Deep Learning based Obstacle Avoidance for Unmanned Aerial Vehicles in Disaster Scenarios

Minh Hieu Tran, Thomas Lundqvist, Jakub Espandr, and Min-Fan Ricky Lee, Member, IEEE

***Abstract*—Obstacle avoidance remains a major challenge for unmanned aerial vehicles operating in visually complex and dynamic environments. Traditional rule-based systems often fail in disaster scenarios due to occlusion and unpredictable object motion. This study proposes a supervised deep learning model that predicts navigational actions directly from sequences of depth images. The model combines a convolutional neural network with a convolutional long short-term memory network to extract both spatial and temporal features. Data was collected in a simulated post-disaster environment using CoppeliaSim and further tested on real-world depth images. The model achieved over 98% training accuracy, with validation accuracy between 60% and 70%, revealing generalization limitations due to label inconsistencies. Despite this, inference speed averaged below 10 milliseconds per frame, demonstrating its suitability for real-time UAV deployment. The results highlight the feasibility of integrating spatial-temporal learning with RGB-D sensing for autonomous navigation.**

***Index Terms*—Convolutional neural networks, Deep learning, Long short term memory, Obstacle avoidance, Unmanned aerial vehicles**

I. Introduction

UNMANNED aerial vehicles (UAVs) are widely used in disaster tasks such as aerial monitoring, rescue missions, and search operations. Because of their usefulness, more organizations, researchers, and individuals are using drones. This results in a growing number of UAVs deployed across various sectors [1].

Despite the growing use of UAVs in post-disaster scenarios, locating victims in obstructed and dynamically changing environments remains a significant challenge. In disaster zones with collapsed structures or fallen trees, visual occlusion and dynamic environments often disrupt traditional sensing and navigation systems due to inaccessible disaster sites. This limitation is critical in time-sensitive rescue operations, where delays in victim localization can reduce survival chances and hinder the coordination of ground rescue teams [2].

Previous research has addressed sensor technologies such as FMCW and UWB radar sensors to penetrate rubble and locate survivors in complex environments [2]. Additionally, a navigation and obstacle avoidance system was implemented with ultrasonic sensors for low-cost drone platforms in unknown settings [1]. Furthermore, a comprehensive review highlighted radar-, LiDAR-, and camera-based methods for collision avoidance, but noted limitations in visual sensing under occlusion [3]. However, real-time detection in visually obstructed dynamic environments remains an unsolved challenge [1] [2]. Traditional search methods relying on human teams or manually piloted drones are often slow, labor-intensive, and restricted by physical access to disaster zones [2].

While previous research [1] [2] [3] demonstrates effective sensing in controlled or static environments, it typically relies on rule-based systems or instantaneous sensor readings, lacking temporal modeling and adaptability under uncertainty. This study builds on these findings by introducing a learning-based alternative that can reason over time and operate in occluded and dynamic environments that prior methods do not explicitly address.

Building on this motivation, this study proposes a supervised deep learning model to be integrated into a UAV, capable of predicting real-time navigational actions directly from labeled depth image sequences. By utilizing a recurrent convolutional neural network, the system learns to map spatiotemporal patterns in the input data to discrete movement actions, enabling it to recognize obstacle configurations and react accordingly without relying on explicit mapping or handcrafted planning rules.

II. Methods

This study introduces a system built around a Convolutional Neural Network (CNN) and a Convolutional Long Short-Term Memory (ConvLSTM) network to process image sequences of depth data. The network is designed to predict one of nine discrete navigational actions by learning both spatial features from visual input and temporal patterns related to obstacle movement. The training data was collected in CoppeliaSim, a physics-based simulation tool for robotics research that enables dynamic environment modeling under repeatable conditions. In addition to simulated data, testing was performed on real-world depth images captured as sequential frames to emulate a UAV's point of view.

ConvLSTM was selected because of its ability to preserve spatial structures while learning temporal dynamics. This property has proven effective not only in UAV navigation but also in other sequence-based tasks such as crash prediction, where spatial-temporal patterns are critical for anticipating future states. [4]. Shi et al. [5] explain that conventional LSTM networks, which use fully connected transitions, inherently lose spatial information by flattening the input, and that although CNNs excel at capturing spatial features, they do not

Minh Hieu Tran is with the Halmstad University, Halmstad, Sweden (email: hietra19@student.hh.se).

Thomas Lundqvist is with the Halmstad University, Halmstad, Sweden (email: thomlu21@student.hh.se).

Jakub Espandr is with the University of Pardubice, Pardubice, Czech Republic (email: st62707@student.upce.cz).

Min-Fan Ricky Lee is with the National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: rickylee@mail.ntust.edu.tw).

naturally model temporal dependencies. These observations support the choice of ConvLSTM and CNN in this project because it combines the advantages of both CNNs and LSTMs, preserving spatial details and capturing temporal changes. Additionally, the primary challenge addressed by this model lies in enabling UAVs to make fast, context-aware navigation decisions in dynamic environments, where traditional rule-based systems often fail due to their inability to process temporal dependencies and adapt to unpredictable obstacle movements without explicit mapping or predefined rules.

Since collecting real-world training data for UAV navigation in disaster environments is constrained by legal, safety, and logistical challenges, this study simulates a disaster scenario in CoppeliaSim to enable the development and evaluation of learning-based obstacle avoidance. An example of the disaster scenario is shown in Fig. 1.

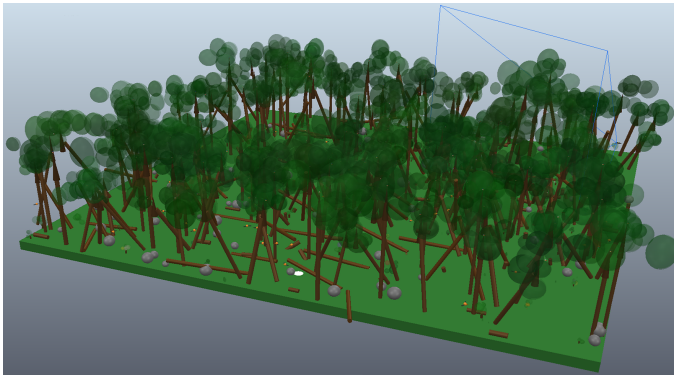

Fig. 1. Simulated post-disaster environment rendered in CoppeliaSim, showing the UAV equipped with an RGB-D camera navigating through dynamic and static obstacles.

The simulation scenario in Fig. 1 consists of a total of 540 objects, utilized in each run. The environment is generated and reinitialized procedurally before every simulation episode, introducing randomized placements of both static and dynamic elements. Among these, birds act as dynamic obstacles with unpredictable movement trajectories, while the remaining objects consist of static obstacles such as trees, rocks, bushes, and foliage. This randomized setup ensures environmental diversity and supports robust evaluation of the navigation and obstacle avoidance algorithms across multiple trials.

A. Data collection

The data collection process was divided into two parts, with one part dedicated to collecting training and validation data and the other focused on gathering test data. The total dataset consists of 5850 images, from which 4540 were used for training, while the remaining 1310 were used for validation. The data collection process for training and validation was implemented using the CoppeliaSim simulation environment to support obstacle avoidance in disaster scenarios.

The simulation features a quadcopter equipped with an RGB-D camera shown in Fig. 2, operating in a dynamically generated environment that emulates outdoor disaster conditions. The radar module illustrated in Fig. 2 measures the Euclidean distance between the drone and the designated target, as formalized in (1). Depth images, victim vector, pose, and actions were collected, ensuring consistency among frames. Each dataset entry consists of a single-frame depth image paired with corresponding action metadata. The action is represented as a discrete label from a predefined set, including the directional movements of the drone. Additionally, a vector between the drone's current position and a predefined goal location is included in the dataset collection.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{1}$$

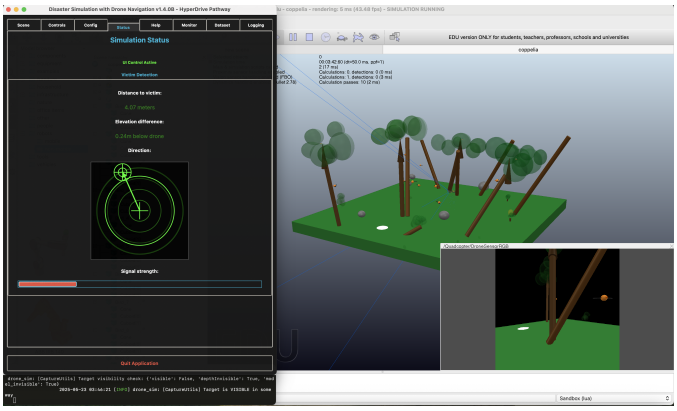

Fig. 2. Configuration settings of the procedurally generated disaster scenario in CoppeliaSim.

The process of collecting real data was conducted for testing purposes. This dataset consisted of 155 depth images, gathered using an iPhone 13 Pro by capturing frames sequentially to ensure comprehensive coverage of all predefined drone actions. Furthermore, data was collected using dynamic obstacles to simulate real-world interactions and enhance the robustness of the dataset.

B. Proposed architecture

The model processes depth images, which are sequences of single-channel frames capturing spatial depth information of the environment. In parallel, it also receives a victim vector that encodes contextual information such as the Euclidean distance between the UAV and the victim. This combination of visual and contextual input enables the model to make informed predictions based on both environmental perception and mission objectives.

Each depth frame in the sequence is first processed by a CNN backbone to extract spatial features from the input images. The extracted feature maps are then passed sequentially into a ConvLSTM module that learns temporal dependencies across the frame sequence and captures motion patterns in the environment. The output from the last timestep of ConvLSTM is subsequently passed through an AdaptiveAvgPool2d layer, which compresses the spatial dimensions into a fixed-size feature vector representing the spatio-temporal dynamics of the entire sequence. The model updates each cell based on its local neighborhood by employing convolution operations

with appropriate padding, effectively capturing spatiotemporal features. This design makes it particularly suitable for real-time obstacle avoidance tasks.

Meanwhile, the most recent victim vector is passed through a Linear Embedding layer, generating a compact contextual representation. This embedding is concatenated with the pooled features in the concatenation block, resulting in a fused vector that contains both spatial-temporal and contextual cues. The fused vector is then passed through a Fully Connected Layer, and the resulting logits are transformed by a Softmax function into a probability distribution over discrete actions. The action with the highest probability is selected as the final predicted output for navigation, as illustrated in Fig. 3.

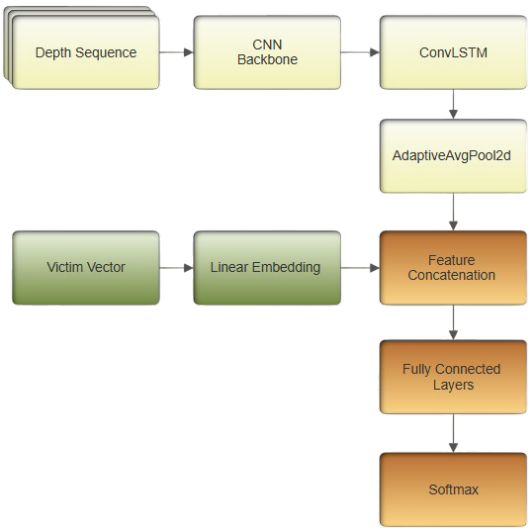

Fig. 3. Overview of the model action architecture.

C. Model training and validation

The training procedure of the proposed model shown in Algorithm 1 was implemented in the Google Colab environment. Each sequence of depth images was processed frame by frame using the widely used MobileNetV3 [6] to extract spatial features. These features were then passed to a ConvLSTM layer, which learned temporal patterns across the sequence.

Algorithm 1: Training Loop for DroneActionNet

Input: Model θ^* , training data loader, number of epochs N, learning rate η
Output: Trained model parameters θ^*
Initialize optimizer with learning rate η;
Define Cross-Entropy Loss as the training criterion;
for *epoch* $= 1$ **to** N **do**
  foreach *batch* (D, V, y) *in training loader* **do**
    Forward pass: compute predicted logits $\hat{y}$ from (D, V) using model;
    Compute loss $L \leftarrow \text{CrossEntropyLoss}(\hat{y}, y)$;
    Backward pass: compute gradients $\nabla_\theta L$;
    Update model parameters: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta L$;
return θ^*

The final hidden state was pooled to create a compact representation. This was combined with an embedded metadata vector and forwarded to a classification head to produce prediction scores. The logits were transformed into probabilities using the softmax function. The output of the network is a vector of logits, one for each of the discrete action classes. These logits are first transformed into probabilities using the softmax function, as shown in (2)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{2}$$

where $\mathbf{z} = (z_1, z_2, \ldots, z_K)$ is the input vector and K is the number of classes. The predicted probability distribution $\sigma(\mathbf{z})$ is then compared against the one-hot encoded ground-truth vector using the cross-entropy loss function, expressed in (3).

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} \log(y_{ij}) \tag{3}$$

where n is the number of samples, k is the number of classes, t_{ij} is an indicator function equal to 1 if the i-th sample belongs to the j-th class and 0 otherwise, and y_{ij} is the predicted output for the i-th sample and j-th class.

During training, the model parameters are updated using stochastic gradient descent with the Adam optimizer, which adaptively adjusts learning rates for each parameter. The computed loss gradient is backpropagated through all layers.

D. Testing and training evaluation procedures

The testing of the trained model was performed following the procedure outlined in Algorithm 2. In this stage, the test dataset consisted of sequences of depth frames and associated victim vectors. To generate frame-wise predictions, a sliding window of length T was applied across all available test frames.

Algorithm 2: Testing procedure

Input: Trained model,
Test dataset folder with depth and victim vector,
Sequence length T
Output: Predicted actions for each frame
Load test dataset using sliding window of size T;
foreach *frame index* i *from* $T-1$ *to* $N-1$ **do**
  Extract T-frame sequence ending at frame i;
  Construct model input tensors (X, V) from selected frames;
  Run model: $\hat{y} \leftarrow \texttt{model}(X, V)$;
  Predict action label $\hat{a}^* \leftarrow \arg\max(\hat{y})$;
  Store $\hat{a}^*$ at position i in `predictions`;
Display a set of test frames annotated with predicted action labels;
return *List of predicted actions and visualization of action distribution*

For each valid window, the model received a sequence of depth images and victim vectors as input. A forward pass

through the model produced a set of class logits, from which the most predicted action was determined using softmax. All predicted actions were stored sequentially. To enable qualitative evaluation, a subset of frames was visualized together with their corresponding predicted action labels. The entire testing procedure was executed on Google Colab using an NVIDIA A100 GPU, enabling fast inference and consistent evaluation across the full dataset.

The evaluation of testing predictions shown in Algorithm 3 was conducted through visual inspection rather than metric-based analysis, due to the absence of labeled test data. Selected sequences were displayed with their corresponding predicted action labels overlaid, enabling a manual assessment of temporal consistency and prediction quality across motion sequences. This evaluation focused on identifying misclassifications observable from model behavior in realistic deployment scenarios.

Algorithm 3: Performance validation

Input: Model,
DataLoaders with labeled training and validation batches:
X: sequences of depth frames,
V: victim metadata per frame,
y: corresponding action labels
Checkpoint logs: training accuracy, validation accuracy, validation F1
Output: Accuracy and macro F1-score for labeled data; plots of performance history
Set model to evaluation mode;
Initialize empty lists $\mathcal{P}$ (predictions) and $\mathcal{G}$ (ground-truth);
foreach *batch* (X, V, y) *from labeled DataLoader* **do**
  Compute logits $\hat{y} \leftarrow \texttt{model}(X, V)$;
  Predict actions $\hat{a}^* \leftarrow \arg\max(\hat{y})$;
  Append $\hat{a}^*$ to $\mathcal{P}$ and y to $\mathcal{G}$;
Calculate classification accuracy and macro-averaged F1-score from $\mathcal{P}$ and $\mathcal{G}$;
Load historical training and validation accuracy and F1-score from checkpoint;
Plot training/validation accuracy over epochs;
Plot validation F1-score over epochs;
return *Accuracy, F1-score, and performance plots*

III. Results

Traditional UAV navigation systems in disaster environments struggle with dynamic obstacles, occlusion, and unpredictable terrain. These limitations severely reduce the reliability of rule-based systems, especially in unstructured and rapidly changing scenarios. Previous research [5] assumed static conditions or relied on handcrafted logic, making them unsuitable for real-time decision-making under uncertainty. The purpose of this study was to develop a vision-based navigation system for UAVs that can learn to predict discrete movement actions from depth image sequences, even in the presence of moving obstacles, occlusions, and environmental variation.

The proposed UAV system combines an RGB-D camera with convLSTM to enable real-time navigation and obstacle avoidance in simulated disaster scenarios. The ConvLSTM-based model achieved a training accuracy exceeding 98% within the first 20 epochs, but validation accuracy remained between 60% and 70% throughout training. The validation F1-score did not exceed 0.4, indicating limited generalization to unseen data. In real-world test sequences, the model correctly predicted forward actions in some cases, but also showed misclassifications due to label noise and temporal inconsistencies.

To enable future deployment in real UAV systems, several key enhancements must be addressed. First, the current action set should be extended to include roll and pitch actions, which are essential for controlling orientation and maintaining flight stability. Second, the discrete navigation actions used during training must be mapped to low-level control signals interpretable by UAVs, such as velocity vectors or thrust commands, to ensure executable outputs.

A. Model training

The training results in Fig. 4 show that the ConvLSTM model quickly achieved high training accuracy, exceeding 98% after a few epochs. However, validation accuracy remained low and unstable throughout the 80 training epochs with a batch size of 32 and learning rate of 1e-4, fluctuating between 60% and 70%. This discrepancy indicates that the model fits the training data well but fails to generalize to unseen validation data.

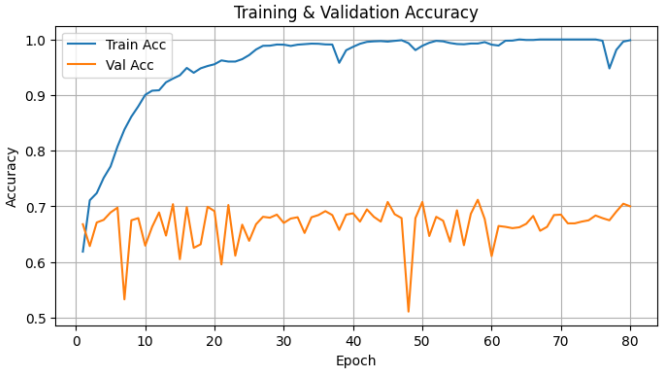

Fig. 4. Training and Validation Accuracy over 80 epochs.

The validation F1-score in Fig. 5 did not exceed 0.4 throughout the 80-epoch training process, suggesting the model, while performing well on training data, struggles to generalize to unseen samples. This low score highlights potential label inconsistencies or insufficient regularization. Despite training accuracy surpassing 98% early, the model's performance on diverse validation sequences remained limited. Several F1-score drops after epoch 40 imply the model began memorizing training data rather than learning generalizable patterns.

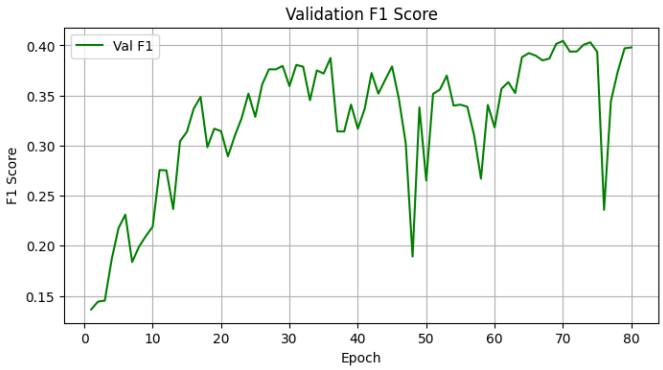

Fig. 5. Validation F1 score over 80 epochs.

The unstable behavior of the validation curve further suggests that the model's learned representations are not robust. Potential contributing factors include limited data diversity, insufficient regularization, and misclassification of labels in the ground truth, all of which introduce confusion during training and evaluation. These limitations can undermine the model's ability to learn consistent patterns that generalize beyond the training set. Additionally, these results reflect challenges commonly observed in previous studies [5] under limited data diversity and label inconsistencies.

B. Prediction results

Results of the simulated data are presented in Fig. 6 to Fig. 8. Fig. 6 shows an example of a ground truth label error in the dataset, where the assigned label does not correspond to the visual content. Fig. 7 displays a sequence in which the model, given nine input frames, predicts an action for the tenth frame that differs from the ground truth label. Fig. 8 demonstrates a correct classification, where the model's predicted action for the tenth frame matches the ground truth label.

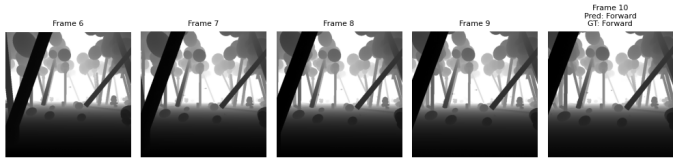

Fig. 6. Example of an incorrectly labeled sample in the dataset (ground truth label error).

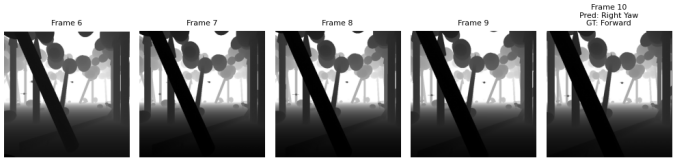

Fig. 7. Sequence showing a misclassification during training.

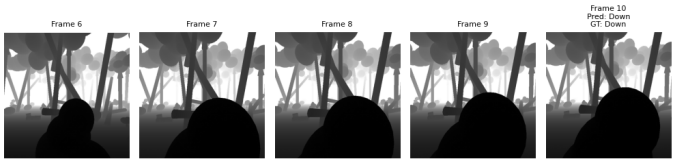

Fig. 8. Sequence showing correct model classification during training.

A real-world test sequence is shown in Fig. 9, which the model outputs correctly predicted the "Forward" action for frames F15 and F16, but incorrectly classifies frames F17 and F18 as 'Down', where the ground truth is 'Forward'. This example highlights both successful and failed predictions within real-world collected data. Overall, these findings reveal both the strengths and the current limitations of the approach when applied to sequential decision-making with noisy real-world data.

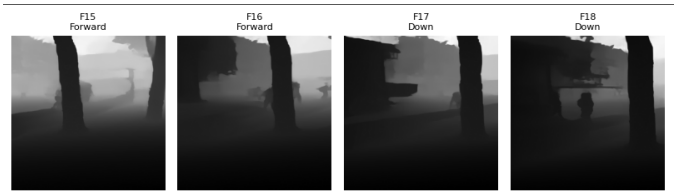

Fig. 9. Predicted and ground truth actions for frames F15–F18, with correct 'Forward' action and misclassified 'Down' action.

C. Inference Speed and Real-Time Suitability

In addition to accuracy and F1-score metrics, inference speed was evaluated to assess the model's suitability for real-time UAV deployment. As shown in Fig. 10, the average prediction time per frame was measured to be approximately 9.31 milliseconds, which is significantly below the 33.33 milliseconds threshold required for real-time inference at 30 frames per second (FPS), and even below the 16.67 milliseconds threshold for 60 FPS. This indicates that the model is capable of producing decisions faster than the visual input stream at both frame rates. These results underscore the importance of computational efficiency in time-critical scenarios such as disaster response. The system's responsiveness suggests its potential for integration into onboard decision-making pipelines in future UAV systems.

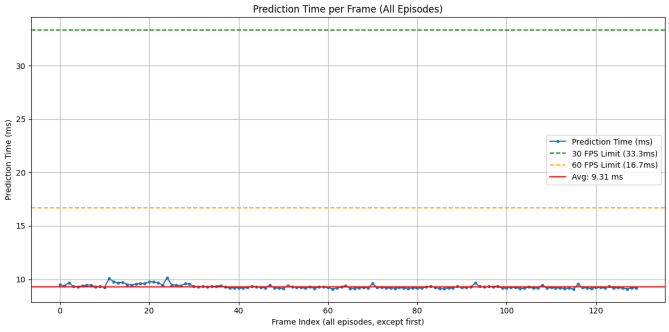

Fig. 10. Average prediction time per frame of the model.

These results indicate that the model is capable of producing decisions faster than the visual input stream at both standard frame rates. The low latency observed reflects the computational efficiency of the model when executed under high-performance conditions. Consistent prediction times across frames further reinforce the model's potential for responsive decision-making in time-sensitive scenarios.

However, such performance may not directly translate to embedded systems commonly used in UAVs, where computational resources are limited. Future work should therefore include benchmarking the model on UAV-compatible hardware and exploring optimization techniques such as pruning or model distillation. This would help evaluate feasibility and responsiveness under realistic deployment constraints.

IV. Discussion

The results indicate that the proposed ConvLSTM-based approach can achieve high training accuracy on the available dataset, but generalization to unseen data remains limited. The persistent gap between training and validation accuracy, along with low validation F1 scores, suggests that the model is susceptible to overfitting. This effect can be attributed to several data collection and labeling factors.

When contextualized within the broader field of UAV navigation and obstacle avoidance, the results reinforce findings from previous research. Earlier works [1] [2] [3] have noted the difficulty of deploying vision-based models in dynamic environments with occlusion and moving objects. This study confirms that real-world constraints such as temporal label inconsistencies, simplified control labels, and unrealistic test sequences can significantly hinder a model's generalization.

One of the main factors affecting model performance was the presence of label noise in the dataset. Inaccurate ground truth labels introduced ambiguity during training, causing the model to learn inconsistent or misleading associations. This directly impacted its ability to make correct predictions, even when the input sequence provided sufficient contextual information.

Another significant issue was the mismatch between training and testing conditions. Labels were assigned based on the movement of a target rather than the drone's actual behavior, and parts of the test data were collected using a handheld phone instead of simulating real UAV flight. These discrepancies reduced the alignment between the training objectives and the evaluation scenarios, which in turn limited the model's generalization capability.

References

[1] J. Singh, M. Dhuheir, A. Refaey, A. Erbad, A. Mohamed, and M. Guizani, "Navigation and obstacle avoidance system in unknown environment," in *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2020, pp. 1–4.

[2] A. A. Abdellatif, A. Elmancy, A. Mohamed, A. Massoud, W. Lebda, and K. K. Naji, "Pdsr: Efficient uav deployment for swift and accurate post-disaster search and rescue," *IEEE Internet of Things Magazine*, vol. 8, no. 3, pp. 149–156, 2025.

[3] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105 139–105 155, 2020.

[4] C. W. Zhang, "Spatio-temporal convlstm for crash prediction," 2021, accessed: 2025-05-23. [Online]. Available: https://medium.com/data-science/spatial-temporal-convlstm-for-crash-prediction-411909ed2cfa

[5] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015. [Online]. Available: http://arxiv.org/abs/1506.04214

[6] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020. [Online]. Available: https://arxiv.org/abs/1905.11946

Minh Hieu Tran received his Bachelor degree of science with a major in Computer Science and Engineering from Halmstad University, Sweden in 2024. He continued his studies at the same institution and is currently pursuing a master's degree, specializing in artificial intelligence.

Thomas Lundqvist completed his Bachelor degree of science with a major in Computer Science and Engineering at Halmstad University, Sweden in 2024. He is currently pursuing a master's degree at the same university in Sweden, specializing in artificial intelligence.

Jakub Espandr received his bachelor's degree in applied informatics Multimedia, in Corporate Practice from the University of Pardubice in the Czech Republic in 2023. He is currently pursuing a master's degree in applied informatics and data science for business at the same university, where he focuses on processing and analyzing UAV image data.

Min-Fan Ricky Lee received his Ph.D. and Master degree from Cornell University, USA in 1996 and 1991 respectively. His Ph.D. dissertation, "An intelligent computer vision control and target tracking system design of an agricultural grapevine pruning robot", received the national design gold award USA.

Dr. Lee is currently the assistant professor at the graduate institute of automation and control in the National Taiwan University of Science and Technology since 2007. He was the research officer in the National Research Council Canada from 1998 to 2007 and the research scientist at the department of mechanical engineering in the University of British Columbia, Canada from 1996 to 1998.

He served as the Past Chair, Chair, Vice Chair, Treasurer and Secretary of the IEEE Vancouver Section from 1998-2003 and was awarded the Outstanding Service Award for that service. He is endowed with the title of "International Taiwan Youth Ambassador" by the President of the Republic of China (Taiwan) on September 30, 2011.