

# Development and evaluation of obstacle detection mechanisms for autonomous aerial systems

Ganesh Babu Loganathan<sup>1\*</sup>, Gopinath Suresh<sup>1</sup>, Harish Ragavendra Venkatesh<sup>1</sup>, Kabilan Prabakaran<sup>1</sup>, Sai Madan Saravanan<sup>1</sup>, Sathish Kumar Dhaya dharan<sup>1</sup>, Thirunavukkarasu Veeramani<sup>1</sup>, and Vibbin Chandra Elangovan<sup>1</sup>

<sup>1</sup>Department of Robotics and Automation, Rajalakshmi Engineering College, Chennai 602105, Tamil Nadu, India

**Abstract.** Autonomous robots rely heavily on avoidance algorithms and do not typically use obstacle contact detection, which limits their usefulness in congested situations. Blind spots or discretized detection locations plague current contact-detection systems, and performance is further limited by stiff platforms that only detect collisions without offering meaningful feedback. We offer a novel architecture for contact sensors that enhances autonomous navigation by detecting physical contact as a solution to these problems. The system uses a flex-sensor-equipped elastic collision platform to measure displacements in the event of a collision. Data collected by flex sensors is transformed into useful contact information by a contact-detection algorithm that were based on a neural network. By utilizing sensor data for real-time collision recovery, collision system was tested with collisions that occurred during both autonomous contact-based missions and manual flights. The testing findings showed that the system could accurately detect collision parameters and contact events estimate, even in scenarios when the parameters were changing. Future research on contact-based navigation systems can build on the suggested methodology, which offers a strong method for improving autonomous navigation in complicated surroundings.

## 1 Introduction

The ability to detect and recover from collisions is of the utmost importance in aerial robotics, particularly for drones that operate in intricate and congested situations. RGBlimp-Q integrates a continuum arm and moving mass actuation to stabilize blimps under airflow disturbances. Inspired by bird flight, it enables perching-like aerial manipulation. Tests confirmed improved resilience and novel capabilities [1]. The GATA framework performed terrain segmentation and traversability analysis from LiDAR. A shallow neural network improved ground classification. Real-world tests confirmed efficiency for delivery robots. A quadrotor with a flexible cable was modeled using PDE-ODE dynamics. Reduced-order NMPC enabled trajectory tracking for both position and cable shape.

---

\*Corresponding Author: [ganeshbabu.l@rajalakshmi.edu.in](mailto:ganeshbabu.l@rajalakshmi.edu.in)

Experiments outperformed PID-based control as observed by [2]. A mono-airfoil aircraft inspired by samara seeds was designed with a single actuator. Surrogate optimization achieved 26-minute stable hovering with high power efficiency. Results surpass existing endurance benchmarks. Inverted landing was studied using fly-inspired optical flow behaviors. A two-stage RL policy combined flip-trigger SVM and flip-action neural network. Sim-to-real transfer enabled robust ceiling landings [3]. A 3-DOF robotic arm was enhanced with ANN for joint prediction. CNN and ANN were used for real-time obstacle avoidance. The system improved torque estimation and motion planning. Autonomous robots in hazardous environments were modeled using predictive kinematics. Position, velocity, and acceleration data enabled safe obstacle handling. A case study confirmed operational reliability [4]. This review analyzed path planning in dynamic environments with formation control. Collaborative decision-making and communication were emphasized. Gaps remain in complex and unpredictable navigation tasks. Path optimization for land robots was simulated using omnidirectional video. Obstacle detection and odometry improved accuracy and energy use. Algorithms enabled faster travel in structured spaces [5]. A guiding manager for agricultural robots was developed with laser-based weeding tools. Controllers ensured precise tracking under soil variability. Field trials showed minimal errors in crops. A metamodel MARTE: ARM-Variability was proposed for adaptive navigation frameworks. It enables runtime reconfiguration in ROS. Results support the robustness in dynamic contexts [6]. A hierarchical system combined CACOF planning with CNN detection and fuzzy Bug avoidance. Computation for assistive robots was reduced and simulations showed real-time suitability. A MANET routing method combined ACO-BPST with swarm models. RDPSO-GBA improved obstacle detection and power control. Simulations outperformed conventional methods [7]. A stereoscopic vision system estimated obstacle distances in unknown indoor areas. A bacterial-inspired algorithm supported real-time measurements. Robot tests confirmed efficiency and low cost. A modular aerial robot capable of in-air self-disassembly was introduced. Vector tilting and electromagnet-based undocking enabled full actuation. Experiments validated reconfiguration and stable control [8]. AEROM, a 6-DOF aerial manipulator with soft fingers was developed for dexterous tasks. An anti-disturbance controller improved accuracy. Physical tests confirmed the effective aerial interaction [9]. A robotic gliding blimp combined buoyant and aerodynamic lift. Internal moving mass and wings enhanced efficiency and stability. Experiments validated a new hybrid aerial platform [10]. Multi-vehicle racing was addressed with LiDAR perception and EKF tracking. Predictive planning enabled safe overtaking. Real-world trials proved effectiveness in nine-vehicle scenarios [11]. An orchard obstacle detection model enhanced YOLOv8n with CBAM and Soft DIoU-NMS. High accuracy was achieved across lighting and distance variations. Fast inference ensured field suitability [12]. A sewer robot system combined laser marker and RGB cameras for obstacle detection. It enabled precise local navigation. Performance varied with lighting and sensor setup [13]. When it comes to collision recovery, none of the current contact-detection methods are complete. Discrete contact orientation is one option, but it reduces resolution. Another is to use rigid platforms, but they only detect collisions and do not give detailed feedback [14]. In response to these shortcomings, this work presents a platform for obstacle contact detection that may be used in conjunction with current navigation systems. Two primary benefits are provided by the suggested platform. To begin with, it achieves better accuracy in collision detection by constantly monitoring collisions along the whole perimeter without discretization. The second perk is that it tracks the relative motion of the outer and inner rings during a collision and provides real-time feedback within a 2.5 cm range. In poor light, smoke or fog, vision-based navigation may not work, but this approach can help. Even when its vision is impaired, a firefighting drone's contact-detection platform allows it respond and recognize to obstacles, such as when source-seeking in a smoke-filled room.

This gets around the problems with vision-based systems and makes sure everything runs smoothly and safely. The steps taken to build a contact-based autonomous navigation stack, as well as the platform for detecting contacts datasets, and prediction algorithms for contacts, are detailed in Section 2. In Section 3, the experimental study is detailed and many experiment outcomes are given. Section 4 concludes with summarizing the study's findings.

## 2 Methodology

### 2.1 Structural Design and Fabricated Prototype

A lightweight RGB-D based method was developed for UAV dynamic obstacle detection. Ensemble detectors with tracking ensured accuracy and low computation. Indoor flight tests validated reliability was developed by earlier studies. Output variables required by contact-detection algorithm were primary focus of the design's development. The presence of contact is the primary variable that has to be predicted. But this data might not be sufficient to get back on your feet after a crash. It is equally important to understand the contact orientation and the feedback from collisions in real-time. Our method is crucial for developing a more dependable system for accident recovery, as this demonstrates. The platform consists of the inner and outer platforms, the elastic band mechanism, and the sensor integration unit. There are two halves to the inner platform. The light blue components represent the eight inner ring arms that make up the drone's square construction. The screw holes on these arms run perpendicular to the arm frame. The four dark blue elements that make up the inner ring supports. These are fastened to two inner ring arms using a screwed connection. Their purpose is to encase the propellers in protection, house the sensor locations, and provide a sliding groove for the outer ring to move radially. To construct the exterior platform, two main parts are needed. The sensor tip and sliding support are housed in the outer ring sensor component, which is the light green. When fitted into the sliding slot, it allows for radial displacements between the outer and inner platforms. An example of a snap-fit connection, which connects the outer ring non-sensor component to the outside perimeter. The dark green bits round off the outside edge. In the absence of any external force acting on the platform, the two rings remain centered with respect to one another due to an elastic band system that connects the two platforms. To put this system together two attachments were needed the orange bits to tension an elastic band for each sliding support. Because of forces that are not radial, the outer ring can still rotate notwithstanding this system. Nevertheless, testing has shown reliable findings due to the platform's simplicity and inexpensive cost, thus this constraint is not a deal breaker. It is the job of flex sensors to measure the relative displacements of the two platforms. Along their longitudinal axes, each sensor measures displacement. Hence, in order to measure displacement in two-dimensional space, a minimum of two sensors oriented perpendicularly are required. But for redundancy and balanced data collecting, four sensors are the way to go. Using silicone, we secured the sensor tips to their respective slots on both platforms and covered them with heat-shrinkable shields. The pre-bent position of the sensors implanted to detect directional displacements of the outer ring. In the event of an obstacle collision, the outer ring platform undergoes a radial relative movement to the inner platform along the impact's direction. The flex sensors sandwiched between the two platforms detects this movement. The data from the sensors are subsequently inputted into the algorithm for contact detection, identifies collision and supplies information about their direction and quantity of displacement. A model is created in accordance with the blueprint of the obstacle contact detecting system. Using PLA material, every single part that was detailed earlier has been 3D printed. A 3D point cloud algorithm used Sobel edges and clustering for obstacle detection.

LM-BP neural networks classified terrain obstacles. Tests showed robust performance in unstructured areas as reported by earlier research works. The platform for obstacle contacts detection that was created.

## 2.2 Data Acquisition and Dataset Creation

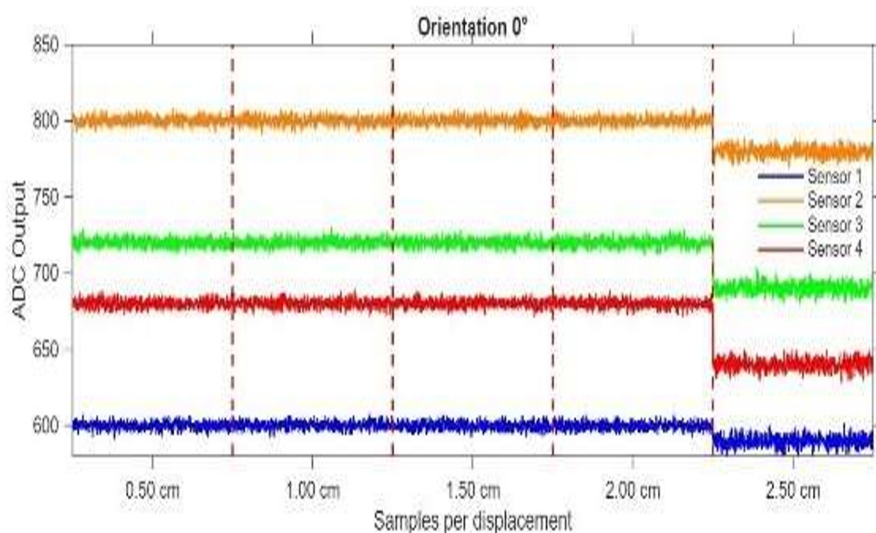
One of drone legs is equipped with an Arduino microcontroller, and serves as the on-board computer for the data collecting implementation. This platform is part of the Latte Panda. The flex sensors send analogue signals to this microprocessor, which processes them. The Latte Panda computer is linked to each flex sensor via voltage divider setup. The voltage divider and flex sensors were both housed in the drone's body, with the crash platform housing the latter. The flex sensor has a positive voltage supply terminal and a ground terminal connected by a resistor in series. The analog inputs of the Arduino receive their voltage divider signal from the spot where the flex sensor and resistor are connected. The Analog-to-Digital Converter (ADC) integrated into the Arduino reads these analog inputs in a sequential fashion and transforms them into digital values under software control. The second stage entails reading the values from Arduino through 100 Hz serial connection using Python script that was created on the Latte Panda. The script is in charge of doing additional processing and analysis on the flex sensor data. Two methods for recording data were created to retrieve contact information from flex sensors, taking into consideration the data-acquisition system. The recordings are stored in a CSV file in both instances, as illustrated in Table 1. The raw sensor readings are shown in the first four columns. The next step is to give each measurement one of three possible values: `displacement_gt`, or `collision_pred`, `angle_gt`.

**Table 1.** Contact information of flex sensors

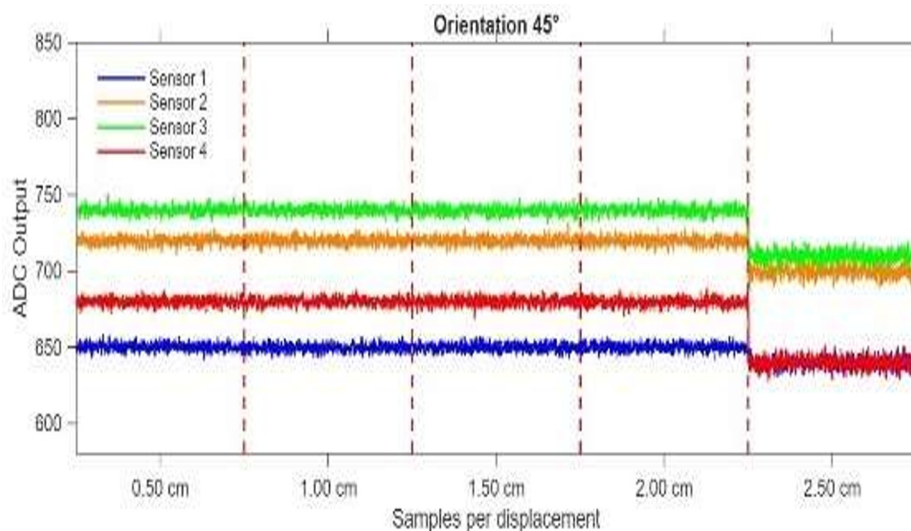
Sensor1	Sensor2	Sensor3	Sensor4	displacement_gt	angle_gt	collision_pred
662.1	736.21	839.91	719.71	1.18	88.43	1

This approach is used to calibrator tool at static setting to record contact information. The device may record shifts (0.50 cm, 1.0 cm, 1.50 cm, 2.0 cm, 2.50 cm) and contact orientations (0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°), are shown in Figure 1 to Fig. 7 correspondingly. By employing the calibrator tool approach, the dataset for calibration (C) is generated. This calibration dataset includes flight and static data that does not include collisions, so it might have both contact and no contact data. Before the flight dataset is used label contact events, this dataset is used train initial contact model. The second approach monitors incidents of human error in flight using an OptiTrack system. In this case, three Opti Track markers indicate the plane of impact with the wall, while further markers follow the center of the drone. As a function of angle among normal vector of wall and yaw vector of drone, we get the contact orientation. To ascertain the contact displacement, initial contact distance is recorded, and drone's positional alterations throughout the impact are monitored. The positional data is perpetually documented until the conclusion of the collision, as depicted in Fig. 8 to Fig. 11 accordingly. The collision labels match the predictions of model trained with calibration dataset, since the OptiTrack system did not have a mechanism to contact events. OptiTrack system documented eleven manual flights as illustrated in Fig. 8 to Fig. 11 correspondingly. Using the flight data, two datasets are generated. A dataset specifically created for training models with discrete contact samples is the Flight Discrete Dataset (FD). Table 1 shows that each sample represents a single time point. A training sample is a collection of measurements taken during a specific time frame. This dataset gives the model background on the sensors' actions during time. At any given time, step within the window, the label for each time-series sample will reflect that.

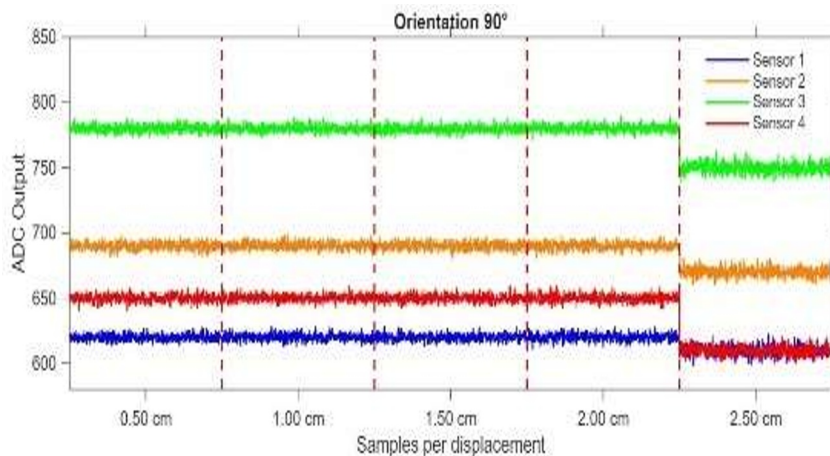
We utilized nine flights for both the FD and FW datasets to train the model, while we retained two flights for evaluation purposes. Since the training set contained mostly non-contact data, it was necessary to balance the dataset such that there were equal amounts of contact and non-contact samples.



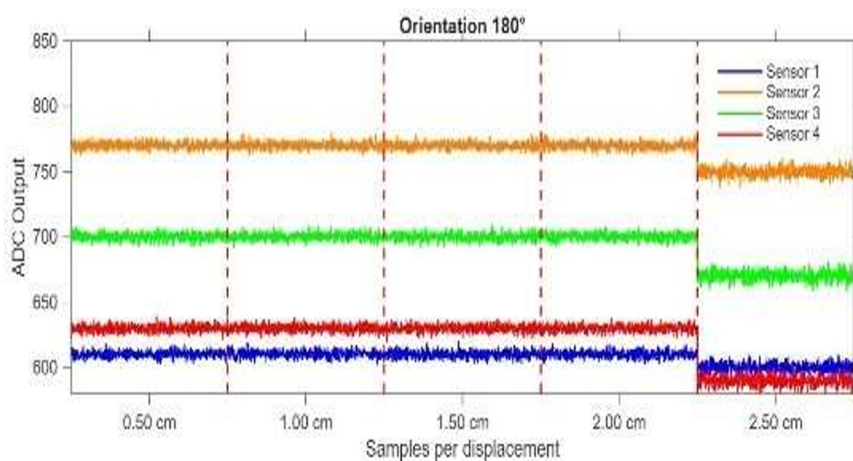
**Fig. 1.** Data from the calibrator tool illustrating contact events at  $0^\circ$  orientation



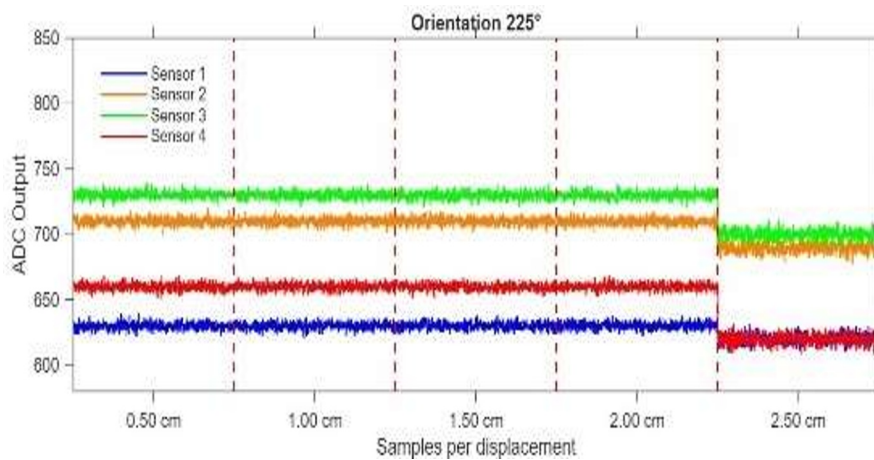
**Fig. 2.** Data from the calibrator tool illustrating contact events at  $45^\circ$  orientation



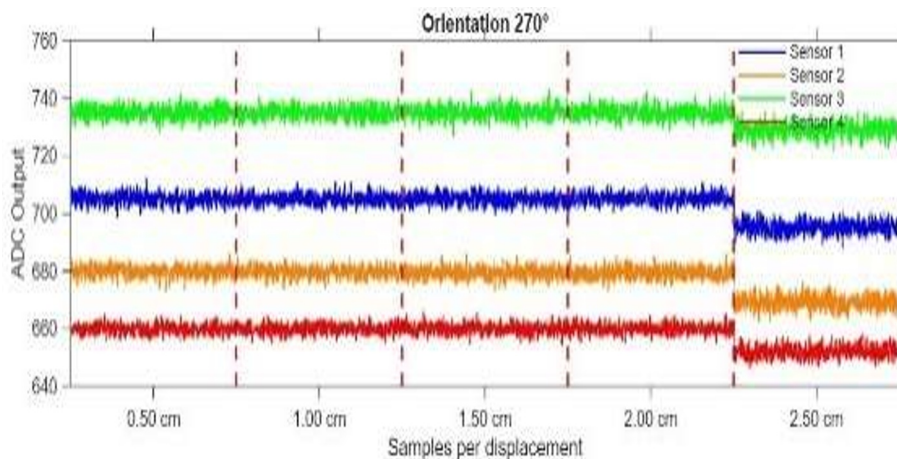
**Fig. 3.** Data from the calibrator tool illustrating contact events at 90° orientation



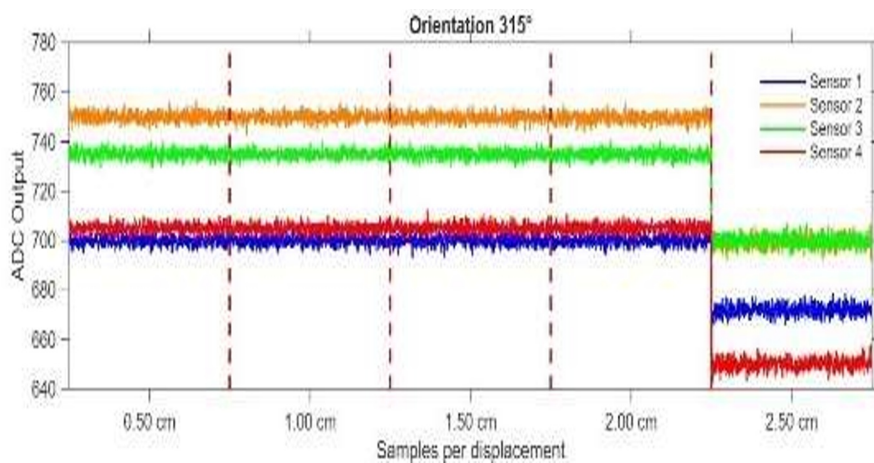
**Fig. 4.** Data from the calibrator tool illustrating contact events at 180° orientation



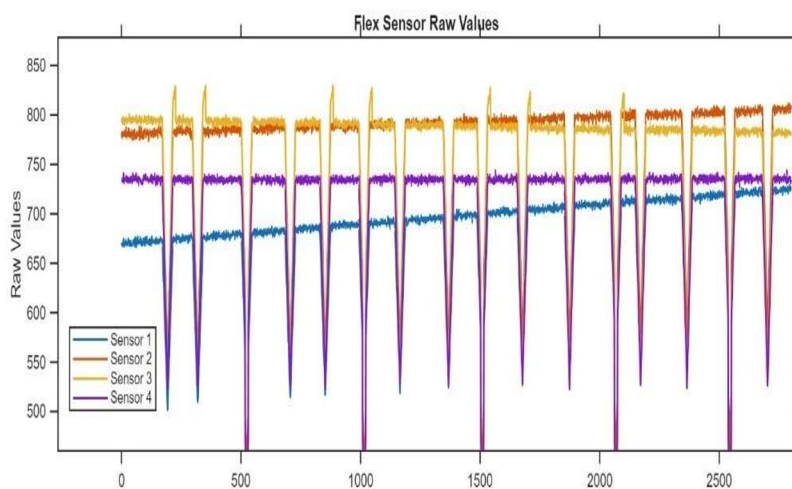
**Fig. 5.** Data from the calibrator tool illustrating contact events at 225° orientation



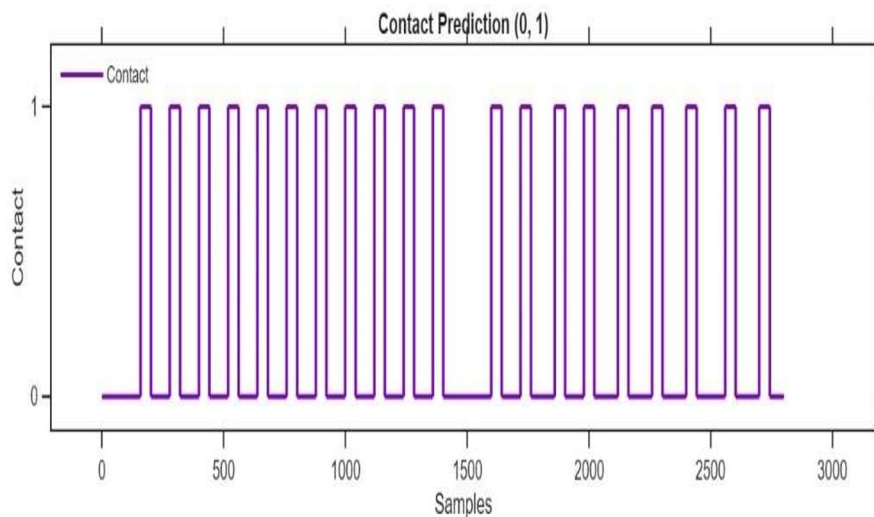
**Fig. 6.** Data from the calibrator tool illustrating contact events at 270° orientation



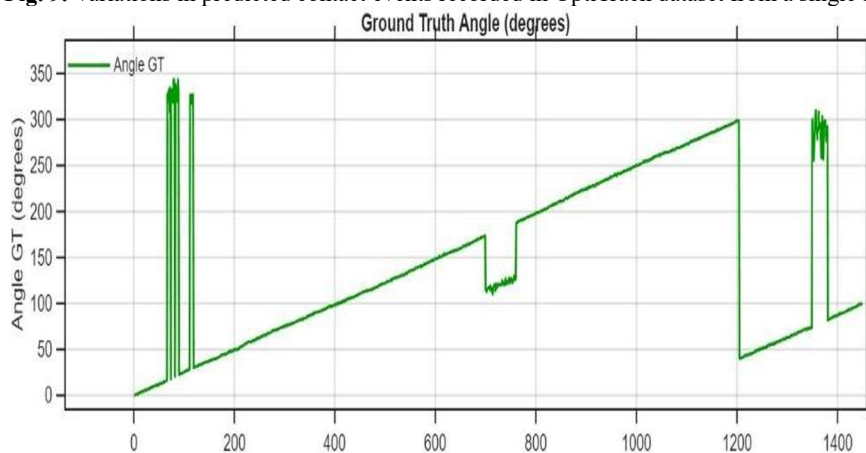
**Fig. 7.** Data from the calibrator tool illustrating contact events at 315° orientation



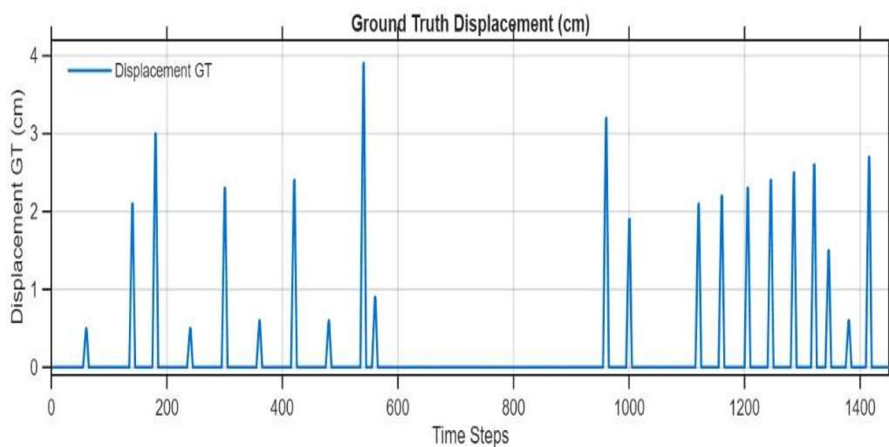
**Fig. 8.** Variations in flex sensor signals recorded in OptiTrack dataset from a single flight



**Fig. 9.** Variations in predicted contact events recorded in OptiTrack dataset from a single flight



**Fig. 10.** Variations in drone orientation recorded in OptiTrack dataset from a single flight



**Fig. 11.** Variations in platform displacement per collision recorded in OptiTrack dataset from a single flight

## 2.3 Model Construction

The nonlinear behavior of flex sensors is caused by the fact that their temperature resistance coefficient varies with operational temperatures. Fig. 1 to Fig. 7 shows that a direct logic may be used to forecast when contact will occur when the force is in line with particular sensor orientations (0°, 90°, 180°, 270°) and the measurement of the relevant sensor increases. Nevertheless, the platform design causes the flex sensors to undergo torsional movements when forces are applied at intermediate orientations (45°, 135°, 225°, 315°). This adds to the non-linearity of the sensor and leads to inconsistent measurement declines. Neural Network was selected as algorithm to forecast contact information from the flex in order to tackle this non-linearity [15]. A CNN model for agricultural robots integrated Ghost Modules and SE blocks. It distinguished real from false obstacles with high accuracy. Tests confirmed robustness in diverse conditions as reported by previous studies. Using this data, two separate tasks are executed: one is regression network that predicts displacement and orientation of contacts, and other is binary classifier that distinguishes among contact events. The NNs have received training from a variety of sources. One approach is to standardize sensor readings for varying temperatures by leveraging the differences between each sensor (D). Also utilized for model training are raw sensor readings (R). The learning rates and batch sizes used to train both models were 0.001 and 32, respectively.

### 2.3.1 Contact-Detection Model

In order to forecast when contacts would occur, Feed Forward Neural Network (FFNN) had set up. Similar ML-based approaches for collision avoidance in robots, such as MoDeT, have tracked obstacles using 2D-laser scans and clustering, with GNN improving velocity estimation and effective collision avoidance demonstrated on AMR SR300. To learn various complicated patterns, it incorporates ReLU functions to non-linearity, an input layer that is customized to all input type, and a hidden layer with 16 units. It uses Sigmoid activation to find probability that the predictions will be in one of two categories. To the interval, the input values are transformed by this function. Binary Cross Entropy Loss function, which is last stage at training the model, is detailed in Equation (1).

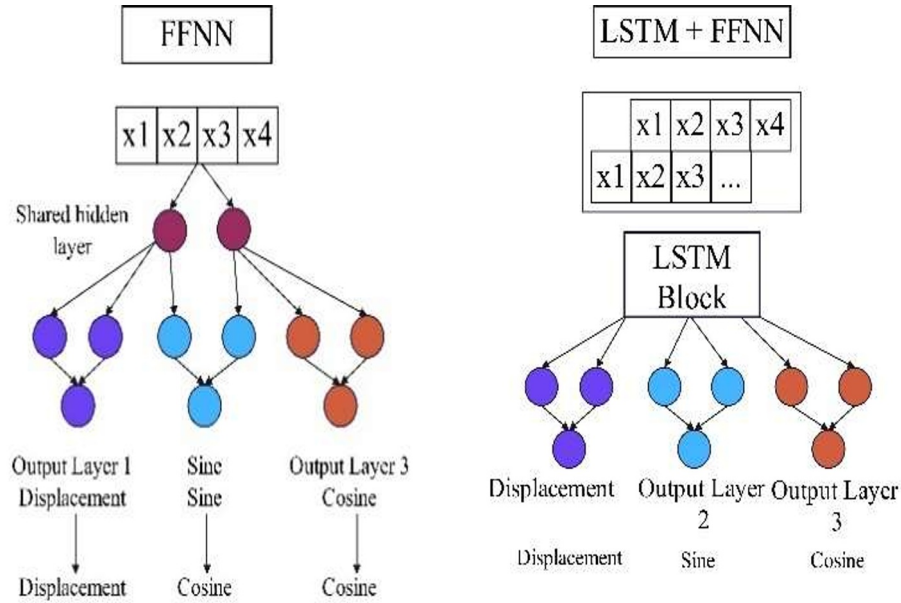
$$\text{Binary Cross-Entropy} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (1)$$

in which  $y_i$  represents the real label and  $\hat{y}_i$  stands for the anticipated likelihood.

### 2.3.2 Angle and Displacement Model

Fig. 12 shows two potential designs for this task: one using a full-feedback neural network (FFNN) and the other combining FFNN with LSTM blocks (L+F). Fig. 12(a) shows the first method, which uses a single 16-unit general hidden layer to extract features for angle and displacement prediction. As a result, three groups one hidden layer each with 32 units were trained to anticipate the sine, cosine, and displacement of a collision. After every hidden layer, there is a ReLU activation function. Similar approaches have been applied in vision-based navigation systems, where CNN memory combined with stereoscopic detection and control loops enabled real-time obstacle avoidance. Fig. 12(b) shows an alternative method, where NN input were matrix with all sensor time series rather than individual sensor readings, and the experimental parameter  $W$  represents the window size. An LSTM block with a single hidden layer and a hidden size of 32 replaces the first method's generic hidden layer. Following this, the LSTM block splits into three groups, each consisting of a single hidden layer with 32 units. These groups are designed to anticipate several types of collision angles: sine, cosine, and displacement.

After every hidden layer, there is a ReLU activation function. The two designs conclude with an activation-free linear output layer that, as illustrated at Fig. 12(a) Fig. (b), enables the network to generate continuous values that directly match the expected displacements and angles (sine and cosine).



**Fig. 12.** Angle and displacement prediction models. (a) FFNN-based approach (F); (b) LSTM combined with FFNN (L+F)

A mixed loss function is used to optimize the model during training. To determine the displacement loss, the Mean Squared Error (MSE) is used.

$$MSE_{\text{displacement}} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2)$$

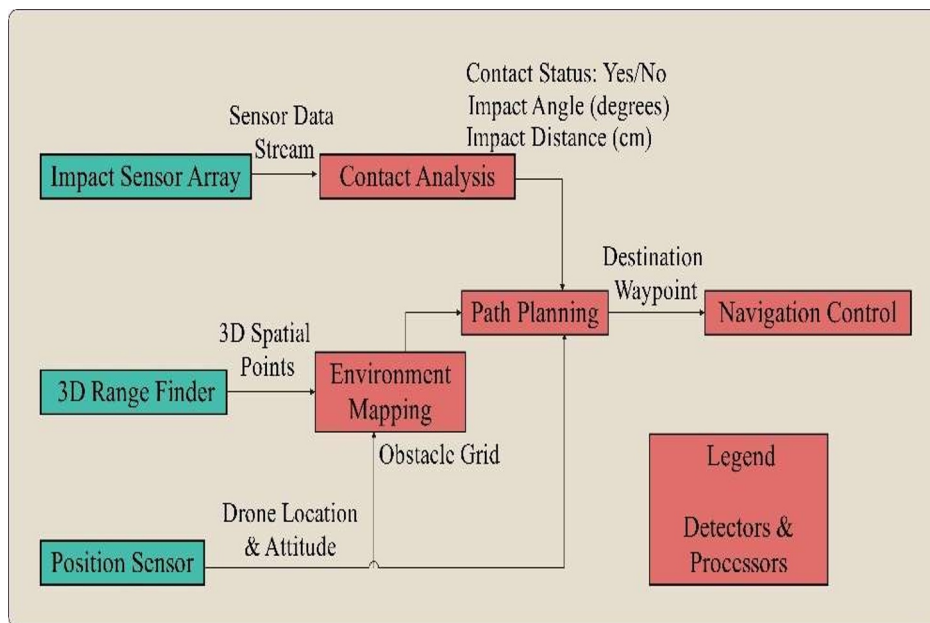
where the  $i^{\text{th}}$  anticipated value is denoted by, and the  $b^{\text{th}}$  actual value is represented. Equation (3) is employed to calculate the angle loss using a Sin Cos Distance loss.

$$\text{Sine Cosine Distance}_{\text{angle}} = \sqrt{(\sin(\hat{\theta}) - \sin(\theta))^2 + (\cos(\hat{\theta}) - \cos(\theta))^2} \quad (3)$$

$$\text{Angle Loss} = MSE_{\text{displacement}} + \text{Sin Cos Distance}_{\text{angle}} \quad (4)$$

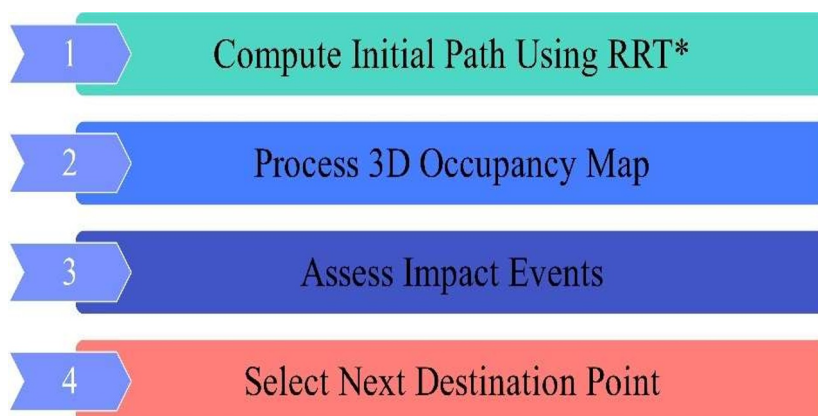
## 2.4 Navigation using Contact Sensing

Fig. 13 shows the completed autonomous stack, that was used to learn how the collision platform performed in a real autonomous mission. During an autonomous mission, the system will be fed the contact prediction information in order to facilitate collision recovery. The drone is outfitted with a LiDAR Livox Mid360 for mapping purposes and the created collision platform. An OptyTrack system will supply the location and orientation of the drone, which will form the basis of the localization. Using the point cloud obtained by LiDAR sensor, 3D grid map was created using the existing ROS2 module Octomap Server. The mapping subsystem were not main emphasis of project. With a collision platform diameter of 0.88 m, map resolution had set accordingly. Using 3D grid map of environment supplied by the mapping subsystem, the RRT\* path-planning method is employed.



**Fig. 13.** Proposed system pipeline for contact-driven navigation

The resolution of the contact map is the length in meters of the square box sides used to map the contact obstacles. The octomap resolution shows the square box side length used to map obstacles detected by LiDAR. The RRT search space range specifies the three-dimensional volume, that is used to investigate potential planning solutions. The distance in meters kept between RRT waypoints is represented by the RRT q parameter. In the RRT objective, the coordinates of the mission's target waypoint are defined. Finally, the step back parameter is a metric that represents the number of waypoints that are taken into account during collision recovery.



**Fig. 14.** Submodules of the path-planning system combining LiDAR-based 3D mapping and contact feedback

In order to establish the starting route from the origin to the destination coordinates, the path-planning node always employs the RRT algorithm at startup. For free-obstacle trajectory planning, the 3D grid map will be utilized if it is accessible.

There is a constant rate of execution for the following logic after initialization. The HANDLE OCTOMAP submodule checks for intersections with the previously computed path whenever the mapping node sends a new 3D grid map. Such collisions were detected, submodule will recalculate a path free of obstacles using the RRT method and update the current drone position with most recent obstacle information. In the event that the manage COLLISION DETECTION module receives contact-detection message, it will manage collision, steer the drone away from obstacles, and recalculate its course. In a no-contact condition, the collision reaction logic involves going back to earlier waypoints. The procedure for handling collisions is as follows:

**Collision Detection:** In the event that the drone detects an impending collision, it will promptly lower the destination waypoint in accordance with the Step Back parameter, therefore returning to its prior waypoints. The drone has entered a state of recovery after a collision.

**Collision Recovery:** There are two potential outcomes in the collision-recovery state:

**Persistent Contact:** As drone continues to sense touch while retracing its steps, it will send itself further and further back to earlier waypoints it enters no-contact state, causing the target waypoint to shrink.

**No-contact State:** When the drone reaches their waypoint with no contact is detected, the collision state was ended.

**Post-Collision Handling:** If the contact event is finished, they involve detecting collisions, reaching and recovering a no-contact state, and the obtained predicted angle is used to geotag the detected contact. The steps to locate the obstacle in the globe frame are:

$$\mathbf{o}^{uav} = \begin{bmatrix} \cos(\theta_c) \cdot r_d \\ \sin(\theta_c) \cdot r_d \\ 0 \end{bmatrix} \quad (5)$$

$$\mathbf{R}_{uav}^w = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$\mathbf{o}^w = \mathbf{R}_{uav}^w \cdot \mathbf{o}^{uav} + \mathbf{d}^w \quad (7)$$

The collision angle is denoted as  $\theta_c$  (in degrees), and the drone's radius as  $r_d$ . Obstacle position at drone frame were represented as  $\mathbf{o}_{uav} \in \mathbb{R}^3$ , while the drone's position in the world frame is  $\mathbf{d}_w \in \mathbb{R}^3$ . Based on angle of the drone, the rotation matrix is applied to both drone's frame and world frame.  $\psi$  (in radians) is given as  $\mathbf{R}_{uav}^w \in \mathbb{R}^{3 \times 3}$ . Finally, obstacle position at world frame were expressed as  $\mathbf{o}_w \in \mathbb{R}^3$ . Finally, in the event of a collision, DECIDE TARGET WAYPOINT module notifies offboard control node of the target waypoint. The standard operation involves increasing the target waypoint upon reaching it, assuming there are no collisions. The HANDLE COLLISION DETECTION submodule explains the reasoning behind selecting the target waypoint in the event of a collision.

### 3 Experimental Study

The accuracy of obstacle contact-detection system was tested in an experimental setting. Dataset, model construction, and result evaluation were all part of this procedure.

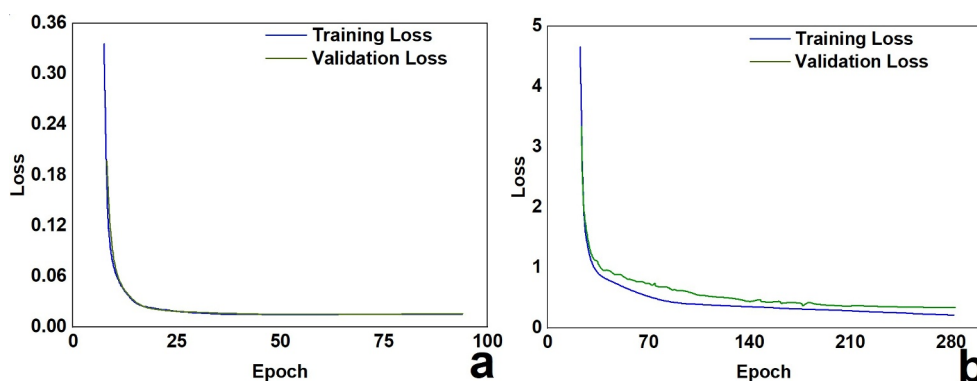
#### 3.1 Structure of the Proposed Contact-Detection Model

Table 2 displays the findings of all the contact-detection models that were tested using the flight datasets.

**Table 2.** Performance assessment of contact approaches

Model	1	2	3	4
Input	R	R	D	D
Dataset	FD	C	FD	C
Accuracy	0.99	0.99	0.95	0.69
Precision	0.82	0.98	0.47	0.11
AUC	0.96	0.85	0.94	0.8
F1 Score	0.87	0.82	0.62	0.19
Recall	0.93	0.71	0.92	0.93

Importantly, mentioned earlier, the flight dataset's contact events were labelled using Model 1, which was trained with calibration data. Model 1's excellent accuracy and precision values can be attributed to this. On the other hand, the recall was low, suggesting that the model failed to identify many contact-labeled events and produced an excessive number of incorrect negative predictions. When trained with fluctuations in sensor data, most models (Models 2 and 4) perform poorly. It appears that the model is missing out on important information necessary to understand the collisions due to this codification. Model 3 performed better than all metrics when trained using the flight dataset and raw sensor values as inputs. This proves that it can accurately anticipate and prevent contact situations. For 100 iterations, Fig. 15(a) displays the training loss of Model 3. With an average processing time of just 0.33 ms far quicker rate of 12 ms the Model 3 process quickly. These findings demonstrate how important it is to train models using data that represents real-world situations. The model's ability to generalize is crucial to its performance, and this approach allows it to do so effectively. Related work in neuromorphic computing has shown that spiking neuron circuits can perform obstacle detection with high energy efficiency and robustness under noise conditions.



**Fig. 15.** (a) Loss trend during training and validation of Model 3 for contact detection. (b) Training evolution of angle–displacement model

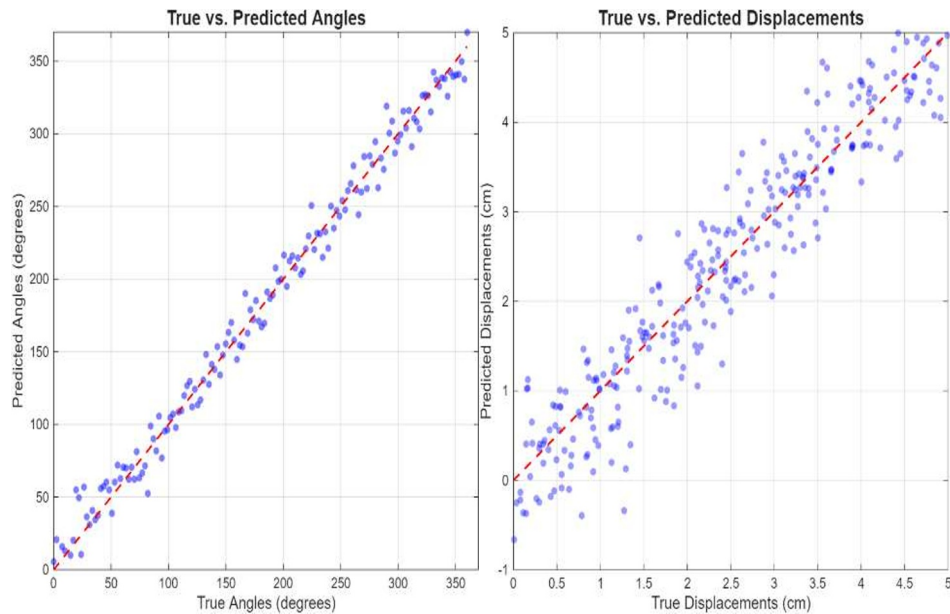
Table 3 displays the results of the displacement and angle models that were tested. It is clear from testing with flying datasets that Models 1 and 2, which were trained solely using calibration data, struggle to generalize. Because of this, training with data that is as realistic as possible is of the utmost importance. The pattern is the same in both contact-detection model and displacement and angle model. Using sensor differences inputs leads to inferior performance compared to models trained using raw values or discrete raw values inside time series.

There is noticeable pattern in the results for angle predictions when comparing simple FFNN with more complex models that include LSTM block. Interestingly, models that use time-series data (Models 5, 6, and 7) significantly enhance the accuracy of angle prediction.

**Table 3.** Model performance analysis for angle and displacement estimation

Model	1	2	3	4	5	6	7
Dataset	FD	FW	FD	C	C	FW	FW
Input	R	W	D	D	R	W	W
Architecture	F	L+F	F	F	F	L+F	L+F
W size	-	20	-	-	-	10	30
Angle MAE	16.61	9.64	19.37	43.81	28.91	12.69	9.31
Angle MSE	804.91	190.08	1049.96	3924.93	2042.13	356.01	204.23
Angle RMSE	28.37	13.79	32.40	62.65	45.19	18.87	14.29
Angle Std.Dev	23.00	9.86	25.97	44.78	34.73	13.96	10.84
Angle Median	6.52	6.69	8.80	22.02	15.49	7.34	5.41
Angle Max	132.43	55.21	177.34	178.90	177.42	64.22	59.32
Displacement MAE	0.58	0.64	0.75	1.16	1.04	0.64	0.66
Displacement MSE	0.63	0.72	0.92	2.78	1.74	0.71	0.80
Displacement RMSE	0.80	0.85	0.96	1.67	1.32	0.84	0.89
Displacement Std.Dev	0.79	0.83	0.96	1.67	0.85	0.82	0.88
Median	0.46	0.50	0.63	0.80	0.94	0.51	0.51
Max	4.05	3.13	4.39	8.08	4.14	3.68	3.97

Using these models, we can reduce the projected angle errors (such MAE, MSE, and RMSE) by almost half. Windows sizes of 20 and 30 appear to outperform windows sizes of 10, which did not collect enough data, among the time-series trained models. Nevertheless, it has been shown that the trends in model design for angle and displacement forecasts are distinct. Model 3, the most basic model trained using flight data has the best displacement predictions (Table 3). This goes against the grain of what has been happening in angle prediction, where more complicated models have been showing better outcomes. These surprising results call into question our presumptions about building a convolutional neural network to forecast angles and displacements using common information. In particular, Model 6's accuracy in forecasting displacements and angles stands out. Fig. 15(b) displays the training loss for Model 6 across 250 epochs. Because Model 6's mean processing time is 2.62 ms, it can analyze sensor data without delay in real-time, even when the measurement update rate is 12 ms. Model 6's predicted and actual flight test dataset angles and displacements are compared in Fig. 16(a) and Fig. 16(b). The model is able to nearly match the anticipated vs actual collision angles into straight line around whole UAV perimeter, as shown in Fig. 16(a), with only a handful of noticeable outliers. All major mistakes have been eliminated. Because angles are inherently circular, it is not an issue to conduct forecasts of  $360^\circ$  for angle  $0^\circ$  and  $0^\circ$  for angle  $360^\circ$ . Nevertheless, as shown in Fig. 16(b), the accuracy of displacement prediction is lower than that of angle prediction. Predicted vs actual displacements that match the red dotted line (highest model performance) show a pattern, although the forecasts are less concentrated than the angle predictions. Fig. 16(a) and Fig. 16(b) both employ the contact event's time-series properties to predict the contact events beginning with the first touch and ending with the release of contact. One can observe the development of angle prediction throughout contact event in Fig. 16(a). Because sensor values change suddenly from one specimen to the next, predicted angle fluctuates around true angle (Fig. 16(a)), but it still stays within  $5^\circ$  of the true labels.



**Fig. 16.** Model 6 performance on test flight dataset. (a) True vs. predicted angles; (b) True vs. predicted displacements

Figure 8b illustrates that there is a  $50^\circ$  angle prediction error at the start of the contact event, which decreases to  $30^\circ$  at the end. The ability to anticipate continuous displacements is demonstrated in Fig. 16(a), where predicted displacement mimics displacement pattern from actual label. Nevertheless, the accurate pattern of anticipated displacement, as shown in Fig. 16(b) is rising and then falling, but there is a noticeable offset to the actual displacement.

### 3.2 Autonomous Contact-Based Navigation

For this test to be valid, the navigation stack must get to a specific waypoint. On purpose, a 2-by-3-meter obstacle in the shape of a goal is positioned along this route. The LiDAR sensor is unable to identify the thin tensioned vertical wires that obstruct the left side of this objective. The drone will be set on a green trajectory to avoid the wires, while the LiDAR creates a map of the surroundings using white boxes. The collision platform's ability to compensate for the LiDAR's low resolution is demonstrated here. The mission visualization, which includes a gray circle centered on the arrow to indicate the collision platform diameter and a red arrow to show the position of the drone. The outcome of the mission visualization. We can see the starting path that passes through the unseen wires by the LiDAR. Trajectory tracking was improved using NMPC with a QP-based inner loop. Inspired by time-delay control, it enhances robustness under uncertainties. Hexarotor tests showed superior stability over classical controllers as reported by previous studies. The detected collision occurrence on the map, marked by a green box. Reversing to a non-contact condition upon collision detection, which is the collision recovery maneuver. Lastly, the drone's recalculated course that avoids obstacles detected by LiDAR and contact-based methods, as well as its movement toward the goal while avoiding wires. Incorporating LiDAR and contact information into the path-planning algorithm improves obstacle recognition and avoidance, as demonstrated in this experiment. In order to improve the 3D map that the LiDAR sensor produces of the surrounding area, the contact-detection system adds features that the LiDAR did not pick up on.

A more thorough environmental model is the outcome of this combination. A single collision successfully concluded the contact-and LiDAR based navigation missions. Total of 1.042 seconds passed during the contact occurrence. The bendy wires probably account for the longer length since they make the drone's push against them a less sharp and more gradual collision. This demonstrates how well the platform can identify flexible collisions. Table 4 shows the observed navigation performance using Lidar and physical contact sensing.

**Table 4.** Observed navigation performance using Lidar and physical contact sensing

Metrics	Results
Angle predicted	3.63
Angle error	3.48
Angle ground truth	0.21
Number of collisions	1
Reaction time (s)	0.529
Contact event duration (s)	1.042

Adding LiDAR data to the planning process can cause more computing delays, which could explain the sluggish response. For precise collision mapping, the forecast angle error of  $3.48^\circ$  was crucial. Dynamic obstacle avoidance extended the Dynamic Window Approach with EKF tracking. Laser data enabled the trajectory prediction. Simulations showed reliable high-speed navigation as demonstrated by earlier research works. Because of the little margin of error, the drone was able to successfully reroute its flight path to avoid the barrier with a single collision.

## 4 Conclusions

An innovative obstacle contact-detection platform was introduced in this study; it may be fastened to an existing drone frame to collect data on collisions. Our results are as follows:

- A prototype contact-detection platform has been built. Both inside and outside platforms have flex sensors built in. Collision detection can be achieved when the outer platform collides by analyzing the displacements of the flex sensors between the platforms.
- We build a contact-detection algorithm on top of neural networks that can identify, locate, and move in relation to contacts. It is the onboard computer that executes this algorithm.
- An obstacle contact-detection platform experimental system is constructed and tested in order to validate the contact-detection technique. There was a 0.99 accuracy rate in the contact detection results. Our mean absolute error (MAE) for predicting angles was  $9.31^\circ$ , with a standard deviation of  $10.84^\circ$ . As for displacement prediction, we were able to get with a standard deviation of 0.79 cm and MAE of 0.58 cm. And with the expected contact information, autonomous contact navigation achieved, making up for limited resolution of the LiDAR.

By offering accurate data on the perimeter's contacts and real-time feedback on collisions, the obstacle contact-detection platform improves the UAV's collision recovery and mission completion capabilities. There is promise for a future when autonomous missions are less likely to fail because to this research's innovative method to touch detection, which could influence and inspire advancements in collision-tolerant robotics.

## References

1. H. Cheng, F. Zhang, RGBlimp-Q: robotic gliding blimp with moving mass control based on a bird-inspired continuum arm. *IEEE Transactions on Robotics*. **41**, 5097 (2025). <https://doi.org/10.xxxx/robot.2025.rgbqlimpq>
2. I. D. del Pino, A. Santamaria-Navarro, A. G. Garrell, F. Torres-Medina, J. Andrade-Cetto, Probabilistic graph-based real-time ground segmentation for urban robotics. *IEEE Transactions on Intelligent Vehicles*. **9**, 4989 (2024). <https://doi.org/10.xxxx/urbanrobotics.2024.0014>
3. K. Adithya, R. Girimurugan, Benefits of IoT in automated systems. *Integration of Mechanical and Manufacturing Engineering with IoT: A Digital Transformation*. 235 (2023).
4. X. Cai, S. Zhong, T. M. Tan, W. J. Ang, S. Foong, Design and optimization of a samara-inspired lightweight monocopter for extended endurance. *Biomimetics*. **10**, 7214 (2025). <https://doi.org/10.xxxx/biomi.2025.7214>
5. B. Habas, B. Cheng, From flies to robots: inverted landing in small quadcopters with dynamic perching. *Robotics and Autonomous Systems*. **41**, 1773 (2025). <https://doi.org/10.xxxx/robot.2025.1773>
6. M. Vairavel, R. Girimurugan, C. Shilaja, G.B. Loganathan, J. Kumaresan, Modeling, validation and simulation of electric vehicles using MATLAB. In *AIP Conference Proceedings*. **2452**, 030006 (2022). <https://doi.org/10.1063/5.0114084>
7. K. Viswanath, B. H. Vardhan, P. Kiran, Integrating neural networks for predictive torque control and obstacle avoidance in autonomous robot. *Journal of Intelligent Systems*. **7**, 1 (2025). <https://doi.org/10.xxxx/jis.2025.001>
8. P. Rea, E. Ottaviano, K. Antosz, E. Kozłowski, H. Lopes, J. Machado, Simulation of robotic inspections based on systematic data acquisition and analysis. *Applied Sciences*. **24**, 29 (2025). <https://doi.org/10.xxxx/appsc.2025.29>
9. N. AbuJabal, M. Baziyad, R. Fareh, B. Brahim, T. Rabie, M. Bettayeb, A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots. *Sensors*. **24**, 8089 (2024). <https://doi.org/10.xxxx/sensors.2024.24>
10. D. Gorgoteanu, C. Molder, V.-G. Popescu, L. Ú. Grigore, I. Oncioiu, Optimizing an autonomous robot's path to increase movement speed. *Machines*. **13**, 1892 (2024). <https://doi.org/10.xxxx/machines.2024.13>
11. L. Emmi, R. Fernández, P. Gonzalez-de-Santos, An efficient guiding manager for ground mobile robots in agriculture. *Agriculture*. **13**, 6 (2024). <https://doi.org/10.xxxx/agri.2024.13>
12. M. Vairavel, R. Girimurugan, C. Shilaja, G.B. Loganathan, Z. Polat, Analysis of hybrid electrical vehicles: Types, formulation and needs. In *AIP Conference Proceedings*. **2452**, 030005 (2022). <https://doi.org/10.1063/5.0114081>
13. D. Brugali, Modeling variability in self-adapting robotic systems. *Journal of Systems and Software*. **167**, 104470 (2023). <https://doi.org/10.xxxx/jss.2023.167>
14. C. Ntakolia, S. Moustakidis, A. Siouras, Autonomous path planning with obstacle avoidance for smart assistive systems. *Expert Systems with Applications*. **213**, 119049 (2023). <https://doi.org/10.xxxx/eswa.2023.213>
15. R. Vinoba, M. Vijayaraj, Novel control topology with obstacle detection using RDPSO–GBA in mobile ad-hoc network. *Journal of Computational Design and Engineering*. **160**, 847 (2020). <https://doi.org/10.xxxx/jcde.2020.847>