comparing four controller algorithms used in automotive, aiming to determine the most effective one. Similarly, Liu and Ferrari (2019) had an approach close to the proposed method, formulating the problem within a simulation tool, a controlled environment. This work introduces a lane and obstacle detection algorithm benchmarked against state-of-the-art models before implementation in this system. The evaluation of the proposed system spans real-world scenarios encompassing various lighting conditions for robustness assessment.

2. Proposed approach

This work proposes a state-of-the-art lane detection algorithm to get the lanes' coordinates based on images from the camera mounted in the aircraft. These coordinates are then fed into the navigation algorithm, which generates the control signal for steering the aircraft. Object detection also works in parallel with the lane detection algorithm. It takes the same feed from the camera and passes it to the object detection module. Stopping control is generated once an object is detected within a certain distance. These controls are then fed to the steer control algorithm, which controls the vehicle through a Robot Operating System (ROS) with the help of a controller. Figure 1 shows the working of the proposed system. These proposed methods are detailed below.

2.1. Navigation algorithm

The lane navigation algorithm is built on a lane detection model that can detect lanes in unconstrained road conditions under different ambient lighting conditions. The lane detection model is a fusion of encoder-decoder and dilated convolution architecture. Finally, a weighting mechanism is used to fuse information between two branches. It is hypothesized that each branch might perform at a varied degree of accuracy for different images, and δ and γ resemble the confidence scores for each

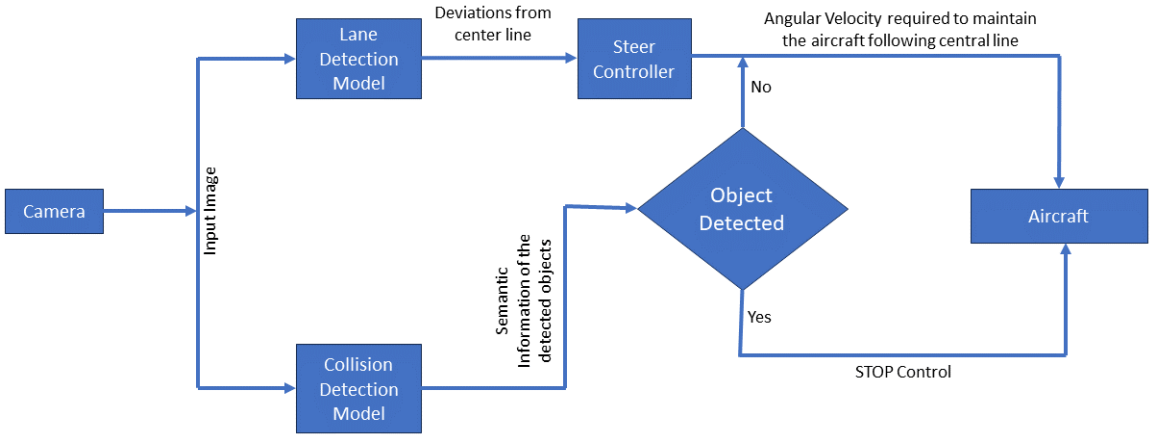


Figure 1. Schematic diagram of the proposed system

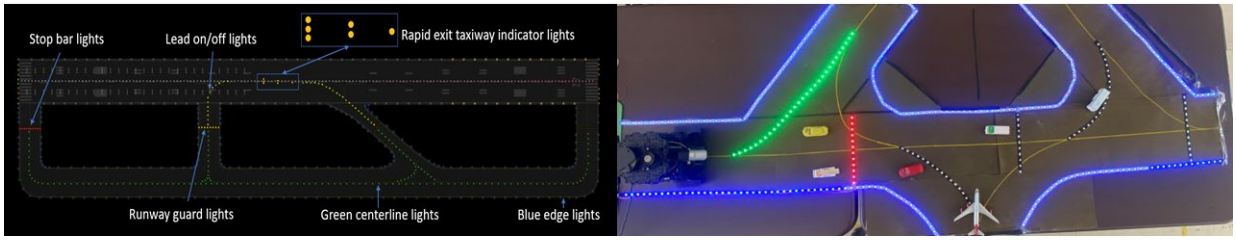


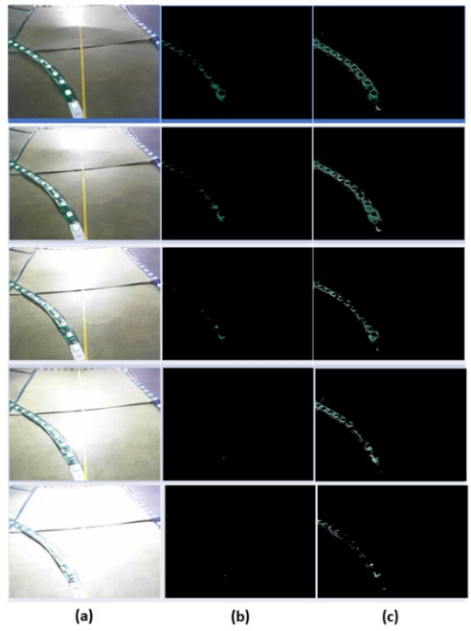
Figure 2. Setup of the taxiway created in the laboratory space

model's prediction. A separate branch of convolutional and fully connected layers takes the input image to predict δ and γ. The feature fusion is formulated as:

$$O(x,y) = \delta \times model1(x,y) + \gamma \times model2(x,y), \quad (1)$$

where, $O(x, y)$ is the feature map obtained by fusing the final pixel-wise classification layer of encoder-decoder ($model1(x, y)$) and dilated convolution architecture ($model2(x, y)$). δ and γ contribute to a weighted sum in this architecture, combining outputs from two branches. The accuracy of each block varies across images, with δ and γ reflecting confidence scores for the models' predictions. A distinct convolutional and fully connected layer branch is trained to predict δ and γ based on input images. The model outperformed other lane detection models when tested on various lighting and road conditions. The model is also reported to work in real-time (30 FPS). Part of the work reported here appeared in Mukhopadhyay et al. (2022b). Figure 4 shows the architecture of the lane detection model. In a taxiway, along with a conventional lane, an aircraft must follow a green light-colored lane to reach the destination (Figure 2). In the case of green light detection, the navigation algorithm uses a light intensity-based color segmentation algorithm.

An extensive investigation by Mukhopadhyay et al. (2019) explored diverse color space models under varying lighting conditions. Their study provided valuable insights, leading us to select the HSV color space model. The performance of color segmentation algorithm varies based on the ambient lighting conditions present across different times of the day and in various lighting scenarios. To achieve this, images were collected at different times throughout the day to capture a wide spectrum of lighting intensities. Following that, the mean grayscale value was computed for each set of collected images, establishing a mapping table that correlates HSV ranges with these mean grayscale values. This mapping enables the proposed system to dynamically select the optimal HSV range for real-time image segmentation. Figure 3 elucidates the advantages of this adaptive HSV masking technique. Notably, this analysis revealed that the mean intensity of the images falls within the range of 121 to 243, signifying both low and bright lighting conditions. This innovative approach has allowed us to create a robust system capable of effectively detecting lanes and lighting conditions across a broad spectrum of weather and lighting scenarios.



Note: Here the mean intensity of the input images is 121, 151, 181, 211, and 243 respectively.

Figure 3. Comparison between constant and varying HSV masking process in different ambient lighting conditions: (a) – input image; (b) – constant HSV range mask; (c) – varying HSV range mask

Once the center yellow lane or green lanes are detected, control points are extracted, and a curve-fitting algorithm is used to fit those points. Here, control points are the points detected as lane or green light by the respective algorithm. The curve-fitting algorithm uses the non-linear least square method to minimize the object function, as shown in Equation (2).

$$S = \sum_i W_{ii} \left(y_i - \sum_j X_{ij} \beta_j \right)^2, \quad (2)$$

where, S is the is the objective function, W_{ii} is the weights assigned to each term in the summation, y_i is the y coordinate of the ith point, X_{ij} is a vector of combination of various powers of x coordinate of the i-th point, and β_j is the vector of curve parameters which are needed to be found through iteration. This curve fitting algorithm gives the equation of the middle lane which helps in finding steer-bias. In this context, "Steer-Bias" is the distance

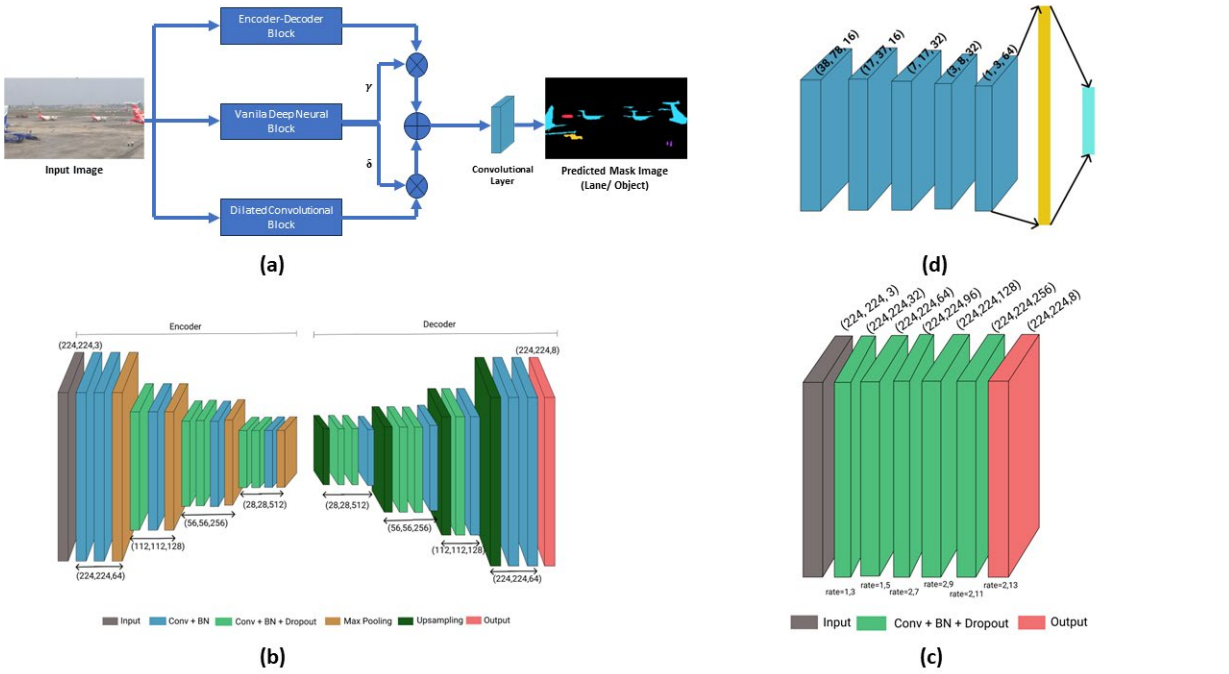


Figure 4. Lane and object detection model. Anticlockwise: (a) – overall block diagram of the lane and object detection model; (b) – encoder-decoder block; (c) – dilated convolutional block; (d) – vanila architecture for predicting δ and γ as two weights to give weightage to different segmentation blocks

between the center of the camera and the middle lane in the x direction is detected. If the center lane is towards the left of the camera's center, the error is considered positive and vice versa. The algorithm also takes care of controls regarding stopping the aircraft in corner cases where no lane is detected, or lanes come to an end.

2.2. Collision avoidance algorithm

The collision avoidance module helps detect any ground vehicle within an unsafe distance and stops the aircraft from avoiding fatal accidents. The aircraft should stop by itself if it finds any objects within the "total stopping distance". In this context, total stopping distance (d_s) is related to (Chandigarh Traffic Police [CTP], 2022; Cox, 2014):

$$d_s = f\left(v_i, \mu\right), \tag{3}$$

where v_i is the velocity of aircraft during taxiing, μ is the coefficient of friction between aircraft tires and road. Assuming μ to be constant,

$$d_s = t_s \times v_i, \tag{4}$$

where t_s is the stopping time. The speed of the aircraft in a taxiway is generally between 20–50 km/h. If the aircraft runs at a speed of 40 km/h, the "total stopping distance" will be 22 meters (CTP, 2022). Substituting these values in Equation (4), the finding was:

$$t_s = \frac{d_s}{v_i} = 1.98\,\text{sec}. \tag{5}$$

In this case, an ultrasound sensor (HC-SR04) and an object detection module was used to avoid collision on the runway. Object detection models are used to detect the ground vehicles on the runway. The ultrasonic sensor is used only to measure the distance (d_s) of those detected vehicles and to send the stop control to the aircraft to stop it. For object detection, the lane detection model is transformed into a multi-class semantic segmentation model for detecting airport ground objects (Figure 4). The architecture is modified with more layers in both encoder-decoder and dilated convolutional branches. It uses the exact weighting mechanism as described in Equation (1). Finally, to address dataset imbalance, the categorical cross-entropy loss function Equation (6) is deployed.

$$L\left(y_i, \hat{y}_i\right) = -\sum_{i=1}^{C} y_i \log \hat{y}_i, \tag{6}$$

where, y_i is 0 or 1, indicates whether class label i is the correct classification out of C classes. This object detection model is evaluated on an airport dataset and was compared with other state-of-the-art models to find its superiority.

2.3. Airport dataset preparation

Preparing the dataset was one of the challenging parts of this work. There is no publicly available dataset desired for validating the object detection model. Hence, the problem is solved bi-directionally, i.e., dataset preparation by collecting images from the real world and generating synthetic data. The real-world dataset is prepared

from videos available on the Internet. These videos have a CC BY license to use them for non-commercial purposes. Initially, every 10th frame is extracted from the videos and stored for labeling. A virtual airport environment was created using the Unity engine to generate the synthetic dataset. The environment followed a real airport scenario, containing airplanes, persons, and other ground vehicles. A video was recorded in the environment, followed by extracting frames. The Computer Vision Annotation Tool (CVAT) was used to manually annotate target classes. The dataset has five classes of objects, including airplanes, buses, cars, other vehicles, and persons. Other vehicles include dollies, pushback tugs, and tractors. Finally, RGB images and corresponding semantic segmented images are populated after labeling the images. The synthetic dataset is added to the train data. In the post-processing steps, labeled images are generated by assigning gray color values from 1 to 5. The total number of instances for each class is described in Table 1. The real-world training dataset is divided into training and validation with a ratio of 80:20. The object detection model is trained with this airport dataset before deploying it to the proposed system. Both CSV and instance-segmented images were generated for other kind of object detection models. CSV labels contain the class label and bounding box details ($[class, x_{min}, y_{min}, x_{min}, y_{max}]$) of each instance in the image. Figure 5 depicts both real-world and synthetic images alongside their respective semantic segmentation labels.

Table 1. Overview of the airport dataset

Dataset	Airplane	Bus	Car	Other vehicles	Person
Train	811	260	278	342	477
Val	221	98	86	93	143
Test	47	18	27	42	51
Synthetic	179	58	47	169	56

2.4. Steer control algorithm

The mobile agent uses a controller to navigate the lane autonomously. The output from the lane navigation algorithm serves as input for the controller. The forward velocity of the mobile agent is permanently fixed during the execution. The required steering input to move along the curved lane is generated from the controller based on the "Steer-Bias". The steer control algorithm is described in Algorithm 1. Based on the application and simplicity of implementation, four controllers were selected to evaluate the performance of this application. These are the PD controller, Sliding Mode Controller, Linear Quadratic Regulator, and Stanley Controller. The working of each controller is described in the following paragraphs.

Proportional Derivative Controller (PD) is a commonly used controller which relates controller output to current error and the rate of change of error. The control law for PD controller is given below:

$$u = Kp \times (Steer - Bias) + Kd \times (Steer - Bias - Previous\,Bias), \quad (7)$$

where, Kp indicates the proportional gain and Kd indicates the derivative gain.

Sliding Mode Controller (SMC) comprises of a control law which tries to keep a system on a sliding surface, where the control action switches its sign based on system's state and thereby achieving desired system behavior and stability. The control law has following structure:

$$u(x,t) = -K \cdot sign(s(x,t)), \quad (8)$$

where $u(x,t)$ is the controller output, K is the positive controller gain, and $sign(s(x,t))$ is the switching function. Here sliding surface $s(x,t)$ is determined by the Steer-Bias values. The switching function is used to determine whether the system state is on the sliding surface. The chattering effect was observed when this controller was used for navigation of the mobile robot.

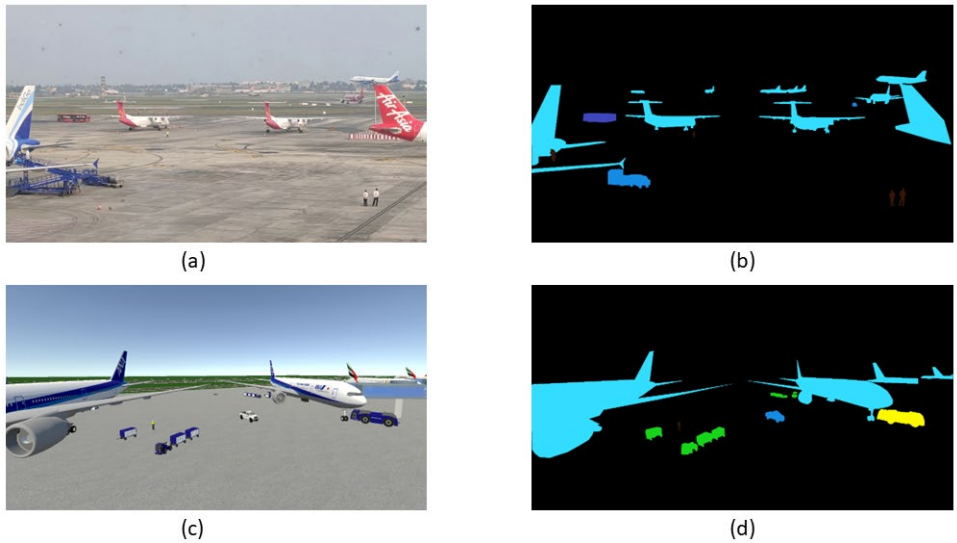


Figure 5. Sample images from the airport dataset: (a)–(b) real-world image and corresponding semantic labelling; (c)–(d) synthetic image and corresponding semantic labelling

Linear Quadratic Regulator (LQR) achieves optimal control by operating a dynamic system at a minimum cost. Considering the state-space equation of a linear time-invariant (LTI) system,

$$\dot{x} = Ax(t) + Bu(t), \tag{9}$$

where, $\boldsymbol{A}$ matrix is used to represent system dynamics and $\boldsymbol{B}$ represents control inputs influence on the system with infinite-horizon cost function given by,

$$J = \int_{o}^{\infty} \left[x^T \boldsymbol{Q} x + u^T \boldsymbol{R} u \right] dt,\ \boldsymbol{Q} = \boldsymbol{Q}^T \geq 0, \boldsymbol{R} = \boldsymbol{R}^T \geq 0, \tag{10}$$

where $\boldsymbol{Q}$ is the state cost matrix and $\boldsymbol{R}$ is the control input cost matrix. The feedback control law which minimizes the value of cost is:

$$u = -Kx, \tag{11}$$

where Steer-Bias values are conveniently used in state vector (x) and K is given by:

$$K = R^{-1} B^T P. \tag{12}$$

The Continuous-time Algebraic Riccati Equation (CARE) is used to obtain the value of P which is given by:

$$\boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T \boldsymbol{P} + \boldsymbol{Q} = 0. \tag{13}$$

Stanley controller is a geometric path tracking controller used in autonomous driving and vehicle control to navigate along a reference trajectory. It is designed to get a desired steering angle for navigation. The Cross-Track Error (CTE) is used in the Stanley controller, which represents the lateral distance between the vehicle's current position and the reference path. CTE is calculated by finding the shortest distance from the vehicle's position to the reference path. If the vehicle is exactly on the path, the CTE is zero which is exactly how steer bias value are also given in this case. Along with CTE Heading Error is also used in the control law. The goal is to minimize the Heading Error which measures the difference between the vehicle's current heading and the direction of the reference path. The control law is expressed as follows:

$$\delta(t) = \varphi(t) + \tan^{-1}\left\{ K \times Steer - Bias | v(t) \right\},\ \ \delta(t) \in \left[\delta_{min}, \delta_{max} \right], \tag{14}$$

where $\delta(t)$ is the desired steering angle, K is a gain parameter which controls how aggressively the controller corrects for lateral errors (CTE, which is Steer-Bias here), $v(t)$ is the vehicle's current speed and $\varphi(t)$ is the Heading Error.

3. Experimental evaluation

The proposed system is evaluated in terms of

1. error of deviation from the lane and comparison between different types of controllers,
2. accuracy of the object detection model and
3. efficacy of lane navigation and object detection in different lighting conditions.

All these are discussed in the consecutive paragraphs. The experimental setup of the system and all other equipment used in this work are discussed in detail.

Algorithm 1. Steer control algorithm

Input data: Steer-Bias/Stop control.
Output data: Set Control Inputs
Initialize Error and Previous Bias to '0'
while process is not terminated **do**
  if mode is Autonomous **then**
    if Stop control received **then**:
    Exit Autonomous and switch to Manual.
    Angular Velocity ← controller(Steer − Bias)
    Previous_Bias ← Steer_Bias
    Mode agent with Forward Velocity (fixed) and calculated Angular Velocity
    Set inputs (Forward Velocity, Angular Velocity)
end

3.1. Experimental setup

The experimental setup consists of a model following an actual taxiway along with a light and lane indicator. The setup of the runway is shown in Figure 2. The green light shows which fork to use for going to the runway or hanger and the red light is used as a guard line to stop the aircraft. In this work, TurtleBot3 (2023) is used as a replacement for actual aircraft, as mentioned before. TurtleBot3 is a programmable Robot Operating System (ROS) based mobile robot. It is used for research and product prototyping. TurtleBot3 was chosen because it is easy to control through ROS and has provisions for attaching sensors like a camera. A Microsoft Livecam is mounted in front of it to perceive the environment. The camera has 37.73° vertical field of view (FOV) and 57.88° horizontal FOV, and it can cover up to 35 cm in the frontal direction (Refer to the input image in Figure 1). Considering the speed of the TurtleBot3 of 0.1 m/s and using Equation (5), total stopping distance (d_s) is calculated as 19.8 cm. The ultrasonic sensor is triggered and halts the aircraft when it identifies any detected objects within the specified range.

3.2. Results

The accuracy of the system depends on how accurately the aircraft follow the central line. Here, to follow the central line, the center of the camera frame should be aligned with the lane. If the camera's line of sight deviates by θ radians, it must correct by d cm to align its center with the lane. Now, the minimum value of d that is d_{min} for which the TurtleBot3 can correct without overshooting can be found by knowing the minimum θ by which the TurtleBot3 can turn. If the value of d is less than d_{min}, then the TurtleBot3 will overshoot because d_{min} is the minimum distance it can compensate for. In other words, d_{min} is the resolution of TurtleBot3 in terms of deviation from the central line. The calculations for the same are described in Equations (15), (16), and subse-

quent calculations. Figure 6 is the representative diagram where P is the center point of camera frame, r is distance from middle of wheel axle of TurtleBot3 to point P, θ is the angle between perpendicular from wheel axle of TurtleBot3 and lane, d is perpendicular distance from lane to point P, and s is the arc made from point P to lane with radius r and angle θ.

$$s = r \cdot \theta; \tag{15}$$

$$d = r \cdot \theta, \tag{16}$$

$q_{\min}$ is obtained from the specification of the TurtleBot3 and r is measured as 0.00058 rads and 0.225 m respectively. Now,

$$\begin{aligned} s_{\min} &= r \cdot \theta_{\min} \\ &= 0.225 \times 0.00058 \\ &= 0.00013 \text{ m} \\ &= 0.013 \text{ cm} \end{aligned}$$

Also, for small θ, $\sin \theta = \theta$.
Hence, $d_{\min} = s_{\min} = 0.013$ cm.

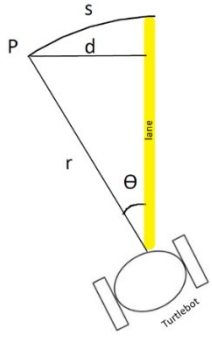
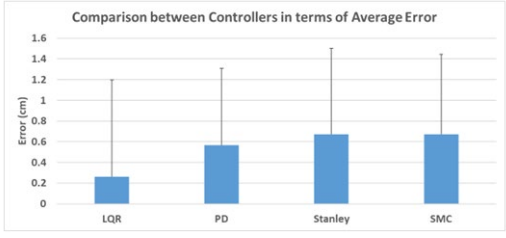
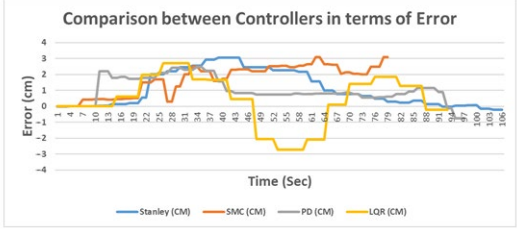


Figure 6. Representative view for minimum distance compensation calculation



(a)



(b)

Figure 7. Comparing accuracy of different controllers: (a) – average error plot across all type of paths; (b) – deviations from central line in case of left turn

Furthermore, a performance evaluation of various controller types to assess the lane-following precision, as detailed in Section 2 was conducted. In this context, the primary objective revolved around comparing the controllers' ability to effectively correct the Turtlebot's deviation from the central lane. To achieve this, three distinct paths – left, right, and central was designed (as shown in Figure 2b). The findings indicated that the Linear Quadratic Regulator (LQR) exhibited the highest accuracy, with a mean error of 0.26 cm and a standard deviation of 0.94. For a visual representation of this comparison, please refer to Figure 7a, which presents a chart illustrating the controllers' general performance. Additionally, Figure 7b shows a comparison chart specifically for the deviations of the Turtlebot in the case of left-turn scenarios.

The accuracy of the object detention model is reported as follows. The model is compared with other state-of-the-art object detection models on the airport dataset. The other state-of-the-art object detection models include two-stage (Mask RCNN), transformer-based (DeTR), and segmentation-based (FCN, UNet). The comparing measurement unit is Intersection Over Union (IoU). Figure 8 shows the comparison chart between the models. NVIDIA GeForce RTX 3070 with Max-Q Design was used for training and evaluating the model. An overall accuracy of (mean IoU) of 0.32 with speed of 9.35 FPS was reported. The performance of the model is summarized in Table 2.

Finally, the efficacy of the overall system is measured in different lighting conditions in terms of true positive rate (TPR). This study tested the system in 94, 201.67, and 2500 Lux values. In each condition, a repetitive test (3 times) was done. The focus was on the system could detect lanes and green lights while avoiding collisions. It was observed that in low light or light room conditions, the rate of lane or line detection rate was 100%. However, in high illumination, the system failed to detect the green light lane in maximum cases (44.44%). In the case of collision avoidance, the rate was lowest in low-light conditions (66.67%). The summary of the experiments is described in Table 3. Figure 9 demonstrates the real-world operation of the proposed system, in line with the explanation provided in Figure 1.

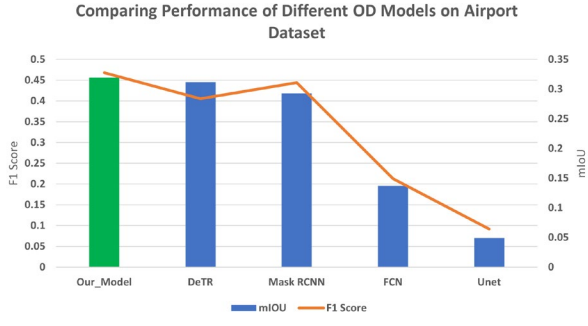


Figure 8. Comparing accuracy of different object detection models on the airport dataset (blue bars: Other models, green bar: Hybrid model)

Table 2. Accuracy analysis of the hybrid object detection model (class wise)

Classes	IoU	Precision	F1 Score
Airplane	0.527	0.73	0.672
Bus	0.263	0.42	0.381
Car	0.237	0.468	0.391
Other vehicles	0.315	0.600	0.504
Person	0.255	0.604	0.394

Table 3. Accuracy analysis of the proposed system in different lighting conditions

Lux Value	94	201.67	2500	Accuracy (%)
Lane Detection (%)	100	100	77.78	92.59
Light Detection (%)	100	100	44.44	81.48
Collision Avoidance (%)	66.67	100	88.89	85.19

4. General discussion

This paper presented a vision-guided and robust system for autonomous aircraft taxiing in the real world, consisting of autonomous navigation and collision avoidance modules. The navigation module detected the lane using a lane detection algorithm and sent control signals to the steer control algorithm. The object detection module in the collision avoidance algorithm was fine-tuned with an airport dataset proposed in this paper. In the following paragraphs, we summarize all the critical points discussed in the Introduction section.

Lane Navigation Algorithm: The lane navigation algorithm is a combination of lane and light detection methods. The lane detection algorithm uses a weighting scheme to weigh extracted features from two segmentation branches. This architecture utilizes only one weight parameter per feature extraction branch, limiting the training parameters of the network to show better performance than the individual components of the network. This architecture was helpful in terms of memory and real-time deployment. Across the lighting conditions, the model could detect lanes (the highest TPR was 100%, and the lowest was 77.78%, as shown in Table 3.

Along with lane detection, a classical computer vision approach was engaged to detect green lines to navigate the aircraft in the desired path. The classical approach is the only alternative to detecting lines, as the deep learning approach would require another data set to train on this kind of lane. The light detection algorithm showed robustness in both "normal" and "low light" conditions. However, it failed to detect green lines (TPR: 44.44%) in a few cases in high illumination conditions. This problem can be overcome in a taxiway as it automatically switches between lane and light detection algorithms. In the taxiway, green lights are placed in the gap to make the yellow lane visible. Thus, the failure mode of the light detection method can be compensated by lane detection for momentary assistance of the aircraft.

Airport Dataset: Although many datasets are available on airport scenarios for different purposes, a dataset for object detection is needed for evaluating any model on airport scenarios. This paper proposes an airport dataset for training and validating object detection models. It is well-known that preparing a dataset is always time-consuming work. Moreover, the security process in the airport makes it more challenging to collect datasets. The current part of the dataset mainly covers Indian commercial airports. It includes all novel classes of objects, including 'Pushback Tugs,' 'dollies for loose luggage,' and so on. It contains the label for bounding box, instance semantic, and semantic segmentation-based object detection model. Initially, the focus centered on creating a dataset through the collection of real-world images. However, as the dataset was small, the overall accuracy achieved by the model was 0.26 (in IoU). Hence, an alternative approach was adopted, inspired by Mukhopadhyay et al. (2021, 2022a), utilizing synthetic dataset creation to expand the dataset size. This strategy not only

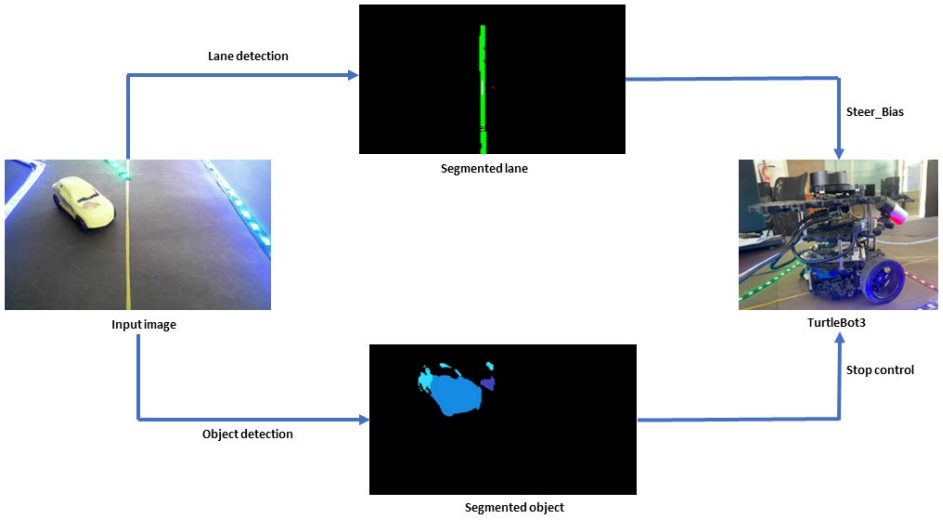


Figure 9. The working of proposed system in real world

augmented dataset volume but also allowed the capturing of object images from multiple perspectives within a VR model. This process enhances dataset diversity, a challenge in real-world data collection. Incorporating synthetic data resulted in a notable 23.08% increase in overall accuracy for the airport dataset. The dataset will be publicly available in the future with more classes like 'traffic cone,' 'de-icing vehicle,' 'airstairs,' and so on. The model's performance in airport scenarios can be seen in the supplementary video.

Working System: This paper emphasizes the end-to-end solution with state-of-the-art components. Existing systems (Liu & Ferrari, 2019) address this in a simulated control environment. In contrast, this paper shows the system's efficacy in a real-world setup. It needs manual inspection once the aircraft lands and is ready to follow the taxiway. This paper proposes an alternative to that by integrating lane and light detection algorithms, maneuvering the aircraft, avoiding any collision, and stopping at specific points. The navigation algorithm not only sends a control signal to the steer-control algorithm, but it also stops the aircraft if it comes to the end of the taxiway or lane is no longer detected. The steer control algorithm, on the other side, always helps the aircraft follow the central line. The error analysis in the result section shows that the deviation of the system can be minimized to 0.013 cm. Additionally, an in-depth comparative analysis was performed on various controllers to assess their real-time performance. Four controllers were selected based on earlier research on aircraft navigation (Ogunwa & Abdullah, 2016) and autonomous vehicles (Lee & Yim, 2023). For example, In the context of aircraft navigation, research investigates using a Linear Quadratic Regulator (LQR) controller to assist commercial Boeing 747-200 aircraft to regain its stability in the event of damage (Ogunwa & Abdullah, 2016). In this case, four controllers are compared and deployed for accurate lane navigation, similar to a study for autonomous vehicles (Lee & Yim, 2023). The findings demonstrated that the Linear Quadratic Regulator controller could correct the Turtlebot's trajectory deviations with an error of only 0.26 cm. In the case of SMC, the chattering effect contributed to the highest error compared to other controllers. Figures 7a and 7b display the controllers' performance. Figure 7b notably illustrates that both LQR and SMC swiftly navigate a left-turn trajectory, with SMC being the fastest. However, Figure 7a depicting average errors across various lane trajectories reveals that LQR maintains minimal deviation from the center lane (0.26 cm) across straight, left, and right trajectories, despite SMC's speed advantage. These results show that LQR can be helpful for accurate autonomous lane navigation.

Discussing the system's performance in real-time, the lane detection module shows promising performance in real-time speed. Although the object detection module works with 9.35 FPS, it will not be affected as the frequency of ground vehicles coming in the line of aircraft is low, and aircraft will have a maximum speed of 40 KMPH in the taxiway. Table 3 supports the claim by showing promising results in the following lane and stopping the aircraft with a TPR of 92.59% and 85.19% across the lighting conditions, respectively. Additionally, the system's real-world performance is showcased in the supplementary video.

Future work can further explore integrating LiDAR or RADAR sensors alongside conventional RGB cameras to enhance the automated system's performance for taxiing aircraft. By incorporating these additional sensors, the system can improve collision avoidance capabilities and provide more accurate real-time control in various lighting conditions. Additionally, investigating LiDAR or RADAR can enhance object detection in unconstrained environments, offering superior accuracy compared to existing models. As demonstrated in the supplementary video, the system's efficacy can be further evaluated by testing it in diverse setups, such as complex forked taxiways**.**

Conclusions

This paper presents a novel automated system for taxiing aircraft providing collision avoidance and momentary assistance. The Proposal of a computer vision-based autonomous aircraft taxiing system integrating lane and collision detection models, showcasing effectiveness in real-world settings rather than simulated environments. The system's navigation algorithm autonomously controls the aircraft's path, detecting lanes, maneuvering, preventing collisions, and halting at designated points, reducing the need for manual intervention. Future advancements could involve integrating LiDAR or RADAR sensors alongside RGB cameras to enhance collision avoidance and provide more accurate control in diverse lighting conditions. While the system performs well with an RGB camera, further exploration of LiDAR or RADAR integration could augment collision avoidance and object detection, addressing potential limitations in unconstrained environments. The system's accuracy in detecting lanes and avoiding collisions in real-world scenarios could improve airport operations' safety and efficiency, reducing the need for manual intervention during taxiing. The use of multiple sensors might impact the economic aspect of implementation; however, the potential enhancement in safety and accuracy could outweigh these considerations. The system's efficacy has been proven in varied lighting conditions; further scaling could involve testing in more complex scenarios, such as forked taxiways to evaluate its adaptability and effectiveness in intricate airport environments. The real-time performance of the proposed system can be found at https://youtu.be/VupBqvHCKPg.

Contributions

Abhishek worked on conceptualization, developing deep learning models, and writing the draft. Prashant worked on software development, dataset preparation, and writing the draft. Anujith and Mukund worked on developing and comparing the steer control algorithms, Pradipta supervised the research. Both first and second authors contributed equally to this work.

References

Acharya, R. (2014). *Understanding satellite navigation*. Academic Press. https://doi.org/10.1016/B978-0-12-799949-4.00002-6

Airbus Aircraft. (2022). *Global Services Forecast* (*GSF*) | *2022–2041*. https://aircraft.airbus.com/en/market/global-services-forecast-gsf-2022-2041

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

BBC News. (2018). *2017 safest year for air travel as fatalities fall*. https://www.bbc.com/news/business-42538053

Boeing. (2023). *Statistical summary of commercial jet airplane accidents-worldwide operations 1959–2022*. https://www.faa.gov/sites/faa.gov/files/2023-10/statsum_summary_2022.pdf

Chandigarh Traffic Police. (2022). *Safe and responsible driving*. Chandigarh Traffic Police.

Cheng, V. H., Sharma, V., & Foyle, D. C. (2001). A study of aircraft taxi performance for enhancing airport surface traffic control. *IEEE Transactions on Intelligent Transportation Systems*, *2*(2), 39–54. https://doi.org/10.1109/6979.928715

Cox, J. (2014). *Ask the captain: Making time on the taxiways*. https://www.usatoday.com/story/travel/columnist/cox/2014/11/23/airport-airplane-taxi-speed/19334661/

Daidzic, N. E. (2017). Determination of taxiing resistances for transport category airplane tractive propulsion. *Advances in Aircraft and Spacecraft Science*, *4*(6), 651–677.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR'05) (Vol. 1, pp. 886–893). IEEE. https://doi.org/10.1109/CVPR.2005.177

Dow, J. H. (2003). *U.S. Patent No. 6,600,992*. U.S. Patent and Trademark Office.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern analysis and Machine Intelligence*, *32*(9), 1627–1645. https://doi.org/10.1109/TPAMI.2009.167

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (*ICCV*) (pp. 1440–1448). IEEE. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(1), 142–158. https://doi.org/10.1109/TPAMI.2015.2437384

Hakkeling-Mesland, M. Y., Beek, B. V., Bussink, F. J. L., Mulder, M., & van Paassen, M. M. (2010). Evaluation of an autonomous taxi solution for airport operations during low visibility conditions. In *Proceedings of the 9th USA/Europe Air Traffic Management Research and Development Seminar* (*ATM 2011*). ResearchGate.

Ismail, A. H., Azmi, M. S. M., Hashim, M. A., Ayob, M. N., Hashim, M. M., & Hassrizal, H. B. (2013). Development of a webcam based lux meter. In *2013 IEEE Symposium on Computers & Informatics* (*ISCI*) (pp. 70–74). IEEE. https://doi.org/10.1109/ISCI.2013.6612378

Kang, D. J., Choi, J. W., & Kweon, I. S. (1996). Finding and tracking road lanes using "line-snakes". In *Proceedings of Conference on Intelligent Vehicles* (pp. 189–194). IEEE. https://doi.org/10.1109/IVS.1996.566376

LeBlanc, E. L. (2001). *U.S. Patent No. 6,305,484*. U.S. Patent and Trademark Office.

Lee, J., & Yim, S. (2023). Comparative study of path tracking controllers on low friction roads for autonomous vehicles. *Machines*, *11*(3), 403. https://doi.org/10.3390/machines11030403

Liu, C., & Ferrari, S. (2019). Vision-guided planning and control for autonomous taxiing via convolutional neural networks. In *AIAA Scitech 2019 Forum* (p. 0928). Aerospace Research Central. https://doi.org/10.2514/6.2019-0928

Mukhopadhyay, A., Mukherjee, I., & Biswas, P. (2019). Comparing shape descriptor methods for different color space and lighting conditions. *AI EDAM*, *33*(4), 389–398. https://doi.org/10.1017/S0890060419000398

Mukhopadhyay, A., Rajshekar Reddy, G. S., Mukherjee, I., Kumar Gopa, G., Pena-Rios, A., & Biswas, P. (2021). Generating synthetic data for deep learning using VR digital twin. In *Proceedings of the 2021 5th International Conference on Cloud and Big Data Computing* (pp. 52–56). ACM Digital Library. https://doi.org/10.1145/3481646.3481655

Mukhopadhyay, A., Reddy, G. R., Saluja, K. S., Ghosh, S., Peña-Rios, A., Gopal, G., & Biswas, P. (2022a). Virtual-reality-based digital twin of office spaces with social distance measurement feature. *Virtual Reality & Intelligent Hardware*, *4*(1), 55–75. https://doi.org/10.1016/j.vrih.2022.01.004

Mukhopadhyay, A., Murthy, L. R. D., Mukherjee, I., & Biswas, P. (2022b). A hybrid lane detection model for wild road conditions. *IEEE Transactions on Artificial Intelligence*, *4*(6). https://doi.org/10.1109/TAI.2022.3212347

Mukhopadhyay, A., Sharma, V. K., Tatyarao, P. G., Shah, A. K., Rao, A. M., Subin, P. R., & Biswas, P. (2023). A comparison study between XR interfaces for driver assistance in take over request. *Transportation Engineering*, *11*, Article 100159. https://doi.org/10.1016/j.treng.2022.100159

Ogunwa, T. T., & Abdullah, E. J. (2016). Flight dynamics and control modelling of damaged asymmetric aircraft. *IOP Conference Series: Materials Science and Engineering*, *152*(1), Article 012022. https://doi.org/10.1088/1757-899X/152/1/012022

Pizzati, F., Allodi, M., Barrera, A., & García, F. (2020). Lane detection and classification using cascaded CNNs. In *Computer Aided Systems Theory – EUROCAST 2019. Lecture Notes in Computer Science* (Vol. 12014, pp. 95–103). Springer. https://doi.org/10.1007/978-3-030-45096-0_12

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). IEEE. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, *28*.

Sun, T. Y., Tsai, S. J., & Chan, V. (2006). HSI color model based lane-marking detection. In *2006 IEEE Intelligent Transportation Systems Conference* (pp. 1168–1172). IEEE.

TurtleBot3. (2023). *TurtleBot3*. https://www.turtlebot.com/turtlebot3/

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*, 154–171. https://doi.org/10.1007/s11263-013-0620-5

Zammit, C., & Zammmit-Mangion, D. (2014). A control technique for automatic taxi in fixed wing. In *52nd Aerospace Sciences Meeting* (p. 1163). Aerospace Research Central. https://doi.org/10.2514/6.2014-1163

Zhang, Y., Poupart-Lafarge, G., Teng, H., Wilhelm, J., Jeannin, J. B., Ozay, N., & Scholte, E. (2020). A software architecture for autonomous taxiing of aircraft. In *AIAA Scitech 2020 Forum* (p. 0139). Aerospace Research Central. https://doi.org/10.2514/6.2020-0139