INTERSECTION WAYPOINT IDENTIFICATION FOR VISION BASED INDOOR NAVIGATION OF NANO-SCALE DRONES

H.K.A.Y.D. Kodithuwakku

248111U

Master of Science(Major Component of Research)

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

February 2026

INTERSECTION WAYPOINT IDENTIFICATION FOR VISION BASED INDOOR NAVIGATION OF NANO-SCALE DRONES

H.K.A.Y.D. Kodithuwakku

248111U

Dissertation submitted in partial fulfillment of the requirements for the degree
Master of Science(Major Component of Research)

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

February 2026

DECLARATION

I declare that this is my own work and this Dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 18.02.2026

The above candidate has carried out research for the Master of Science(Major Component of Research) Dissertation under our supervision. We confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Dr. Chathuranga Hettiarachchi

Signature of the Supervisor: Date: 18/02/2026

Name of Supervisor: Dr. Sulochana Sooriyaarchchi

Signature of the Supervisor: Date: 18/02/2026

ACKNOWLEDGEMENT

This thesis would not have been feasible without the help, direction, and inspiration of numerous people and organizations, for which I am extremely grateful.

First of all, I would like to express my sincere gratitude to my supervisors, Dr. Chathuranga Hettiarachchi and Dr. Sulochana Sooriyaarachchi. Throughout this work, their visionary insights and meticulous dedication to detail improved the work significantly. I am especially grateful for the valuable time they dedicated to guiding me, despite their demanding schedules. Their unwavering faith in my abilities, helpful criticism, and patient coaching inspired me to overcome obstacles and hone my concepts into a solid, unified system.

Additionally, I owe a debt of gratitude to my examiners, Professor Chathura de Silva and Dr. Sunimal Rathnayake. I strengthened my arguments and clarified my contributions as a result of their thorough examination of my study during progress reviews and their insightful inquiries.

I would like to express my sincere gratitude to Senior Professor N.D. Gunawardena, Vice Chancellor of University of Moratuwa and Professor J.M.A. Manatunge, Dean of the Faculty of Engineering for their continued support towards administrative matters. I also gratefully acknowledge Dr. Uthayasanker Thayasivam, Head of the Department of Computer Science and Engineering, University of Moratuwa for his support. I would like to express my appreciation to all of the faculty and administrative staff for their effective management of academic and administrative issues, which enabled me to devote all of my attention to my project. I sincerely appreciate the generous support from the University of Moratuwa Senate Research Committee grant SRC/LT/2024/02.

I gratefully acknowledge Mr. Ignatious Peiris for the technical and moral support. I would also like to thank Mr. Kenul Perera for his efforts in creating video to simulation tool. I would also like to thank Mr. Vihanga Wickramage, Mr. Sandun Dushyantha for the synthetic environment building. I am grateful to my colleagues at the IntelliSense Lab for their technical support, early-stage discussions, and the camaraderie that made the research process both productive and enjoyable.

Personally, I am extremely thankful to my parents and my brother. My biggest source of strength has been their faith in my abilities. I want to express my sincere gratitude to my friends for their support and encouraging remarks.

Lastly, I would like to thank everyone who helped with this research, whether directly or indirectly: the creators of open-source datasets and tools, the peer reviewers whose work influenced my literature review, and the larger academic community whose findings served as the foundation for this thesis.

ABSTRACT

Nano-scale drones represent a promising platform for indoor inspection and search operations in GPS-denied, constrained environments, as their compact form factor enables access to narrow spaces. However, they require lightweight, resource-efficient autonomous navigation algorithms. Vision-based navigation offers an infrastructure-free substitute to GPS and external positioning systems. Simple vision-based navigational algorithms such as ring attractor, optical flow-based approaches, suffer from drift and cumulative error over time. Introducing semantic waypoints provides a mechanism to mitigate such drift.

This work proposes an egocentric, vision-based waypoint navigation framework that formulates waypoint detection as an intersection understanding problem, cast as a concept learning task. We first formulate the intersection identification problem as a multi-class classification. To ensure a model can be trained with limited data and still preserve the conceptual understanding, each intersection is decomposed into its fundamental directional components: left, right, and forward. This novel multi-label classification is advantageous because it can extend to zero-shot inference and complex junction types. Further, we augment the limited real data with synthetic visual data collected from a safer digital twin environment which replicates the real-world scenes. We design five experimental setups in which model weights are initialized from Kaiming He, ImageNet or synthetic pre-trained weights obtained using the Digital Twin dataset.

We show that, (i) Intersections can be used as waypoints for the navigation of nano-drones. (ii) Multi-label classification outperforms multi-class formulation in intersection identification. (iii) Model trained using digital twin data alone can yield an F1 score of 0.6 in real-world predictions, compared to 0.5 from a random classifier. (iv) Introducing limited real-world training data can bring the model up to near-perfect accuracy with faster convergence. (v)The multi-label classification of junctions provides insightful visual explanations ensuring reliability and generalizability of the conceptual framework. Finally, the proposed framework is effective in environments such as libraries and supermarkets where structured aisle naturally exists and GPS is unavailable. This method offers an infrastructure and drift-resistant solution for the navigation of nano-scale drones by leveraging intersections as semantic waypoints.

Keywords: ego-centric vision, explainability, concept learning, nano-scale drones

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AVL	Audio-Visual-Language
CfLib	Crazyflie Python Library
CNN	Convolutional Neural Network
COTS	Commercial-off-the-shelf
CRTP	Crazyflie Real Time Protocol
DORY	Deployment ORiented MemorY
EVC	Early Visual Cortex
FLA	Fast Lightweight Autonomy
fMRI	Functional Magnetic Resonance Imaging
GPS	Global Positioning System
IIR	Infinite Impulse Response
IMU	Inertial Measurement Units
INDI	Incremental Nonlinear Dynamic Inversion
IoT	Internet of Things
MCU	Microcontroller Units
NEMO	NEural Minimizer for pytOrch
NUC	Next Unit of Computing
OPA	Occipital Place Area
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative
PPA	Parahippocampal Place Area
PULP	Parallel Ultra Low Power
RSC	Retrosplenial Complex
SaR	Search and Rescue
SLAM	Simultaneous Localization and Mapping
ToF	Time-of-Flight
TSM	Temporal Shift Module
UAV	Unmanned Aerial Vehicles
VLA	Vision-Language-Action Navigation
VQA	Visual Question Answering
WPN	Waypoint Prediction Networks
YOLO	You Only Look Once

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Vehicles that can fly without a human pilot on board are known as Unmanned Aerial Vehicles (UAV) [8]. UAVs are becoming more and more popular in both military and civilian uses due to their high mobility, ease of deployment, and minimal maintenance. UAVs can also carry a variety of possible sensors for any important operations. Wildfire monitoring[9], crowd monitoring, target tracking, commodities delivery, medical assistance, Search and Rescue (SaR) [10], emergency cellular deployment, and intelligent transportation are just a few of the numerous uses for UAVs.

1.1 UAV Navigation

UAV navigation can be thought of as a process in which robots plan how to swiftly and safely reach to the target place, primarily based on their current location and environment. Inertial navigation, satellite navigation, and vision-based navigation are the three basic categories under which a variety of navigation techniques have been presented[8]. Navigation is a challenging task for a drone due to many reasons. The complexity of operating in a variety of frequently unforeseen conditions while preserving safety and autonomy makes UAV navigation difficult. Drones must rely on onboard sensing and awareness in many situations where Global Positioning System (GPS) signals are inconsistent or nonexistent, such as indoor spaces, urban canyons, or densely populated places. Reliable localization and path planning are made more difficult in these environments by the frequent presence of dynamic obstacles, tight spaces, and unclear visual clues. Furthermore, the quick and highly linked dynamics of UAVs necessitate precise and timely perception-decision-control loops. Sensor noise, motion blur, and variations in lighting all affect perception, which can impair situational awareness. When combined, these difficulties make robust and generalizable UAV navigation a challenging topic, especially in locations with limited GPS and difficult visual conditions.

Conventional navigation systems use global positioning sensors to determine a robot's location and orientation in its surroundings. While Inertial Measurement Units (IMU) IMUs and wheel encoders give continuous motion estimates, allowing for robust path tracking and control, GPS offers absolute global localization in outside environments. Because of these systems' dependability, advancement, and low computational cost, they have been extensively used in robotics, automotive, and aerospace applications. However, GPS signals are not available in every location, such as indoor, underground, underwater, dense forests and urban canyons. Due to developments in deep learning and computer vision, vision-based navigation has become a

strong substitute for conventional sensor-dependent systems. Cameras are perfect for small aerial robots operating in challenging or GPS-denied areas because they offer extensive, dense environmental information at low cost and weight. Robots can recognize landmarks, follow motion, infer depth, and comprehend scene semantics with growing robustness owing to modern learning-based and geometric vision algorithms.

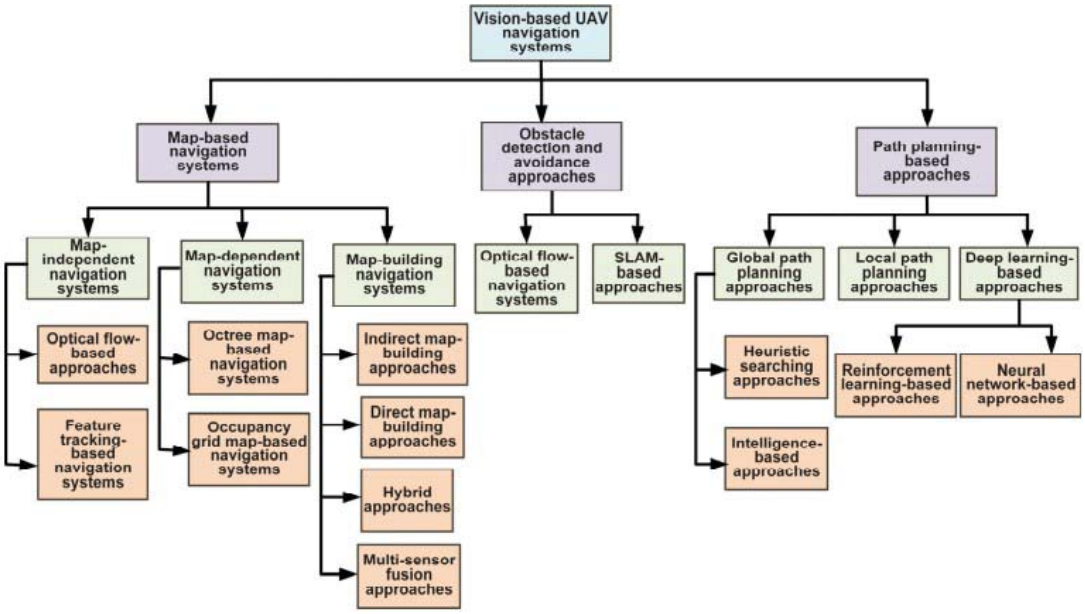


Fig. 1.1: Taxonomy of vision-based UAV navigation systems, reprinted from[1].

According to the paper [1], vision-based UAV navigation systems can be primarily categorized into three main approaches. As depicted in the Figure 1.1, there are map-based navigation systems, obstacle detection and avoidance approaches and as the third path-planning based approaches. The UAV can navigate with detour behavior and plan movement strategies due to the map-based system, which is based on a predetermined map and a laid-out environment. The degree of detail in a map can range from a 3D model of a complete location to a diagram showing how its components are connected. Three types of map-oriented navigation systems exist: map-independent, map-dependent, and map-building-based. The second category which is obstacle detection and avoidance, is an essential part of autonomous navigation because it can recognize and communicate vital information about nearby obstacles, lowering the possibility of crashes and pilot error. Two types of obstacle detection and avoidance systems exist: optical-flow, Simultaneous Localization and Mapping (SLAM) SLAM-based. The final category is the path-planning approach which determines the most efficient path from the starting to the destination point. Two types of path planning-based approaches exist: global path planning, local path planning and deep learning based approaches.

There are lightweight vision algorithms such as the ring attractor, optical flow and event based methods which are used in navigation. However, there are limitations of

these methods. For example, we observe a significant drift in events only and standard frames only settings in Figure 1.2 from the paper [2]. Ring attractor network also suffers from this drifting issue due to the rapid changes in velocity as mentioned in the literature[11]. Slow drift also affects the optical flow-based pose estimate as it just observes at the relative velocity of features[12].

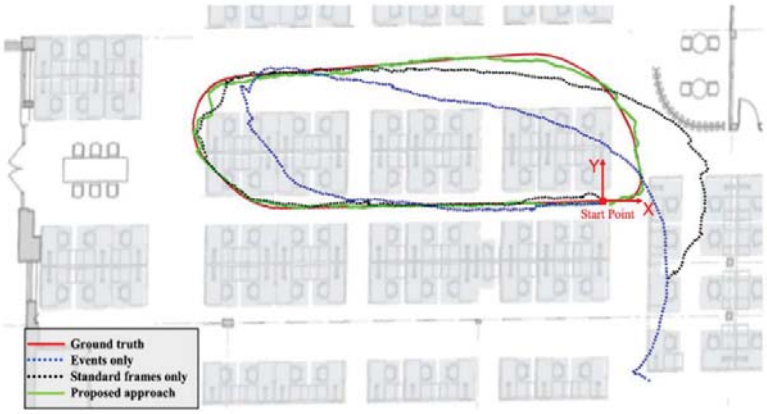


Fig. 1.2: Drift trajectories of event-only and frame-only navigation methods in loop closing, reprinted from [2].

1.2 UAVs taxonomy based on the size

TABLE 1.1: UAVS TAXONOMY BY VEHICLE CLASS-SIZE

UAV Class	Diameter:Weight[cm:kg]	Power [W]	Onboard Device
Standard-size	$\sim$50 : $>$1	$>$100	Desktop
Micro-size	$\sim$25 : $\sim$0.5	$\sim$50	Embedded
Nano-size	$\sim$10 : $\sim$0.01	$\sim$5	MCU

Note. Classification of UAVs based on physical size, power budget, and onboard computational capability from [3].

There are several types of drones, depending on their size. Major size description is mentioned in the Table 1.1. Standard and micro-sized drones, even when functioning independently, already exhibit impressive onboard intelligence. They can carry powerful processors (such as CPUs and GPUs)(which have a peak throughput of 200 TOPS/s at 40W) and sophisticated sensors (such as high-resolution cameras and lidars) because of their large size (more than 25cm) and weight (more than 0.5kg)[4]. For instance, Fast Lightweight Autonomy (FLA) platform uses a number of sensors, such as laser-range finders and lidars, in addition to a strong onboard computer[13]. The ASL-Flight[14] uses an Intel Next Unit of Computing (NUC) as its primary compute resource and is based on the DJI Matrice 100 platform. The quad described in [15]

is 250 g in weight and uses a monocular camera and an in-built inertial measurement unit (IMU). This platform includes a Qualcomm Snapdragon platform. The Agilicious UAV which was introduced to support the next generation of scientific and industrial quadrotor research has Jetson TX2[16]. These advanced drones can navigate complex environments by leveraging sophisticated sensors and navigation approaches, as described in Section 1.1.

Nano-scale drones are an emerging research area with many use cases. Applications that are unattainable for their larger equivalents are made possible by reducing the size of UAVs, which has several benefits. This includes the capacity to maneuver in confined places[17], and their modest weight guarantees safe operation in close proximity to people [18] in indoors. Furthermore, a key component of widespread adoption is the affordability of manufacturing small, basic electrical components. With these features, small flying robots have the potential to become ubiquitous in daily life, particularly in the rapidly expanding Internet of Things (IoT) world, which is made up of a huge network of wirelessly connected smart gadgets[4]. However, the power and hardware limitations of nano-scale drones make it difficult to use conventional sensors and sensor fusion methods. Consequently, it is challenging to implement many of the vision-based approaches covered in Section 1.1 on such systems except for the simple algorithms. Using AI algorithms on such resource-constrained platforms presents a number of difficulties: Onboard computation is confined to low-power processing devices like Microcontroller Units (MCU) and lightweight sensors due to limited payload capacity which limits the size of printed circuit boards and batteries.

1.3 State-of-the-Art Navigation for Nano-Scale Drones: PULP DroNet

PULP-DroNet is a deep learning-driven visual navigation engine that provides pocket-sized and nano-scale drones with cutting-edge autonomous navigation capabilities. Constructed on Parallel Ultra Low Power (PULP) hardware, like the GAP8 SoC, and based on an effective convolutional neural network derived from DroNet[19], it performs all perception and control onboard, enabling fully autonomous flight without the need for external computers, motion capture devices, or GPS. For the nano-scaled drones, PULP-Dronet has been identified as the state of the art[20, 21].

DroNet is a vision-based navigation system that predicts a collision probability and a yaw (steering) angle for reactive obstacle avoidance. This ResNet-based architecture was first presented in [19]. It is a small, eight-layer residual convolutional neural network that makes it possible for UAVs to navigate safely in dense urban settings. DroNet concurrently predicts a collision probability that enables the UAV to recognize dangerous circumstances and react appropriately, as well as a desirable steering angle to direct path following and obstacle avoidance from a single monocular vision.

The PULP research group then carried out a number of studies with an emphasis

on autonomous navigation for nano-scale drones. The commercial Crazyflie quadrotor from Bitcraze[1] was chosen as the target platform for this line of research. The PULP-Shield, a lightweight, modular, and Printed Circuit Board (PCB) with a highly optimized layout and a form factor compatible with the Crazyflie nano-sized quadrotor [21], was the first significant result. This work demonstrated onboard CNN inference on the Crazyflie 2.0 with a power consumption of only 64–284 mW and a throughput of 6–18 frames per second, marking the first successful deployment of DroNet on the PULP-Shield.

With a focus on automating the entire Convolutional Neural Network (CNN) deployment pipeline, DroNet v2 later added support for the commercially available AI-deck[2]. To enable dependable onboard execution on a flying nano-drone, considerable model compression, complexity reduction, and fine-grained manual optimization were needed. Using both academic toolchains (NEural Minimizer for pytOrch (NEMO) and Deployment ORiented MemorY (DORY)) and commercial solutions (GAPflow by GreenWaves), the authors suggested specialized approaches and software tools to expedite deployment on low-power multi-core systems-on-chip (SoCs) [3].

Deploying numerous tasks involving AI concurrently was difficult because of the incredibly small payload capacity of nano-UAVs, which limits onboard computation to ultra-low-power microcontroller-class devices with severe memory and compute constraints. As a result, DroNet v3 concentrated on reducing and optimizing AI tasks while maintaining reliable flight behavior in practical operating scenarios[20].

1.3.1 DroNet on board control framework

This subsection explains how the Crazyflie navigates using dronet's on-board inference. The dronet model is deployed onto the AI deck's GAP8 microcontroller. AI deck's monochrome camera captures the images and GAP8 runs the inference on board. The outputs of the microcontroller are the steering angle and the probability of collision. This information is sent via UART to the flight control(STM32) of the Crazyflie to perform the navigation.

This process is illustrated in the Figure 1.3 from the paper [3]. The two pieces of data (data[0] and data[1]) linked to the CNN's output are first dequantized by multiplying them by the scaling constants that were produced during the quantization procedure. Next, I(k) is calculated. The authors devised I(k) to penalize the long-term effects of obstructions in the field of view. If the probability of collision is high, I(k) value will increase with time. This will parallelly increases the drone's confidence on actually facing an obstacle. It is the other way when the field of view is free of obstacles. To convert the probability of collision to forward velocity, researchers followed

[1] https://www.bitcraze.io/products/crazyflie-2-1-plus/
[2] https://store.bitcraze.io/products/ai-deck-1-1

a square-low penalizing velocity equation. Moreover, to mitigate the noises, a first order Infinite Impulse Response (IIR) filter defined by the coefficient alpha(α1=0.6, α2=0.7[3]) was used. Finally, the drone's flight controller receives the updated flight setpoint represented by the filtered forward velocity and steering rate values (vset and ωset).

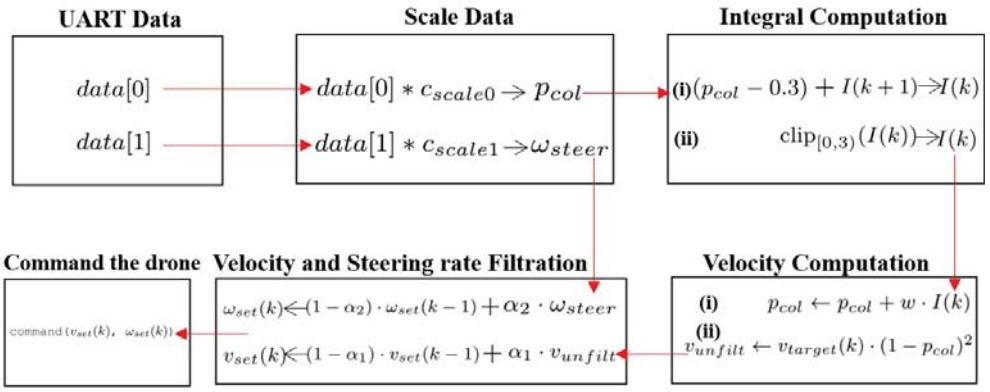


Fig. 1.3: PULP Dronet's data processing framework until the setpoint is communicated to the drone's commander[3]

This state-of-the-art navigation framework for nano-scale drones offers a number of significant advancements. It has proven to perform well in terms of navigation accuracy and has been mostly utilized for reactive obstacle avoidance. However, without a high-level understanding of the decision points such as junctions, DroNet formulates the navigational problem as an obstacle avoidance task, rather than a way-point based planned navigation task. The authors specifically pointed out that the model in the original DroNet research [19] chose a random route at intersections, which is not enough. Navigating in complex environments such as intersections is a multi-layered task where i. The high level vision based way-points need to be detected, and ii. The navigation decision is made. The decision can be either based on a preplanned mission plan, or as a high-level real-time piloting decision, where the autonomous robot awaits a decision from the pilot after reaching the decision point. An example mission planning and navigation scenario is given in Fig. 1.4d, where the mission plan of the drone starts from the coordinate A and ends at the coordinate G.

In contrast to the classical coordinate-based mission plan depicted in Fig. 1.4a,b we propose a visual way-point based mission plan Fig. 1.4c. In traditional coordinate based navigation trajectory is defined by position or velocity points. In contrast, the proposed conceptual framework uses a high-level ego centric vision based mission plan as described in Fig. 1.4c. The actions that the drone should take during an intersection are highlighted in red.

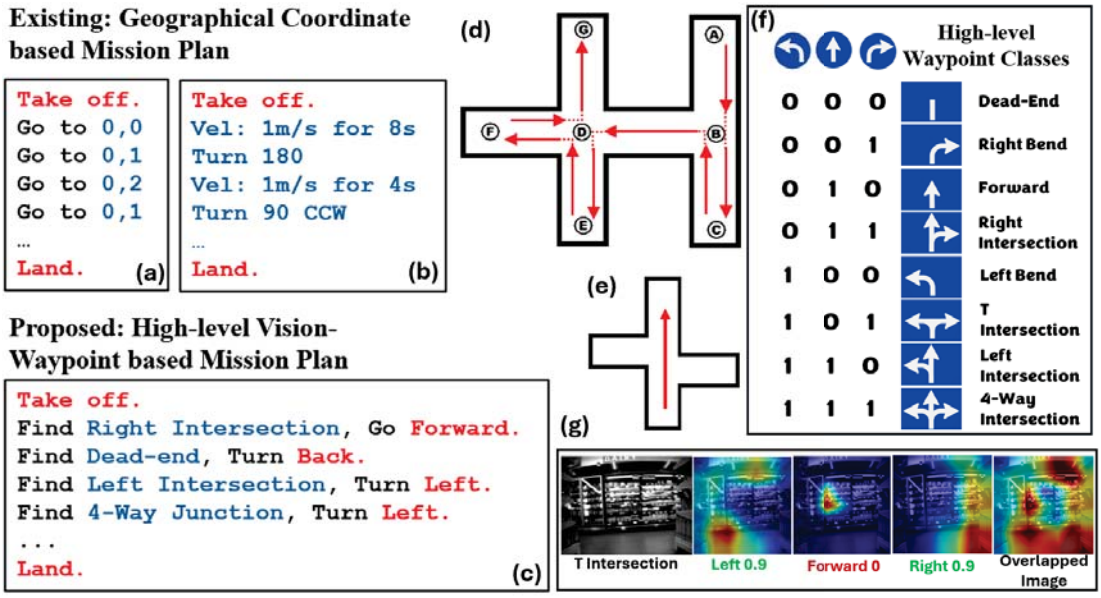


Fig. 1.4: Overall Concept(a) Global coordinate based mission plan (existing) (b) Global coordinate velocity setpoint based mission plan (existing) (c) Egocentric vision-waypoint based mission plan (proposed) (d) Mission plan of the drone (e) Staggered junction - an example of a complex junction which can be inferred zero shot. (f) Orthogonal decomposition of junction vision-waypoints to left-forward-right navigability (g) Validation of conceptual understanding of left-forward-right navigability of the proposed model with Grad-CAM.

1.4 Problem Statement, Objectives and Contributions

In this research we address the necessity for vision based way point navigation in resource constrained applications for nano-scaled drones.

The objectives of this research is to

1. Identify existing vision based navigation methods for nano-scale drones.
2. Implement and evaluate selected vision based navigation methods and find the limitations
3. Develop a vision based navigation method addressing the identified limitations.
4. Assess and improve the robustness of the vision based navigation waypoint perception.

In this thesis, I make the following key contributions:

- Successfully demonstrating that the nano-scale UAV equipped with a low-resolution camera can be used to identify intersections, which can be used as waypoints for vision-based navigation.

- Illustrating multi-label classification, achieved by decomposing high-level intersection categories, outperforms conventional multi-class classification in intersection identification.
- Demonstrated that the proposed model attends to semantically meaningful intersection cues, with Grad-CAM visualizations exhibiting the human brain's occipital place area like navigational affordance representations, thereby enhancing interpretability.
- Evaluating the generalization capability of the proposed model across unseen real-world data.
- Collecting and publishing both synthetic and real-world datasets for intersection-level navigation tasks[3].
- Publishing a paper at the International Conference on Artificial Intelligence in Information and Communication (ICAIIC26).

[3] https://github.com/yeko31/ego-intersect/

CHAPTER 2

LITERATURE SURVEY

Egocentric vision has been frequently adopted in autonomous navigation research due to its close analogy to human perception and decision making. When navigating complicated and novel environments, humans solely rely on egocentric visual cues without explicit access to global positional information. Robust and adaptable navigation in a variety of environments is made possible by this capacity to immediately interpret navigational affordances from first-person visual observations. Another bio-inspired example for this is the honey bee navigation. They store landmark memories in an egocentric frame of reference[22]. Inspired by this natural intelligence, recent robotic systems have increasingly employed egocentric vision as the primary sensing modality.

2.1 Vision based Intersection Navigation

2.1.1 Egocentric Vision

Egocentric vision is a first-person perspective visual input where the camera captures the observer's viewpoint for navigation tasks. There are several navigation strategies that can be used in this setting such as path integration and landmark processing. In contrast to the allocentric method, the egocentric method relies on a first-person perspective and path integration which updates orientation and position. Egocentric vision has been used in many applications today. For example, current applications include wheelchair navigation[23], humanoid navigation[24], 3D hand pose estimation[25] and etc. Additionally, Plizzari et al. [26] outlined a number of expected study findings for egocentric AI, a device that can be worn intended for on-site multimodal sensing from the viewpoint of the user. Using artist-drawn illustrations, they highlighted how such ego-based support could influence everyday living. Their work categorizes five separate use cases: EGO-Home, EGO-Worker, EGO-Tourist, EGO-Police, and EGO-Designer.

Navigating securely in the real world from egocentric observations is an ability that we humans possess, yet incredibly difficult for robots to master. This is mostly because real-world scenarios involve a wide range of complicated circumstances. Such characteristics are vital for different applications like humanoid robotics, VR / AR, and assistive navigation[24]. There are several research that explores the possibility of egocentric navigation for both humans and robots. Qiu et al.[27] focused on developing a system which predicts human movement and feelings of uncertainty during navigation from a first-person perspective, integrating sensory data and cognitive factors to better understand and assist human navigation in complex environments. Wang et al.[28]

focused on forecasting a person's future walking path utilizing egocentric vision from wearable cameras, leveraging diffusion models and scene semantics to accommodate for unstructured settings and various alternative trajectories. Pan et al.[24] offered an approach that predicts both translations for collision-free navigation and rotations for learning active information-gathering behaviors similar to the head-turning activities people perform to enhance perception. Their extensive trials demonstrated that their model can generalize to unfamiliar surroundings while learning human-like navigation behaviors, including waiting/slowing down, rerouting, and scanning the area for traffic. As explained in the Chapter 2, we use egocentric vision for this research because of the inspiration we got from the natural egocentric navigation.

2.1.2 Visual Way Point based Navigation

The state of the art of egocentric vision navigation for nano-drones is PULP-Dronet. As outlined in the Section 1.3, understanding high-level decision points is a limitation of Dronet. Therefore, visual way points for high-level understanding are important. Visual way point based navigation uses optical features as intermediate targets to guide a robot to a destination without relying on GPS or maps. Waypoints facilitate sequential navigation in dynamic or unfamiliar environments, such as indoor spaces, by acting as reference points indicated by visual cues. This method combines control techniques for path execution with computer vision for waypoint detection and selection. The use of waypoints helps mitigate drift errors, which remain a major challenge in vision based UAV navigation, as discussed in Section 1.1.

Instruction-guided visual navigation is a major subsection of visual way point based navigation. This is also known as Vision-Language-Action Navigation (VLA) , where robots understand natural language instructions by interpreting visual cues in the environment. Krantz et al.[29] effectively showed that using waypoint prediction to move toward linguistically-motivated, high-level action spaces not only offers advantageous real-world execution features but also establishes a new standard for instruction-guided navigation in continuous environments. Their primary contribution is the creation of a class of language-conditioned Waypoint Prediction Networks (WPN) that use panoramic vision and natural language instructions to anticipate relative waypoints. To enhance the use of vision and language navigation, using real-world panoramic RGB-D photos from the Matterport3D dataset, Anderson et al.[30] presented the Matterport3D simulator, a novel large-scale interactive reinforcement learning environment. This simulator outperformed earlier synthetic environments by providing a special blend of repeatability, interactivity, and visual realism. The AerialVLN task [31] was created to address difficult situations in UAV-based delivery or aerial navigation surveillance. This work suggested a city-level UAV navigation environment where agents must use ego-centric view visual perceptions to follow natural

language commands. AerialVLN offers a crucial platform for furthering research into reliable, practical aerial navigation skills by concentrating on continuous navigation in realistic city environments and offering a challenging dataset with lengthy pathways and varied instructions.

Audio-Visual-Language (AVL) navigation enables robots to navigate in three-dimensional space using sound cues and as well as RGB-D features. The research [32] presented Audio-Visual Waypoint Navigation (AV-WaN), a unique deep reinforcement learning framework for the AudioGoal challenge, in which an agent uses both visual and auditory inputs to navigate to a sound source in an unmapped 3D environment. The model's organized acoustic memory is a crucial component that, when combined with a geometric map, is essential for effective navigation and abstraction away from particular training sounds.

The study by [33] addresses the drift-related errors seen in previous context-specific techniques by methodically examining the design requirements of a trustworthy fiducial-marker-based localization framework for autonomous indoor navigation. The authors employ fiducial markers as visual landmarks to enable robust robot localization and navigation. The robot uses this Marker Network Map (MNM), which is created by placing these markers at crucial points like hallways, doors, and crossroads, to locate itself, create the best routes, and travel on its own. However, such markers are not always available in real-world environments to use them as landmarks. In contrast, we propose using natural intersections as waypoints to support navigation task.

2.1.3 Intersection Identification

The researchers approach the intersection identification task from several angles. Giusti et al. [34] classified drone photographs into three class types (turn left, turn right, and go straight) so that a quadrotor could follow an outdoor hiking course. Researchers acquired a hiker with three GoPro cameras mounted on the head at left, forward and right angles. These data were then categorized into left,forward,right classes. Authors have used a deep neural network with 10 layers, 150k weights, 500k neurons and 57M connections as a classifier for the task. This classifier takes an image as input and put one of three classes indicating whether a trail is visible on the left, right or center side of the image. This approach is one of our motivations to frame the problem as multi-label which will be discussed in Section 3.4.6.

Garcia et al.[35] proposed a method which use typical vision techniques with geometric shape analysis. First, video frames were captured from the drone's front camera. Next, significant lines in the scene were extracted using Hough Line Transform and Canny edge recognition. After that, these lines were processed to find right triangles created by the intersection of the floor and walls, which were used to accurately identify upcoming intersections. The size of these intersection triangles was analyzed,

particularly the vertical side, which inversely correlates with the distance to the intersection. According to the findings, the drone was unable to accurately identify the specific intersections when it is positioned closer to junctions. Advancing his previous research to address the limitation of the environmental lighting condition, he proposed a Convolutional Neural Network classifier which ran on the base station to classify intersections and dead-ends[36]. Garcia et al. [7] further have framed the intersection detection problem as an object detection task. They have labeled 1,484 photos from a wide range of corridors, with bounding boxes for different intersection types, doors, poster boards etc. A You Only Look Once (YOLO) based object detection model has been used to successfully detect the intersections.

Padhy et al. [37] also have reported successful likelihood estimations for class labels such as stop, shift left, shift right, and move forward, which can be used for autonomous maneuvering of the drone in corridor environments. The authors have used Dense-Net-161 and fed real-time images from a front-facing camera as the input.

Mansouri et al.[38] suggested an approach that uses a monocular camera and Convolutional Neural Networks(CNNs), specifically utilizing transfer learning with AlexNet, to identify tunnel junctions in underground mines. The CNN is based on a multi-class classification scheme of four classes which are left, right, left & right and no junction. The author followed a transfer learning approach since the real-world data was limited[38].

2.2 Performance Metrics Evaluation

To evaluate the performance of our framework, we utilize some of the performance metrics that have been widely used in research applications. For example, Dack et al.[39] used hamming loss, precision, recall and F1-score to evaluate a zero-shot multi-label classifier for medical image analysis, particularly using Covid-19 Computed Tomogrphy(CT) scans and unprocessed reports. The following equations are sourced from the website[1] to evaluate our multi-label classification strategy.

[1] https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

2.2.1 Hamming Loss

The Hamming loss is the fraction of labels that are incorrectly predicted.

$$L_{Hamming}(y, \hat{y}) = \frac{1}{n_{\text{samples}} * n_{\text{labels}}} \sum_{i=0}^{n_{\text{samples}}-1} \sum_{j=0}^{n_{\text{labels}}-1} 1(\hat{y}_{i,j} \neq y_{i,j})$$

$\hat{y}_{i,j} = Predicted\ value\ for\ the\ jth\ label\ of\ a\ given\ sample\ i$

$y_{i,j} = Corresponding\ true\ value$

$n_{samples} = Number\ of\ samples$

$n_{labels} = Number\ of\ labels$ (2.1)

2.2.2 Precision

Precision is the proportion of all the model's positive classifications that are actually positive. It is mathematically defined as:

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP} \tag{2.2}$$

2.2.3 Recall

The true positive rate (TPR), or the proportion of all actual positives that were classified correctly as positives, is also known as recall. It is mathematically defined as:

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN} \tag{2.3}$$

2.2.4 F1 score

As a harmonic mean of precision and recall, the F1 score can be understood as having a maximum value of 1 and a minimum value of 0. Recall and precision both contribute equally to the F1 score. The F1 score is calculated as follows:

$$\text{F1} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{2.4}$$

2.2.5 Grad-CAM for interpretability

Deep Convolutional Neural Networks have been widely used in many research and have achieved significant breakthroughs in various domains such as image classification, image captioning, visual question answering, embodied question answering and etc[40]. Though the models in these applications achieve significant accuracy, they

are difficult to interpret since they cannot be broken down into individually intuitive parts[41]. As a result, when today's intelligent systems fail, they often fail without warning or explanation, leaving the user staring at an unclear output and wondering why the system did what it did[40]. CNNs are one type of black box model. This indicates that the expert is unaware of the reasoning behind the final prediction. It is dangerous to utilize CNNs in high-risk or safety-critical applications such as autonomous vehicles and healthcare without first creating ways to explain their conclusions. There is low assurance that they will continue to make accurate decisions without knowing the reasoning behind a decision[42]. Grad-CAM, a class-discriminative localization technique that produces visual explanations for any CNN-based network without requiring architectural modifications or retraining, was presented by Selvaraju et al.[40] to address this problem.

We present a comparative literature synthesis in Table 2.1 to summarize the size of the drone, camera resolution, classification methods, intersection identified or not, explainability and number of training epochs. As limitations and gaps of the existing research, to the best of our knowledge, the explainability of these trained models and their relation with human cognitive system for navigation is not studied. Furthermore, high-level understanding frameworks for nano-scale drones remain unexplored, and datasets covering diverse indoor intersection types are limited.

TABLE 2.1: COMPARATIVE SYNTHESIS OF RELATED LITERATURE

Paper	Size of the Drone	Camera Resolution	Classification	Intersection Identification	Explainability	Training Epochs
Giusti et al.[34]	Standard	High(RGB) drone	Multi-class	No	No	90
Garcia et al.[35]	Standard	Med(RGB) drone	Image Processing (Contours & Lines)	Yes	No	-
Padhy et al.[37]	Standard	Med(RGB) drone	Multi-class	No	No	500
Garcia et al. [36]	Standard	-	Multi-class	Yes	No	30
Garcia et al.[7]	Micro	High(RGB) drone	Object Detection	Yes	No	-
Mansouri et al.[38]	Micro	High(RGB) data	Multi-class	Yes	No	6
Our Work	Nano	Low(Gray)	Multi-label	Yes	Yes	11

CHAPTER 3

METHODOLOGY

In this chapter, we discuss the hardware framework that we use for our experiments in Section 3.1, the software stack in Section 3.2. Next, details on replicating the state-of-the-art PULP-Dronet are presented in Section 3.3. Finally, the methodology of the proposed intelligent waypoint navigation is described in Section 3.4.

3.1 Hardware Framework

Crazyflie 2.1+ drone is considered to be the most widely used commercially available nano-drone. This is also found to be the state of the art nano-scaled drones in [3, 20, 21].

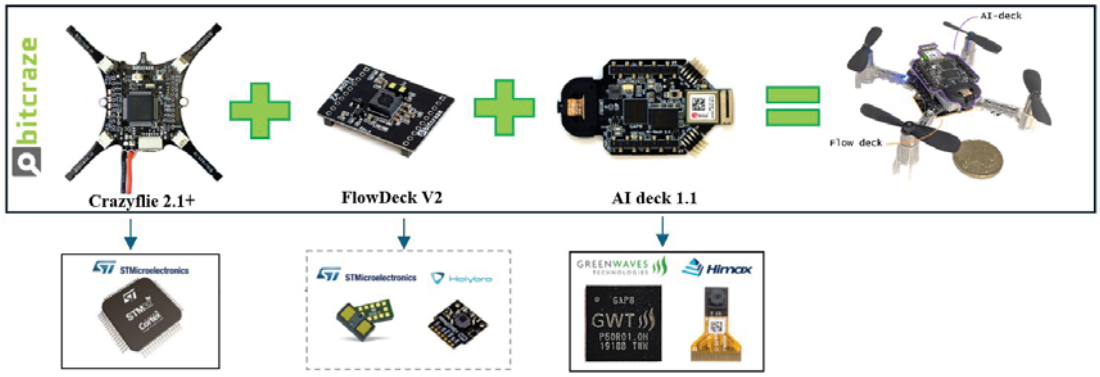


Fig. 3.1: The robotic platform Crazyflie 2.1+, Flow deck and AI-deck with their microcontrollers, adapted from[4]

3.1.1 UAV Platform

The drone platform which we used in this research is the Commercial-off-the-shelf (COTS) Crazyflie 2.1+ from Bitcraze[1]. The miniaturized Crazyflie 2.1+ weighs 29g, has a size of 92x92x29mm (motor-to-motor and including motor mount feet). The drone has a total maximum recommended payload of 15g. The Bosch BMI088 inertial measurement unit (IMU), which combines an accelerometer and gyroscope, and the STM32F405 MCU serve as the flight controller for this open-source and open-hardware drone. The STM32 MCU is responsible for all low-level flight controller functions, including low-level control, state estimation, and sensor interface. It can operate at up to 168 MHz and incorporates 48 kB SRAM and 128 kB flash. The Crazyflie 2.1+ also integrates a nRF51822 MCU for 2.4 GHz ISM band radio communication.

[1] https://www.bitcraze.io/products/crazyflie-2-1-plus/

3.1.2 Sensor Peripherals

Our configuration extends the robotic platform with two COTS pluggable decks from Bitcraze: The Flow deck and AI-deck. Together with a VL53L1x Time-of-Flight (ToF) ranging module that measures distance from the ground, the 3.5 g optical flow sensor has a low-resolution, down-looking PMW3901 optical flow visual sensor that allows the drone to detect motions in the 2D plane parallel to the floor. By improving the accuracy of onboard state estimation, these inputs reduce long-term drift.

The commercially supported version of the PULP-Shield research prototype(first presented in [43]), is the AI-deck. The main onboard processing device, weighing 4.4 g, is in charge of carrying out complex AI tasks like the CNNs suggested in this book. With an energy-efficient GAP8 processor [44], off-chip DRAM and Flash memory (8 MB and 64 MB, respectively), a QVGA resolution low-power grayscale camera (such as the Himax HM01B0 sensor), a UART communication channel between the STM32 and the GAP8, and a flexible ESP32-based WiFi module, the PCB expands the nanodrone's onboard capabilities. The complete hardware framework is illustrated in Figure 3.1.

3.2 Software Stack

3.2.1 Crazyflie PC Client

Flashing and controlling the Crazyflie are made possible by the Crazyflie PC client. Cfclient can be used to maneuver the drone using a gamepad, log the flight parameters, monitor real time positioning and battery levels, update flight firmware and etc. The Crazyflie Python Library (CfLib) project manages the Crazyflie Real Time Protocol (CRTP) implementation and communication with Crazyflie.

As illustrated in Figure 3.2, 1. Shows the connection status, 2. Connect/disconnect, scan and the drop-down connection list as well as the address of the drone 3. Battery and link quality (from 0% to 100%), 4.Tabs with specific functionalities, 5.The content for the active tab, 6. Emergency stop button, the motors will stop when clicked.

Figure 3.3 illustrates the general framework of the Crazyflie[2]. The module stabilizer includes key building blocks of estimator, commander and controller. In quadrotors (and robotics in general), state estimation is crucial. The Crazyflie must first determine its roll, pitch, and yaw angles. When the drone is flying towards a particular direction, the controller should accurately evaluate the status of the existing angles and make adjustments. A good position estimate also becomes crucial for a step higher in autonomy since it's crucial to move from A to B consistently.

[2] https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/commanders_setpoints/

Fig. 3.2: User interface of CFClient for connecting to, controlling, and logging data from the Crazyflie.

3.2.2 Estimator Module

The Crazyflie firmware has two different kinds of state estimators: a Complementary Filter and an Extended Kalman Filter.

The complementary filter, which typically uses only the IMU output of the gyroscope (angle rate) and the accelerometer, is considered to be extremely lightweight and efficient. The estimator has been expanded to incorporate the Zranger deck's[3] ToF distance measurement as input. The Crazyflie's attitude (roll, pitch, and yaw) and altitude (in the z direction) are the estimated outputs. These values are intended for manual control and can be utilized by the controller.

Compared to the complementary filter, the (extended) Kalman filter is more sophisticated since it can handle more sensor inputs from both internal and external sources. It is a recursive filter that uses incoming measurements, the measurement model, and the system model to estimate the present state of the Crazyflie (along with a forecast standard deviation of the noise).

In summary, because of the multiple state estimate possibilities extended Kalman filter is preferred[4]. In our system, IMU data and flow deck measurements are the only inputs to the Kalman filter, as illustrated in Figure 3.4. During our experiments, we observed certain limitations of the optical flow sensor and proposed a safe landing mechanism to safeguard the drone from crashes and collisions as discussed in the

[3] https://www.bitcraze.io/products/z-ranger-deck-v2/

[4] https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/

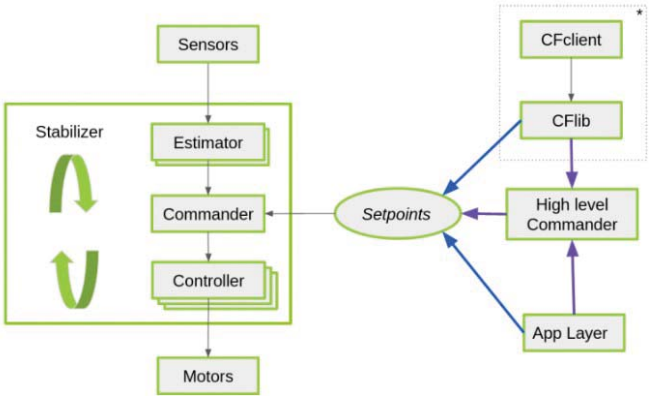


Fig. 3.3: General framework of the stabilization structure of the crazyflie with setpoint handling. * This part takes place on the computer through the CFlib for python, so there is also communication protocol in between. It is left out of this schematics for easier understanding(Reprinted from Bitcraze documentation)

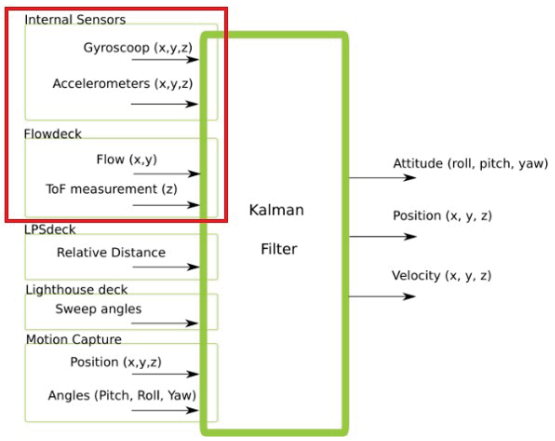


Fig. 3.4: Schematic overview of inputs and outputs of the Kalman Filter, the inputs of our system are highlighted in red(Adapted from Bitcraze Documentation)

Appendix 5E.

3.2.3 Commander Module

Incoming setpoints from many sources are handled by the commander module. The high-level commander module (purple pathway in Figure 3.3) or a Python script utilizing the cflib/cfclient or the app layer (blue paths in the Figure 3.3) can both directly create a setpoint. In turn, the Python library or the Crazyflie itself can be used to remotely control the High-level Commander.

There are several modes of providing setpoints to the drone. Position(X,Y,Z) and attitude (pitch, roll, yaw or in quaternions) are the two levels to control. These can be controlled in different modes, namely:

- Absolute mode (modeAbs)

- Velocity mode (modeVelocity)
- Disabled (modeDisable)

If setpoint.mode.xyz is set to modeAbs, the controller will follow the values provided by setpoint.position.xyz if absolute position control (move to point (1,0,1) in x,y,z) is requested. If setpoint.mode.xyz is set to modeVel, the controller will listen to the values provided in setpoint.velocity.xyz if we would prefer to control velocity (move 0.5 m/s in the x-direction). At that point, all of the attitude setpoint modes will be deactivated (modeDisabled). All position modes are set to modeDisabled if only the attitude has to be controlled. This occurs, for example, when we use a controller to operate the Crazyflie in attitude mode via the CFclient.

3.2.4 Controller Module

The controller module's task is to generate the required thrust to navigate from one point to another by controlling the motors. Although there are Incremental Nonlinear Dynamic Inversion (INDI) and Mellinger controllers available, Proportional-Integral-Derivative (PID) controller is the default setting in the Crazyflie firmware.

As illustrated in Figure 3.5a, the PID position controller will get the desired position set points from the high-level commander. The attitude PID controller receives the desired pitch and roll angles as a result. In the end, the angle rate controller converts the intended angle rates into PWM signals for the motors, independent of the control mode[5]. The proposed framework is illustrated in Figure 3.5b. It includes adding a layer above the high-level controller, which is the waypoint detection. This waypoint detection is controlled by either a human in the loop or preplanned based maps.

3.3 Replicating Vision based Obstacle Avoidance Algorithm - DroNet

As we explained in the Section 1.3, PULP-Dronet is considered as state of the art egocenric vision based model for nano-scale drones. Therefore, our first task was to replicate the PULP Dronet. To do this, we followed the pipeline described by the PULP researchers[6].

3.3.1 System Prerequisites

In order to reproduce PULP-Dronet-v3, we first made sure that all software versions were set up appropriately by following the guidelines in the official repositories. We set

[5] https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/

[6] https://github.com/pulp-platform/pulp-dronet/tree/master/tiny-pulp-dronet-v3

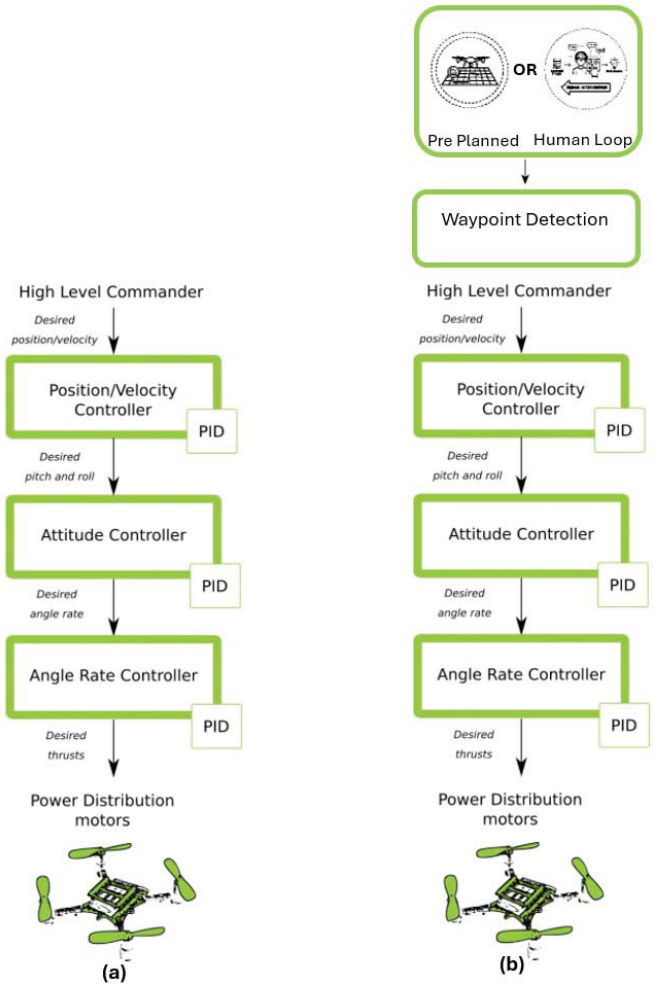


Fig. 3.5: (a)Pathway of the PID controller(Reprinted from Bitcraze Documentation) (b) Proposed controller framework(Adapted from Bitcraze Documentation)

up the development environment on an Ubuntu 20.04 virtual machine. The required components comprised the GAP GNU toolchain, GAP8 OpenOCD, and the major Software Development Kit (SDK), GAP-SDK. The GAP-SDK repository was cloned [7] and downgraded its version to 3.9.1. A special Conda environment was constructed for this task, using Python 3.6.10, and the SDK was built under these conditions.

As stated in the tiny-pulp-dronet-v3 repository[8], we next installed the GAP GNU toolchain and examined the correct branch. We also installed Gap8 OpenOCD and downgraded it to the version recommended in the documentation. After completing the configuration, we validated the installation by performing tests on the GVSOC simulation platform, which confirmed that the environment was successfully established.

3.3.2 Flashing and Testing

After finishing the initial setup, we flashed the Crazyflie firmware to STM32 flight controller and the PULP-Dronet model onto the GAP8 microcontroller. The files which needed for the compilation process were already provided by the PULP researchers using NEMO and DORY tools. The Olimex ARM-USB-TINY-H debugger was used to program the GAP8. For the Crazyflie, we initially used firmware version 2022.01, which was a downgraded release required for compatibility with the PULP-Dronet-v3 framework.

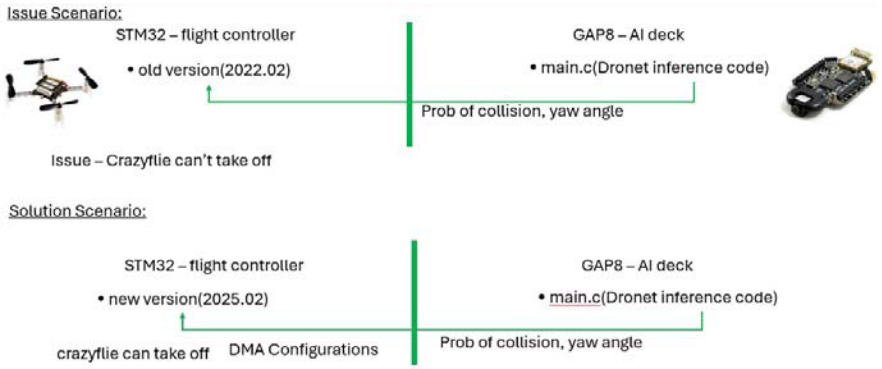


Fig. 3.6: Overview of the Problem scenario vs Solution scenario

Once flashing was complete, we tested the drone by providing commands using the Crazyflie client interface. However, during takeoff, the drone demonstrated instability and usually tended to crash.

The device has two primary firmware components, as shown in Figure 3.6. The GAP8 microcontroller on the AI deck powers the PULP-Dronet inference code, while

[7] https://github.com/GreenWaves-Technologies/gap_sdk

[8] https://github.com/pulp-platform/pulp-dronet/tree/master/tiny-pulp-dronet-v3

(a) (b) (c)

Fig. 3.7: Three representative examples of drone collisions during onboard execution of the PULP-DroNetV3 model

the Crazyflie flight controller (STM32) powers the Crazyflie firmware. UART is used for communication between the GAP8 and the STM32. The AI deck takes pictures from the onboard camera, processes them on the GAP8 cluster, and produces two values: the steering angle and the collision probability. The Crazyflie flight controller receives these outputs via UART and handles the incoming data using a UART interrupt handler. The drone's maneuvering is controlled by the flight controller, which reads these values after being triggered.

In our instance, however, the Crazyflie was unable to take off and hover steadily even though the STM32 flight controller correctly received the outputs. We determined that this issue was likely due to the earlier firmware version operating on the Crazyflie. We changed the Crazyflie firmware to a more recent and reliable version in order to fix the issue. As instructed by Crazyflie developers, we upgraded the Crazyflie firmware from version 2022.02 to 2025.02. In order to assure compatibility with the upgraded firmware, the control commands had to be reimplemented from scratch. This solved the take-off and hovering issue. After implementing this on the actual drone, we noticed that the drone was unable to avoid obstacles; while the prediction was accurate on the trained data, it had trouble making predictions in unseen situations. Some of the colliding instances were captured in Figure 3.7.

We downloaded the pretrained PULP-DronetV3 model and tested it in both a seen and an unseen environment to learn more about this behavior. Figure 3.8a illustrates that in all three cases within the observed environment, the model generated accurate predictions for both the steering angle and the probability of collision. However, in the unseen environment, its performance significantly declined. Specifically as illustrated in Figure 3.8b, DroNet predicts the probability of collision as 1 when there is no obstacles and predicting 0 as when there is an obstacles in front.

(a)

(b)

Fig. 3.8: Probability of collision and steering angle from (a) seen environments and (b) unseen environments.

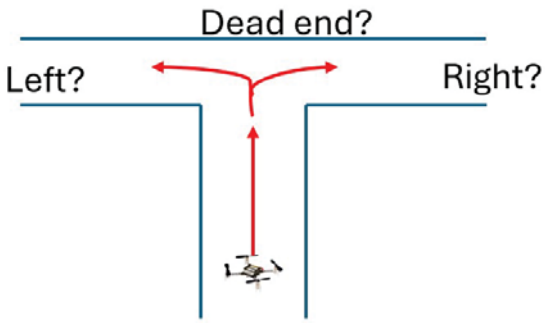


Fig. 3.9: DroNet fails in Decision points

3.3.3 Collision avoidance vs Intelligent Navigation

The control framework of the DroNet model is explained in Section 1.3.1. This closed-loop framework is effective but is limited for obstacle avoidance task only. This is because dronet's steering angle does not justify the requirement and can be chaotic in some locations like intersections, as illustrated in Figure 3.9. The system we propose is an intelligent navigation framework. This has two level of control mechanisms. (i) The detection of possible pathways(Waypoint detection) and (ii) Decision execution. The former mechanism includes identifying the orthogonal concepts as described in Figure 1.4f. The latter mechanism can be an open-loop control framework or pre-planned. The open-loop control framework is similar to the agency concept, where the AI model escalates the problem to a human to decide the correct pathway.

3.4 Intersection Identification Framework

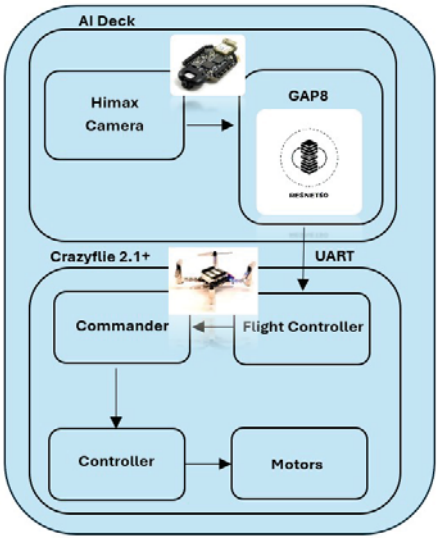


Fig. 3.10: On Board Processing

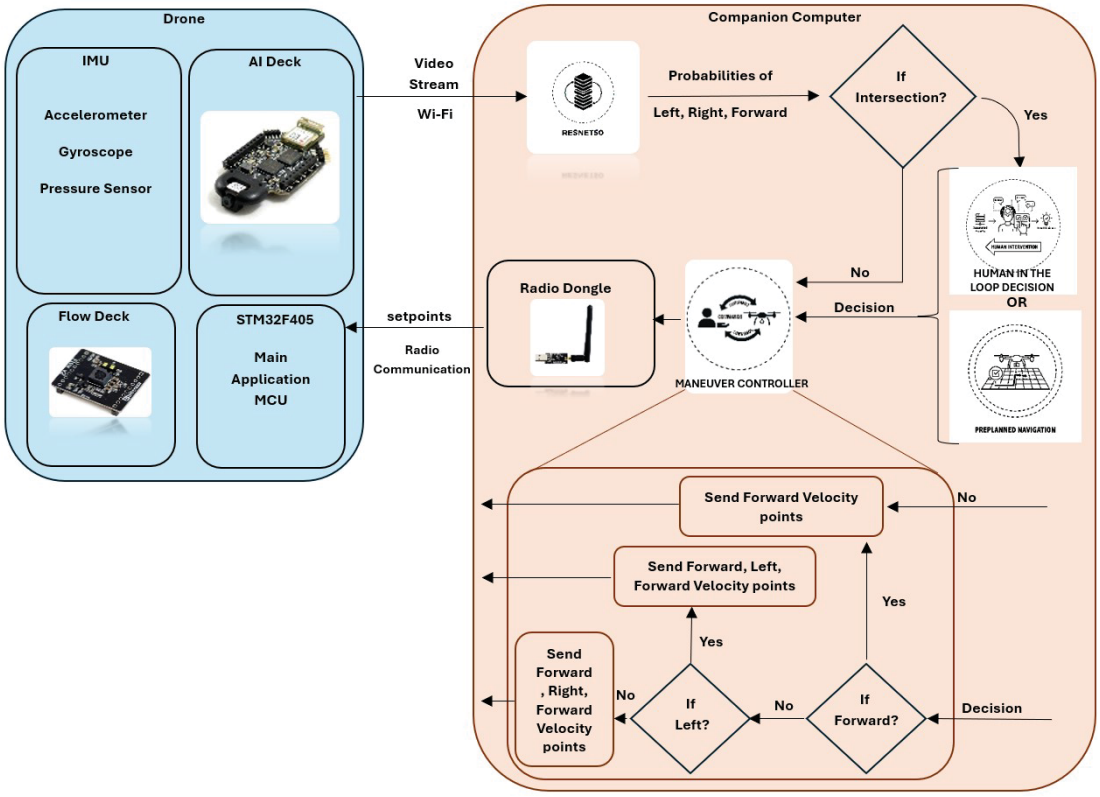


Fig. 3.11: Off Board Processing

Deployment with entirely on-board processing as shown in Fig. 3.10 in network-constrained environments is the ultimate goal of this effort. Off-board processing as

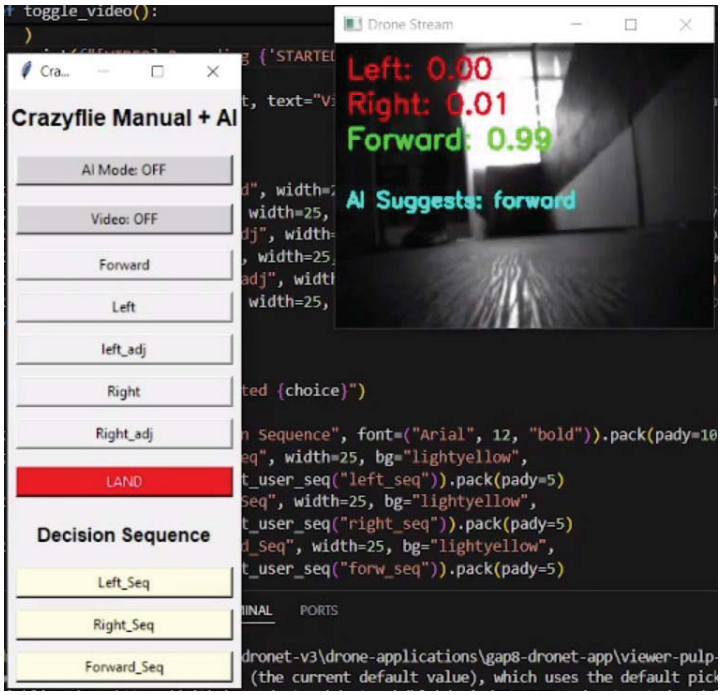


Fig. 3.12: GUI for human-in-the-loop decision-making during intersection-based navigation

shown in Fig. 3.11 is now used as a transitional stage since it allows for quick testing and adjustment of the maneuvering execution. In this method, the onboard camera's live video stream is sent to a companion computer, where it is processed and inference is generated, and the corresponding control commands are then sent back to the drone via radio communication. A ResNet50-based deep neural network is used off-board to interpret image frames transmitted from the AI-deck via a TCP/IP connection in order to predict navigation actions (left, right, or forward) with 10 frames per second. To deal with predicted noise and uncertainty, temporal smoothing and confidence thresholding are used. While ambiguous predictions(where 2 or more labels attain high) initiate a safe hovering behavior with optional human intervention(or preplanned map) via a graphical user interface as illustrated in Figure 3.12. Three decision sequences are available, which are forward, left or right. Depending on the decision, distinct predefined sequences of velocity setpoints are sent to the drone. For example, as illustrated in Figure 3.11, if the decision is left, it will trigger a set of velocity setpoints of going forward, turning 90 degrees left and forward velocity setpoints. During maneuver execution, if an obstacle is encountered within the field of view, the forward prediction will be zero, preventing forward motion and causing the drone to safely hover. The system then awaits the human command and can take manual control of the drone. When the AI mode is ON drone navigates in the forward direction in 0.08 meters per second until it reaches an intersection. If the drone encounters an intersection, it will work as per the mechanism described in Figure 3.11. These setpoints are transferred

to the flight controller of the drone for autonomous flight control. The companion computer uses a multi-threaded architecture that simultaneously manages visual perception, decision-making, flight control, real-time visualization and video recording, and manual override, enabling safe and interpretable autonomous operation.

The entire workflow of our suggested method is described in this section. First, we discuss the data collection framework. As detailed in Section 2.1.3, the availability of real-world intersection data is limited. Therefore, to address this limitation, a simulated environment was developed using Webots for data collection. As mentioned in Section 3.4.1 and 3.4.2, we first chose real-world environments and built digital twin supermarket environments. Section 3.4.3 then goes into detail about the data collection procedure used in both real-world and simulated settings. We discuss the two classification methods as outlined in Section 3.4.4 and Section 3.4.6. Lastly, Section 3.4.6 presents the experimental setups and model training and testing pipeline. Our implementation, experimental framework and supplementary materials are available on GitHub.[9]

3.4.1 Real-World Environmental Setup

We used AI-deck's monochrome camera to manually gather grayscale data at 5 frames per second in two supermarkets and a library to make sure the model learnt real-world features and textures. During the data collection process, a custom tool was created to easily label images into their appropriate class. To provide wide visual variation, we recorded a variety of supermarket and library aisle settings, such as different lighting, intersections, walls, and bookshelf layouts. We determined the middle waypoint for every intersection and chose a data point that was 1.15 meters before mid-waypoint. This distance guarantees that the drone can see the rack edges clearly inside its field of vision, allowing it to make the right, left or right judgments without running the risk of colliding. Nevertheless, this method of gathering data is tedious, entirely manual, and may damage the drone while it is in use. As a result, we attempted to create a tool that could capture videos from a mobile camera and then process them through a series of programs to produce a three-dimensional model. It should be possible to load this 3D-generated model into the Webots simulation platform. Thus, the process of gathering data is considerably simpler. Details are provided in Appendix C.2.

3.4.2 Digital Twin Environmental Setup

The digital twin environment offers complete control over parameters, repeated experimentation, and secure, collision-free data collection. This provides a solution for the real-world data limitation problem. It is possible to accurately change the robot's alti-

[9] https://github.com/yeko31/ego-intersect

tude, intersection distance, camera height, and field of view. These advantages are not achievable in real-world configurations. We focused on aisle structures that resemble supermarkets and libraries because of their neat 90° layouts and practical applicability. The AI-deck's grayscale camera and 87° field of view were matched to the virtual Crazyflie camera in Webots simulation platform, resulting in realistic pictures. To train our navigation model, we produced a variety of data pertaining to lighting, texturing, rack configurations, and shadows using these parameterized digital twins, as shown in Figure 3.13.

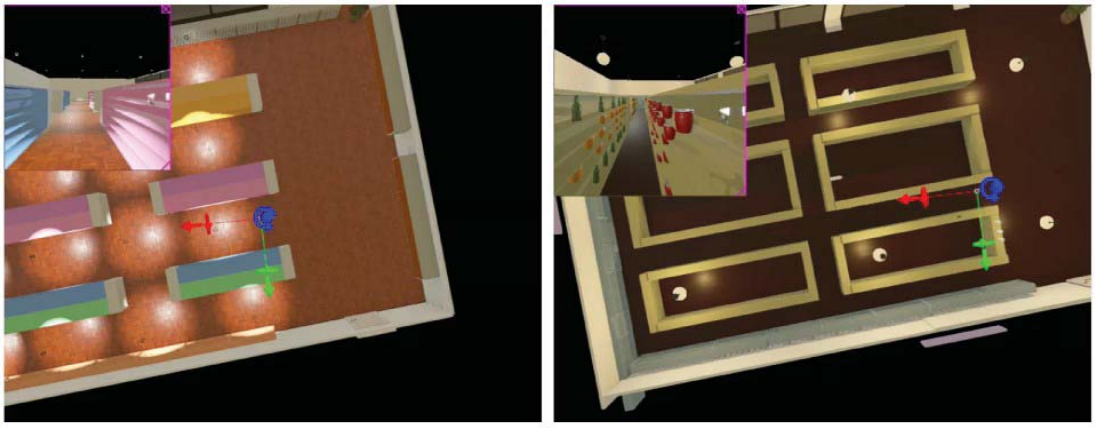


Fig. 3.13: Two different environment configurations in Webots

Webots controllers were designed to automate the egocentric data collection pipeline which is time and resource-consuming if done manually. Datapoints were defined as 1.15 m before the middle point of any intersection. The drone followed a predefined set of waypoints which were hard-coded into folders corresponding to each intersection class inside the simulated environment. At each datapoint, the drone performs a slight lateral wiggle to capture different views of the racks. In some waypoints, the drone performs yaw rotations to secure the egocentric view of the camera. Altogether, there were 22 way points covering 7 different intersection types, including forward pathways and bends. We only selected the simulated environments which has aisle widths of 1 m and 1.3 m to preserve the views of the racks before reaching an intersection point.

3.4.3 Data Collection

In both synthetic and real-world environments, the data was gathered at two altitude levels. Scenes were first classified with seven high-level classifications, such as left bends, left intersections, right bends, right intersections, T intersections, four-way intersections and straight class, before being broken down into a single three-bit directional format that indicated whether left, right, and forward pathways were available. Contrast and blur augmentations, as well as the creation of artificial bends to boost

TABLE 3.1: DATA DISTRIBUTION ACROSS TRAINING AND TESTING CONFIGURATIONS

Data Type	Number of Samples
Multi-class – Train (Real)	3929
Multi-label – Train (Synthetic)	2560
Multi-label – Test (Real)	100
Multi-label – Train (Real)	3929

Note. Summary of the number of samples used for multi-class and multi-label experiments across real and synthetic datasets.

samples, were used to rectify the imbalance in left and right intersections found in real-world data. In the end, 3929 training images, 100 real test images, and 2560 synthetic images were collected for the studies as described in Table 3.1. For the real-world dataset, images for each high-level class were collected and prepared in approximately equal quantities to maintain class balance. Data collection with the digital twin method is relatively simple because the configuration can be carried out iteratively. However, collecting data from the real world is still difficult. We use a syn-to-real generative model that converts synthetic images into realistic ones in order to better explore this problem. Additional details can be found in Appendix C.1.

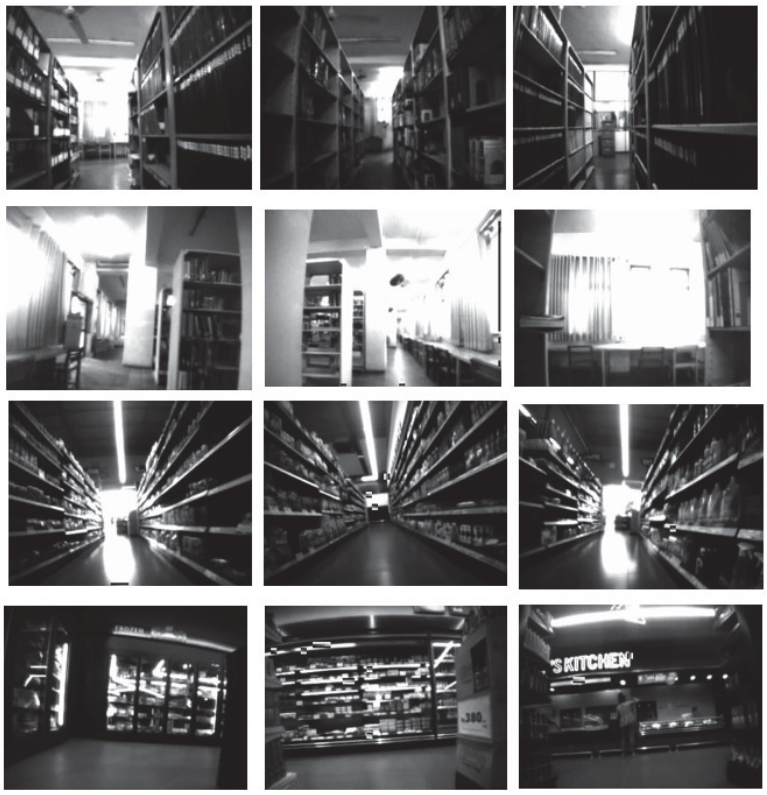


Fig. 3.14: Data collected from real-world scenes

TABLE 3.2: TRAINING CONFIGURATION FOR MULTI-CLASS AND MULTI-LABEL EXPERIMENTS

Parameter	Multi-Class	Multi-Label
K-Fold	5	5
Random State	42	42
Patience (Early Stop)	3	3
Minimum Delta (Early Stop)	0.0001	0.0001
Criterion	CrossEntropyLoss	BCEWithLogitsLoss
Optimizer	Adam	Adam
Max No of Epochs	30	30
Learning Rate	0.001	0.001
Batch Size	16	16
Non-linearity (Kaiming Config)	–	ReLU
Mode (Kaiming Config)	–	fan_in

Note. Summary of hyperparameter settings used for training multi-class and multi-label models.

3.4.4 AI model Creation Strategy A- Multi Class classification

We first framed the task as a multi-class classification problem, assigning each image to one of seven intersection types: straight, left bend, left intersection, right bend, right intersection, four-way, or T-junction. We followed this method at first because most of the related work addressed this challenge as a multi-class problem, as described in Table 2.1. For intersection detection, we tested both MobileNetV2 and ResNet50. ResNet50 was chosen for its robust image-classification capabilities[28]. Furthermore, learning left,right,forward navigational affordances as illustrated in Figure 1.4f is not a direct geometric feature learning. Consequently, a moderately deep network such as ResNet50 is more suitable than very deep detection architectures, such as YOLO. MobileNetV2 was picked for its lightweight architecture appropriate for resource-constrained drones. First, all grayscale images were duplicated across three channels to ensure compatibility with the pretrained architectures, as both networks were initially intended for three-channel RGB inputs. The final fully connected layers of the original models, which had been pre-trained on ImageNet, were swapped out for seven classification heads, which matched the number of output classes in our dataset. We then trained a ResNet50 model from scratch using the data collected in the synthetic world(Section 3.4.3). Next, training using real-world data was done using five fold cross validation with early stopping to observe overfitting. To initialize the weights and biases of these models, the synthetic weights which was trained in the step before were used. Training as a multi-class classification included total real data of 3929. Criterion was chosen as CrossEntropyLoss and Adam as an optimizer with

a learning rate of 0.001. We froze all the layers except for the final two layers and trained the network. Further details are available in Table 3.2.

3.4.5 Importance of learning orthogonal features

Multi class classification method has certain issues. For example, left intersection and left bend have a shared concept that is captured as unrelated in this setting. Therefore, learning two different feature sets in this case will not lead to the correct interpretation of the above intersection types because of the similarities in left features. This is further investigated in the Chapter 4. Another drawback is that the difficulty of categorizing a heterogeneous decision point into a single class, as shown in Figure 1.4e. Improving the robustness of this framework requires substantially more data; however, in real-world settings, such data are limited [38]. Therefore, it's important to use methods like few-shot and zero-shot learning. To extend, from limited data to complex junction types like staggered, which is illustrated in Figure 1.4e, learning the orthogonal concepts becomes essential, as these concepts enable scalable extensions to more complex intersection structures. Therefore, to solve the way point identification problem, a multi-label image classifier was developed, which shows the possible pathways the drone can take. As summarized in the Figure 1.4f, multi-hot binary label classification was used where each image could simultaneously include multiple directional indicators (left, forward, right). In this setting, we can still generalize to more complicated heterogeneous scenarios due to the same concepts being shared. We expect that the use of multi-label targets can further extend to previously unseen and more complex environments by decomposing decisions into orthogonal yet related concepts.

3.4.6 AI model Creation Strategy B- Multi Label classification

We modified MobileNetV2 and ResNet50 to take into account the requirements of the task and the characteristics of the dataset. The final fully connected layers of the original models, which had been pre-trained on ImageNet, were swapped out for three classification heads, which matched the number of output classes in our dataset. We applied values of 0.2 color jitter for brightness and contrast as augmentation. All grayscale images were duplicated across three channels to ensure compatibility with the pretrained architectures, as both networks were initially intended for three-channel RGB inputs.

We have listed the main phases of our multi-label class experiments in Figure 3.15. The reasons to carry out these experiments are represented in Table 3.3. We conducted five focused experiments to methodically assess the robustness and transferability of our intersection identification framework. By evaluating a model that was completely trained from scratch on both training and test data, Experiment A creates a baseline and reveals the task's inherent learnability. Motivated by the scarcity of real-world data,

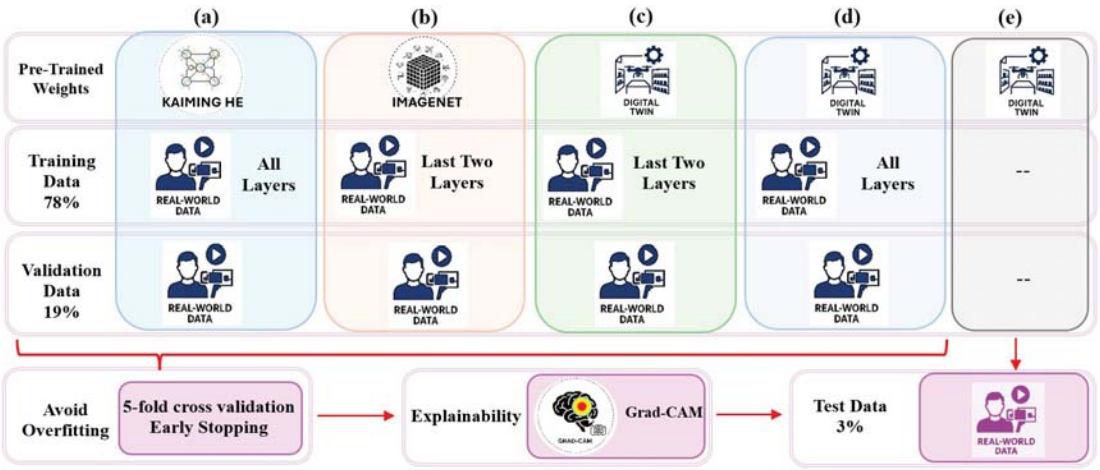


Fig. 3.15: Experimental design workflow of the proposed methodology. Five different experiments were conducted, varying in weight initialization and training procedure. 78% of data used for training, 19% used for validation, 3% used for testing (a)Kaiming He initialization (b)ImageNet initialization (c)Digital twin initialization trained on last two layers (d)Digital twin initialization trained on all layers(e)Digital twin initialization tested on real data, no training using real-world data.

TABLE 3.3: SUMMARY OF THE OBJECTIVES OF THE EXPERIMENTS

Experiment	**(a)**	**(b)**	**(c)**	**(d)**	**(e)**
Objective	Test on how training from scratch behave on train and test data	Test on how transfer learning behaves on train and test data (since the real data is limited)	Test how transferable the features from digital twin domain to real-world domain on train and test data(with training)	Test how transferable the features from digital twin domain to real-world domain on test data(with training all layers)	Test how transferable the features from digital twin domain to real-world domain on test data(without using real data for training)

Note. Summary of the motivation behind each experimental configuration (a)–(e).

Experiment B assesses transfer learning to see if pretrained features enhance learning effectiveness and generalization. In order to better understand domain adaptation during training, Experiment C examines how successfully fine-tuned digital-twin traits transfer to real-world data. This is further investigated in Experiment D, which measures the complete adaptability of synthetic-domain representations when exposed to

real-world examples by training all layers. Lastly, Experiment E highlights the model's capacity to generalize solely through simulation by testing the pure sim-to-real transferability of digital-twin attributes by assessing performance on actual data without any real-world training. When taken as a whole, these trials offer a thorough examination of domain transfer.

Artificial neural network weights can be initialized using different methods. In this research, we are mainly focusing on Kaiming He, ImageNet and synthetic weight initialization methods. The Kaiming initialization method, also referred to as Kaiming He initialization or He normal initialization, is one of the ways of doing it. The publication "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun presented this technique[45]. The vanishing or exploding gradient issue that might arise during deep neural network training is the driving force behind Kaiming initialization. Conventional weight initialization techniques like random normal or Xavier initialization may result in gradients that vanish or explode as they propagate through the layers during backpropagation in deep networks, particularly those that use rectified linear unit (ReLU) activation functions. In contrast, He initialization helps mitigate the vanishing and exploding gradient problems that can impede the training of deep networks.

"ImageNet weight initialization" is the process of initializing a neural network (usually a CNN) with weights that have been pretrained on the extensive ImageNet dataset. In this type of transfer learning, the model makes use of its past knowledge of basic structures and characteristics that it has acquired from the variety of ImageNet data.

In the digital twin initialization approach, we train all layers from scratch using the synthetic data collected from Webots simulation as described in Section 3.4.2. We train the whole network for 30 epochs with a learning rate of 0.001 and a batch size of 16. Then we save the weights and use them to initialize the network appropriately. The details are available in Table 3.2.

These experiments are done for both ResNet50 and MobileNetV2 architectures. Data distribution is explained under the Section 3.4.3. The intention of using cross-validation is to observe the overfitting issue. As shown in Figure 3.15, weight initialization was done in three main ways which are digital-twin, ImageNet and random. There are three stages in the experimental design framework. The first stage is the weight initialization phase. The second stage is the real-world training phase. In this stage, some models were fully trained, others were partially trained, and a few were not trained at all under the real-world five-fold cross-validation setting. We set the criterion for BCEwithLogitsLoss and set the optimizer as Adam. Next, Grad-CAM was used to observe the significant areas of the input image which were used to predict the outcome. We used the model which achieved the highest validation accuracy across the folds for this and chose the layer before the final layer as the target layer. This allows

the model to confirm that the model ignores background noise and instead pays attention to semantically significant areas like stock shelves, aisle borders, or intersection cues.

Testing is the final stage where we use different metrics to observe the genralizability. To check on this, we chose four metrics which are Hamming loss, precision, recall and F1 score, using the libraries from Scikit Learn as explained in the Section 2.2.

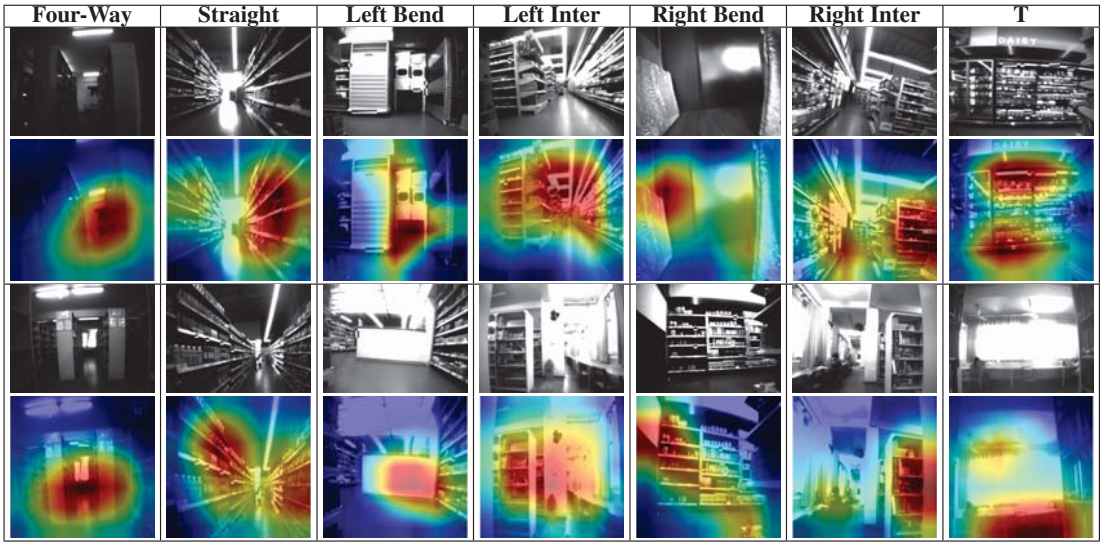
CHAPTER 4

RESULTS AND DISCUSSION

In this chapter, we present the results of multi-class classification in Section 4.1, and the improved version which is the multi-label classification, in Section 4.2. The results include Grad-CAM visualization and as well as training, validation losses and accuracies.

4.1 Multi Class Classification

TABLE 4.1: GRAD-CAM VISUALIZATION RESULTS FOR DIFFERENT INTERSECTION TYPES FOR MULTI-CLASS CONFIGURATION (TWO SAMPLES FROM DIFFERENT LOCATIONS FOR EACH CLASS)



Note. Original input images are shown alongside their corresponding Grad-CAM visualizations.

Figure 4.1 represents the training and validation curves. Table 4.1 shows that ResNet50 repeatedly misses salient visual clues needed to differentiate between various junction types. The areas that are highlighted don't provide enough information to be reliably interpreted. In T-junction scenarios, for example, the network mostly concentrates on the top and bottom portions of the image instead of the structural junction features. Similar to this, attention is skewed toward right-edge features in left-bend circumstances, but the opposite behavior is seen in right-bend events.

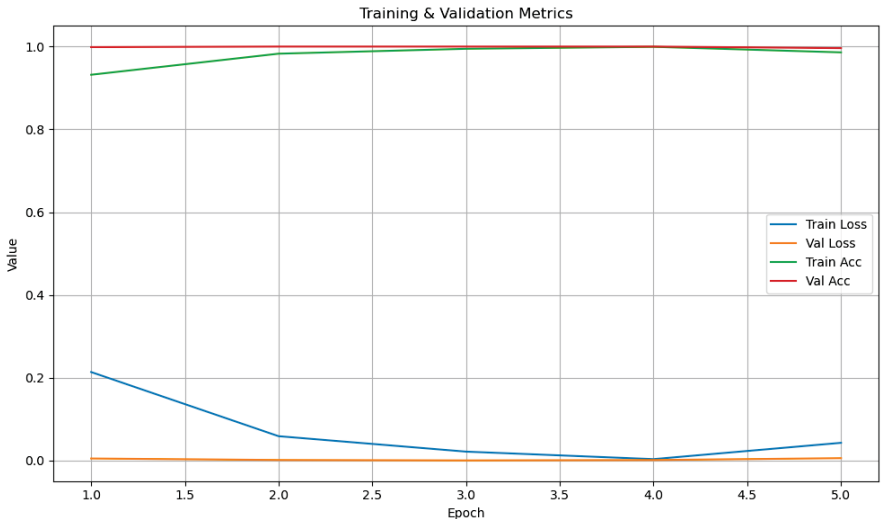


Fig. 4.1: Training graphs of multi class configuration

TABLE 4.2: K-FOLD PERFORMANCE COMPARISON ACROSS EXPERIMENTS A–D

	Kaiming He		**ImageNet**		**Digital Twin (Last 2)**		**Digital Twin (All)**	
Fold	**Best Val. Acc.**	**Epochs**	**Best Val. Acc.**	**Epochs**	**Best Val. Acc.**	**Epochs**	**Best Val. Acc.**	**Epochs**
1	0.9898	10	1.0000	6	0.9949	10	1.0000	15
2	0.9707	7	1.0000	4	0.9962	11	0.9987	16
3	0.9975	13	1.0000	4	0.9822	8	0.9796	8
4	0.9835	11	1.0000	6	0.9924	11	0.9707	8
5	0.9567	7	1.0000	5	0.9924	12	0.9949	12

Note. Comparison of validation performance across five folds under different weight initialization and digital twin training strategies.

4.2 Multi Label Classification

4.2.1 Results of ResNet50

In multi-label configuration, we report the best validation accuracies in each fold and the maximum number of epochs until early stopping in the Table 4.2. Here, outcomes of He initiation, training from scratch requires more epochs and exhibit slight instability in between folds. ImageNet weight initialization demonstrated stronger and faster convergence compared to He and Digital Twin initialization methods. ImageNet-initialized models regularly achieve perfect validation accuracy (1.0) in very few epochs across all folds. The accuracy of the digital twin configuration trained on the last two layers, remains very high in most folds. A good trade-off between speed and accuracy is achieved by fine-tuning only the classifier (final layers), which maintains pretrained knowledge and adapts effectively. We utilized the highest validation accuracy model across the folds and plotted the training accuracy and loss, validation accuracy and loss as illustrated in Figures 4.2, 4.3, 4.4, 4.5.

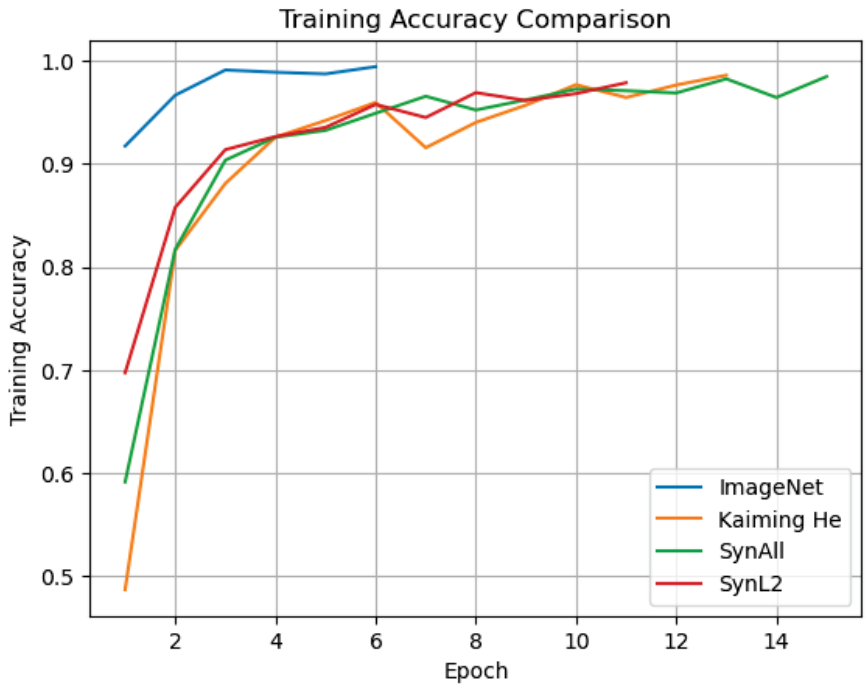


Fig. 4.2: Training Accuracy across different weight initialization methods

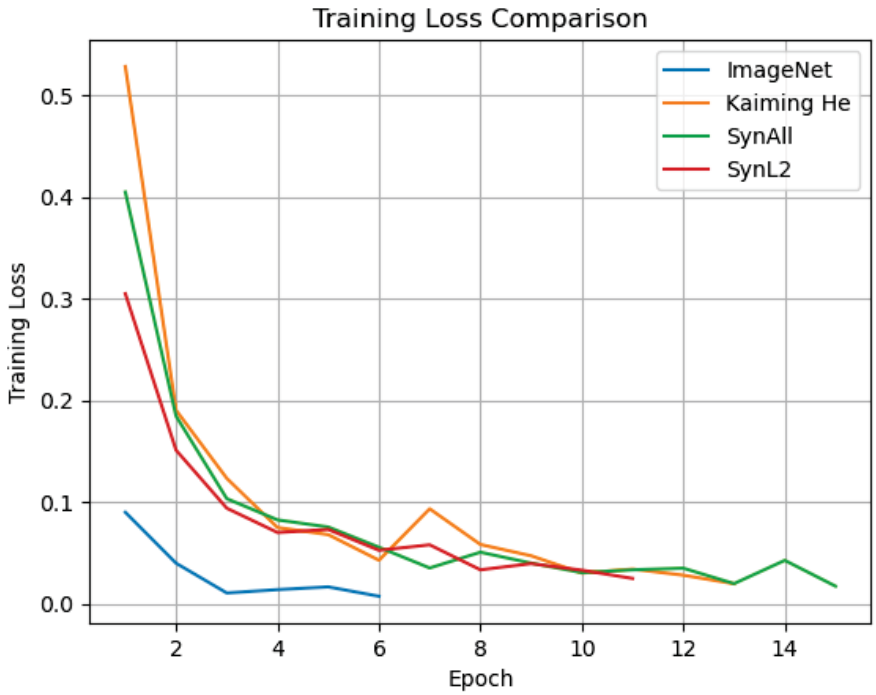


Fig. 4.3: Training Loss across different weight initialization methods

Grad-CAM was utilized to confirm if the model recognized relevant concepts instead of random features. In the graphics, the four experimental sets displayed different activation patterns. We observed that the Grad-CAM experimental configurations produce distinct activation patterns across configurations (a), (b), (c), and (d) in

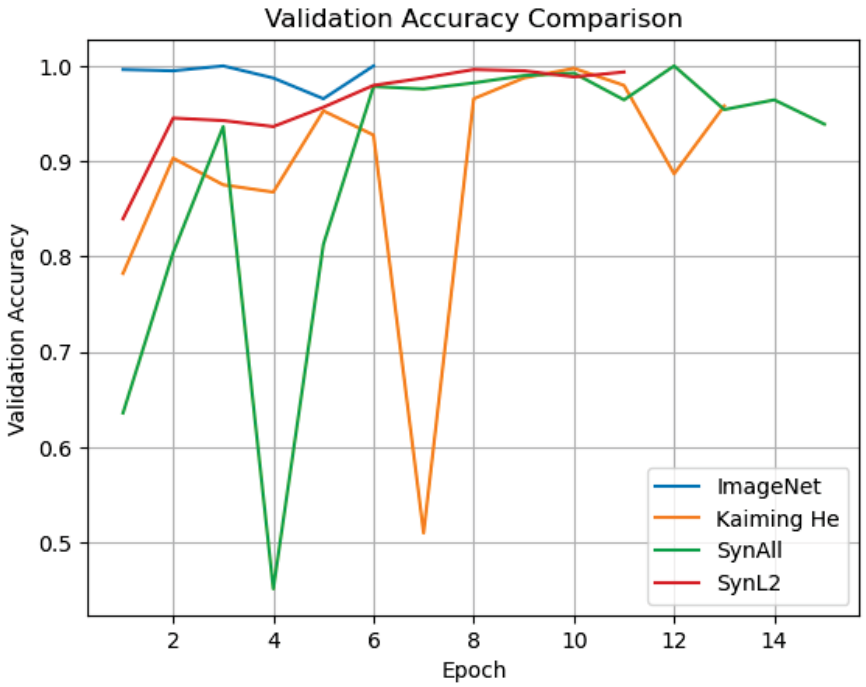


Fig. 4.4: Validation Accuracy across different weight initialization methods

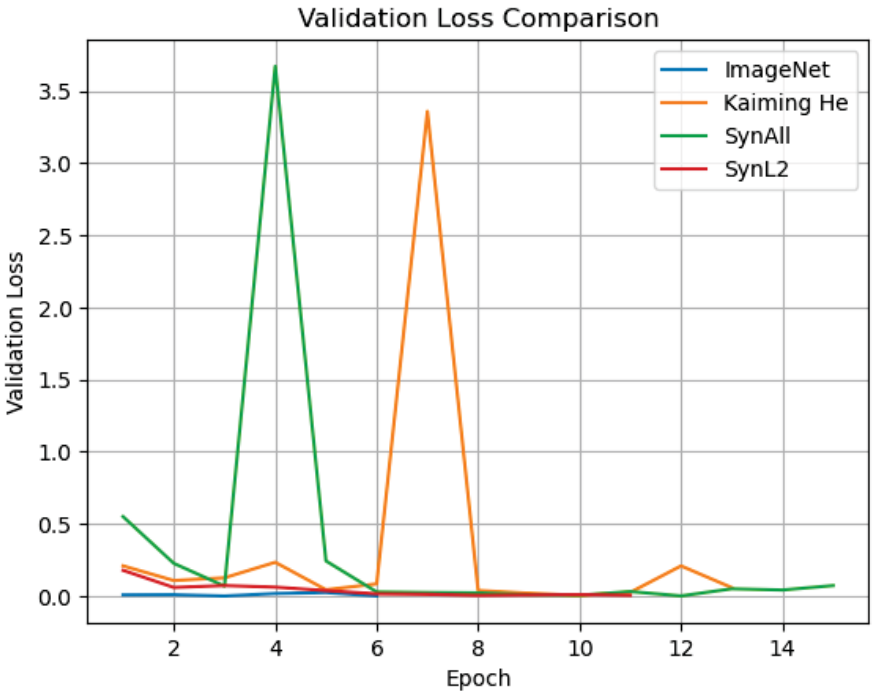
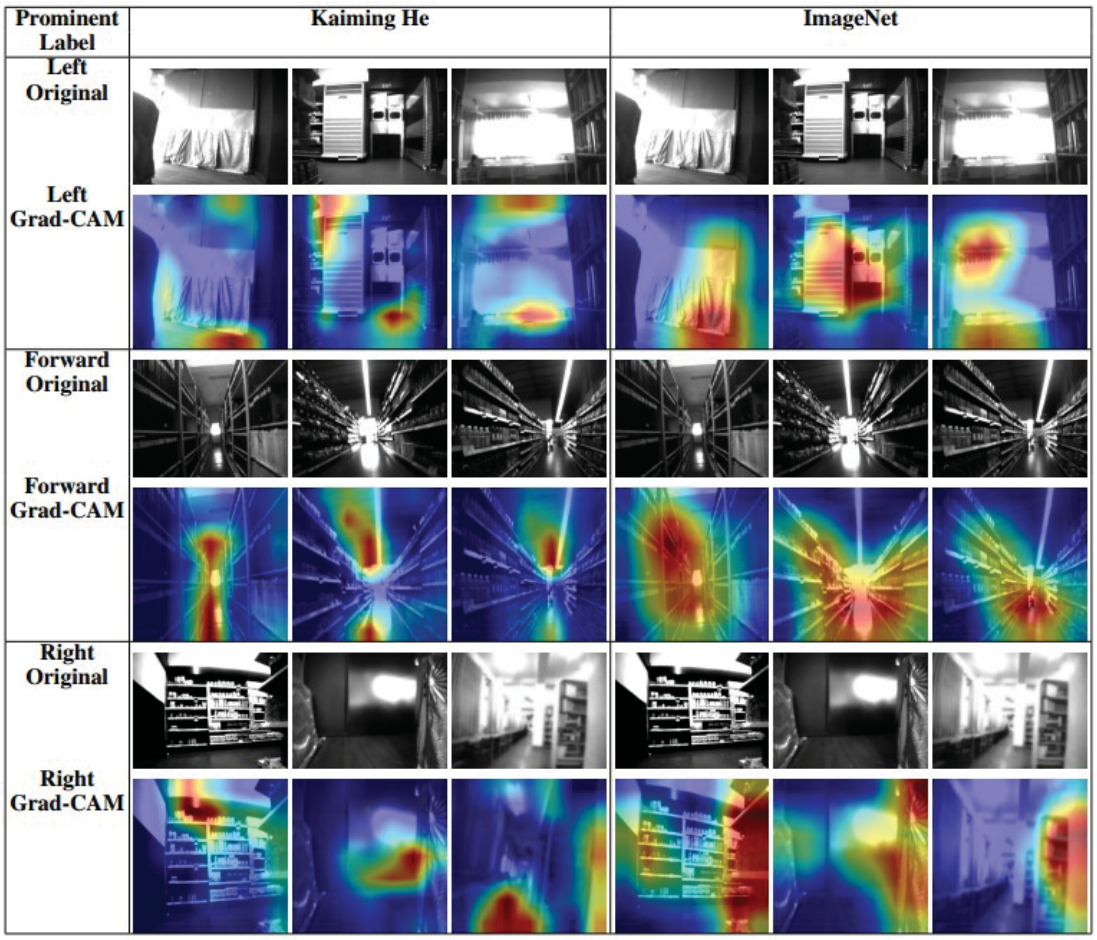


Fig. 4.5: Validation loss across different weight initialization methods

Figure 3.15, as evident from the visualizations presented in Table 4.3. Grad-CAM heatmaps were produced in relation to the most prominent label. This means the label with the highest projected probability is displayed in the tables. To demonstrate spatial consistency and variance, samples from various places are displayed for

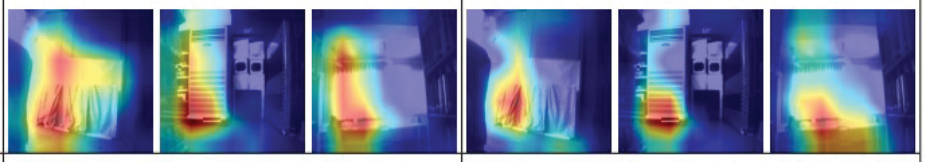
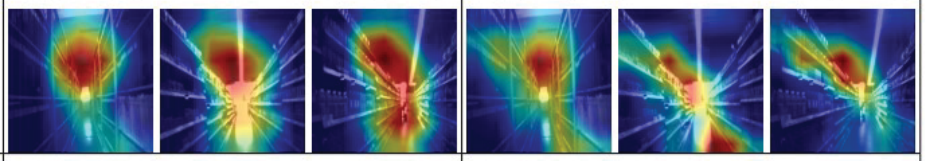
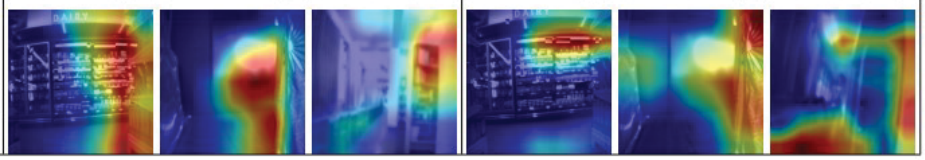
TABLE 4.3: GRAD-CAM VISUALIZATION RESULTS FOR MULTI-LABEL CONFIGURATION ON THE HE AND IMAGENET SETTINGS USING THREE DIFFERENT SAMPLES (RESNET50)



Note. Distribution of samples across different classes used in the study.

each label. Grad-CAM maps under configuration Kaiming He look weaker for feature localization. It's possible that early layers didn't develop strong spatial hierarchies, which made the heatmaps less interpretable. Compact and consistent activation zones are seen in configuration ImageNet, indicating more stable representations compared to configuration Kaimimg He. Explainability findings show that transfer learning increases both accuracy and visual interpretability. However, it is visible that the configuration ImageNet is not able to correctly identify the left visual cues in the Table 4.3. Digital twin-based fine-tuning produces activation maps which are more closely aligned with the pertinent scene elements as illustrated in the final two columns of Table 4.4. The digital twin last two configuration performed better than the other configurations in all four experiments, resulting in Grad-CAM heatmaps that were sharper and more domain-focused. This configuration shows enhanced interpretability and task awareness since the attention regions corresponding to the left, right, and

TABLE 4.4: GRAD-CAM VISUALIZATION RESULTS FOR MULTI-LABEL CONFIGURATION ON DIGITAL TWIN SETTINGS USING THREE DIFFERENT SAMPLES (RESNET50)

Prominent Label	Digital Twin(Last 2)	Digital Twin(All)
Left Original		
Left Grad-CAM		
Forward Original		
Forward Grad-CAM		
Right Original		
Right Grad-CAM		

Note. Distribution of samples across different classes used in the study.

forward labels are well aligned with the key spatial locations.

We chose the top-performing model in explainability, the digital twin configuration trained on the last two layers, to look into this behavior in more detail. We examined several kinds of intersections and bends, as shown in Figure 4.6, and created heatmaps that matched their labels. The left, right, and forward heatmaps were then overlapped to generate composite visualizations. The predicted probabilities of the corresponding labels are shown by the color bar beneath each image. Interestingly, in both the T-junction and four-way intersection scenarios, the T-intersection mostly activates the left and right regions, whereas the four-way situation also displays activation (red regions) in the central area. In the same manner, the overlapped heatmaps for left and right bends only show the pertinent edge regions, illustrating the model's capacity to direct attention based on directional context.

Grad-CAM was applied to consecutive image frames taken during supermarket

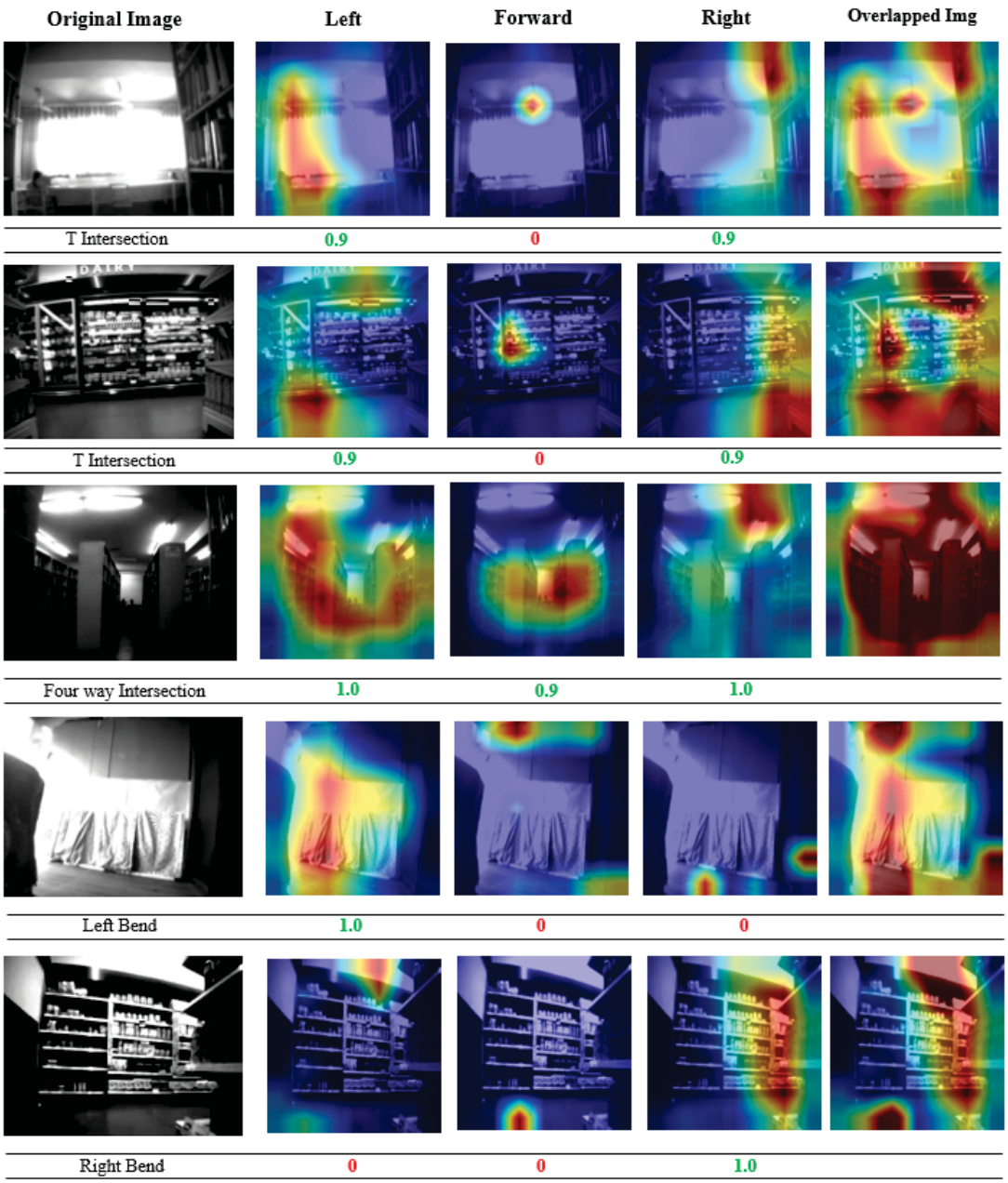


Fig. 4.6: Successful decomposition of Left-Forward-Right mutually exclusive (i.e. orthogonal) conceptual understanding by the digital twin (2-layers) model depicted using the Grad-Cam heatmaps across different junctions. Predicted probabilities are indicated below each example(ResNet50).

navigation in order to further examine the consistency and robustness of the Grad-CAM heatmaps, as illustrated in Figure 4.7 along with the prominent labels for the digital twin(last 2) configuration. Figure 4.7a represents a right turn at a right bend. The image sequence shows that there is more attention along the right edge in the first two frames; this edge-focused attention eventually fades once the turn is completed. A front-facing image of a rack is shown in Fig. 4.7b, where right-edge activations appear

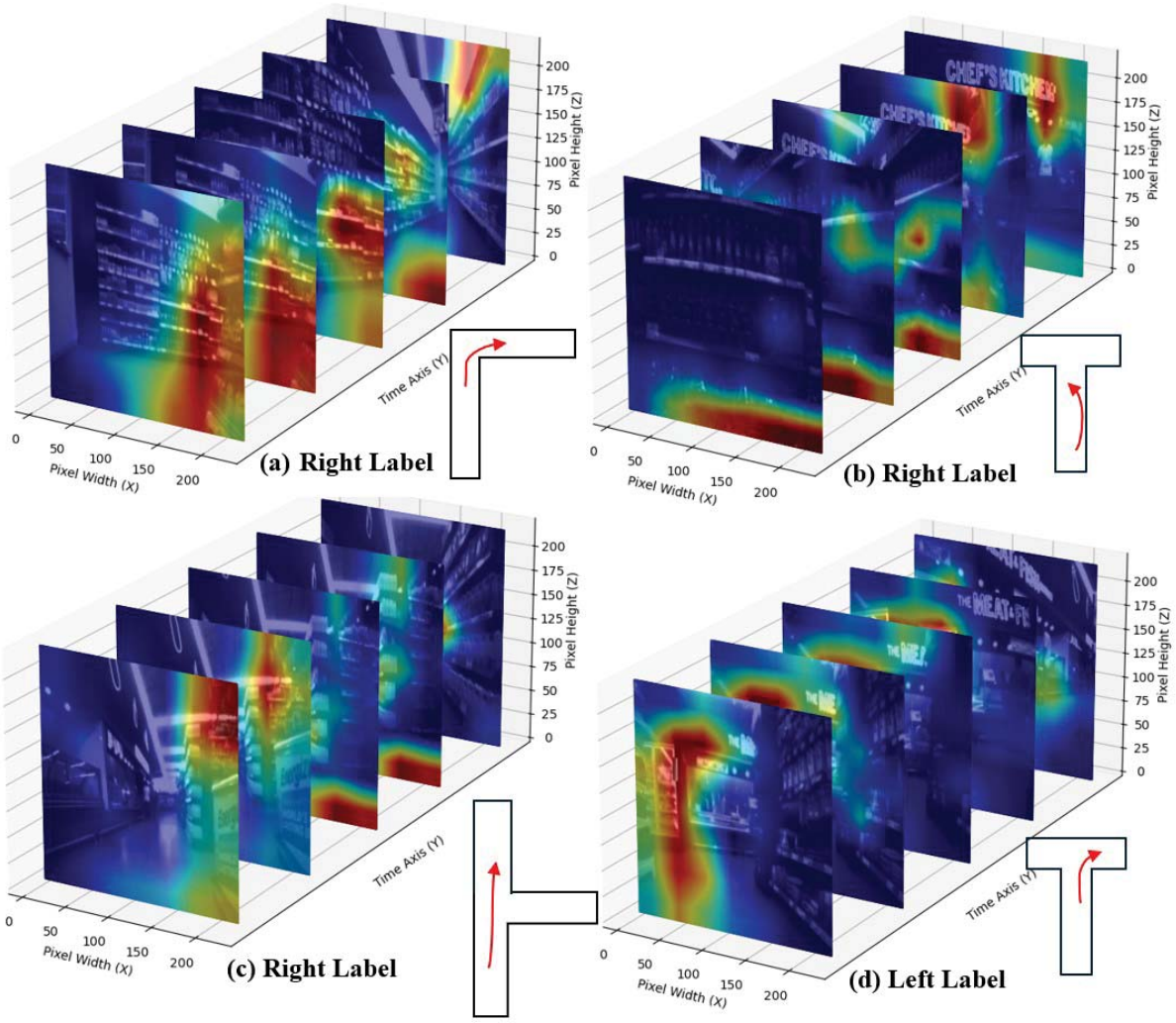


Fig. 4.7: Sequentially generated Grad-CAM heatmaps illustrating label-wise consistency across a navigation sequence. (a) Right-label activation at a right bend; after executing the turn, right-edge features are no longer highlighted. (b) Right-label activation and Re-emergence of right-edge activations as the platform approaches a T-intersection. (c) Right-label activation from right intersection to forward path. (d) Left-label activation indicating leftward navigability at T-intersections

as the platform gets closer to a T-intersection. As the platform moves from a right junction, and continues onward navigation, Fig. 4.7c illustrates that right attention disappears when the intersection is passed. A T intersection is shown in Fig. 4.7d, where left-focused activations gradually decrease over time, after taking a right turn at the T-intersection. This is further analyzed using the masking concept for explainability debugging in Appendix 2.

Figure 4.8 shows the comparison results on the test data. With the lowest Hamming Loss (0.08) and the highest F1 Score (0.92) of any configuration, the ImageNet weight initialization approach showed the best generalization performance. Our synthetic-

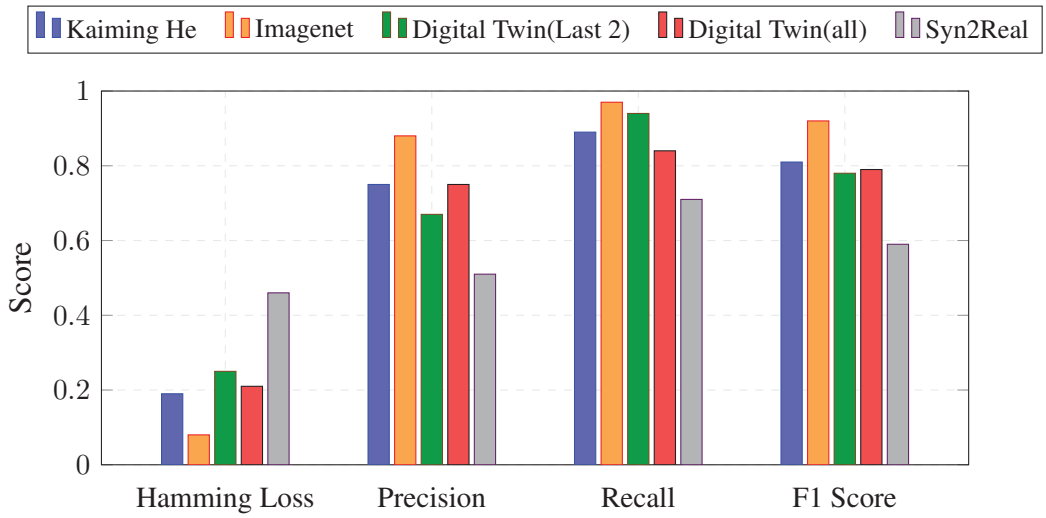


Fig. 4.8: Comparison of five experiments across Hamming Loss, Precision, Recall, and F1 Score on test data(Resnet50).

to-real configuration shows that digital twin data can reasonably generalize to real-world images with a F1 score of 0.6. The performance can be further improved by incorporating real-world data, highlighting the importance of developing a digital twin when real datasets are limited.

To evaluate our model on a publicly available dataset, we chose the EgoCart dataset[46], which is a benchmark dataset for Large-Scale Indoor Image-Based Localization in Retail Stores. We randomly selected 150 RGB images covering various types of intersections such as left,right,t, four-way and different bends and based on visual inspection, we categorized them into respective classes. The original images were RGB with a resolution of 1280×720 pixels. They were converted to grayscale and rescaled to a resolution of 324×244 pixels. The Imagenet pre-trained model was chosen for this evaluation because it demonstrated the best generalization capability among other models in Figure 4.8, enabling assessment on previously unseen data.

TABLE 4.5: IMAGENET MODEL ON OUR TEST DATA VS EGOCART DATA

Metric	**Our Test Data**	**EgoCart Data**
Hamming Loss	0.08	0.15
Precision	0.88	0.89
Recall	0.97	0.83
F1-score	0.92	0.86

We observe that ImageNet model was performing well on publicly available datasets, achieving an F1-score of 0.86, which represents a slight reduction compared to our test data. This discrepancy may be attributed to differences in image characteristics, as our test data was captured from the AI deck's Himax camera while the egocart dataset was

taken from a high quality camera.

4.2.2 Results of MobileNetV2

We follow the same set of experiments for MobileNetV2. This is a mirror analysis of Resnet50 and the analysis of the MobilentV2 is detailed in the Appendix A1.

4.3 Explanation of Grad-CAM results

It is evident that the left, right, and forward navigability directions have now been learned by the digital twin model trained on the final two layers. However, the question we now wish to investigate is whether the model has actually learned the correct features of an intersection. We selected two significant studies from the literature to investigate this in greater detail. One study is from the cognitive science aspect [5] and the other is from the Grad-CAM interpretability [6].

In order to ascertain how the brain automatically recognizes pathways for movement in the local environment, Bonner et al.'s work[5] explores the neural coding of navigational affordances in the human visual system. The characteristics of a visual scene that define possible movements and navigational activities are known as navigational affordances. The main idea is that even when a subject is not actively planning a route, the navigational affordances of a scene, where one can and cannot go, should be automatically encoded during scene perception if perceptual systems are optimized to extract features that support potential actions. The researchers have conducted two experiments in both artificially rendered environments and one with complex natural scenes by studying the different areas of the human visual cortex and their ability to encode navigational affordances. They have examined the Retrosplenial Complex (RSC) , Occipital Place Area (OPA) , Parahippocampal Place Area (PPA) , and Early Visual Cortex (EVC) . These experiments include the reconstruction of navigational affordance maps from the Functional Magnetic Resonance Imaging (fMRI) . Figure 4.9A illustrates the procedure that they followed to generate the pixelwise predictions projected to each image. Higher intensity (or "heat") ratings on these maps usually indicate places where movement is feasible.

OPA was the study's main focus and produced the best results among these locations. The strongest results are evident through two experiments, which are the Representational Dissimilarity Matrix (RDM) and Reconstruction analysis. By figuring out whether the activation patterns might be utilized to forecast the navigational heat maps of new scenes, the Reconstruction Analysis assessed the information's richness. OPA reconstruction analysis is the reconstruction that most closely matches the ground truth, as seen in Figure 4.9B. It is evident that OPA encodes fine-grained navigational affordances, which include the angular directions one can move in a scene.

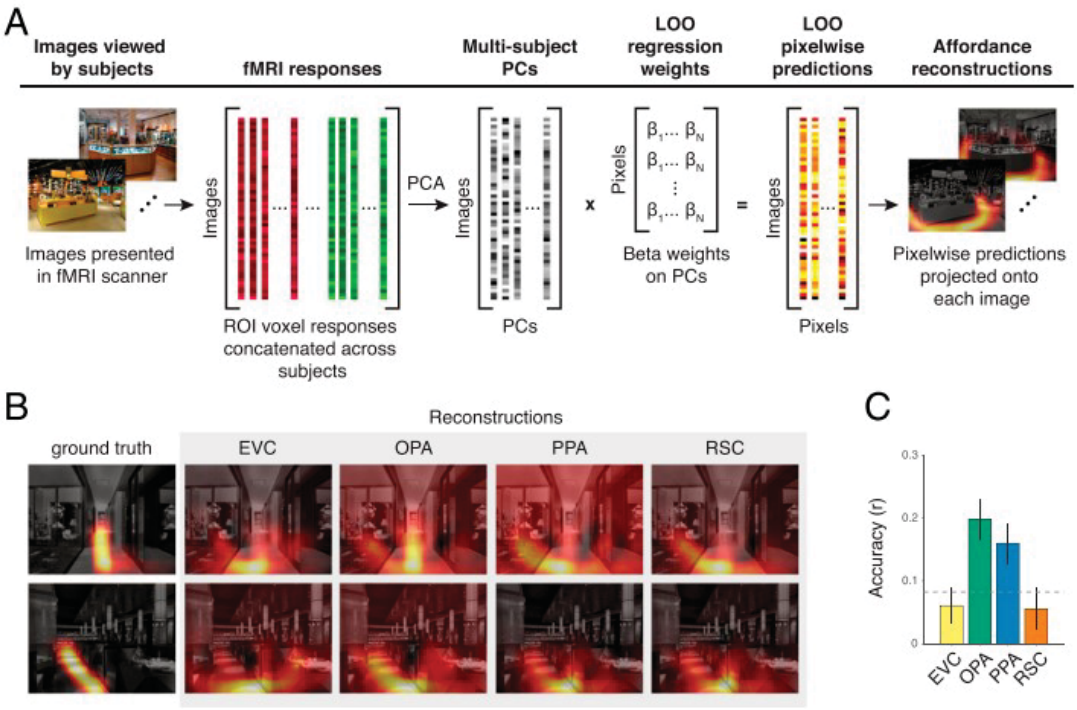
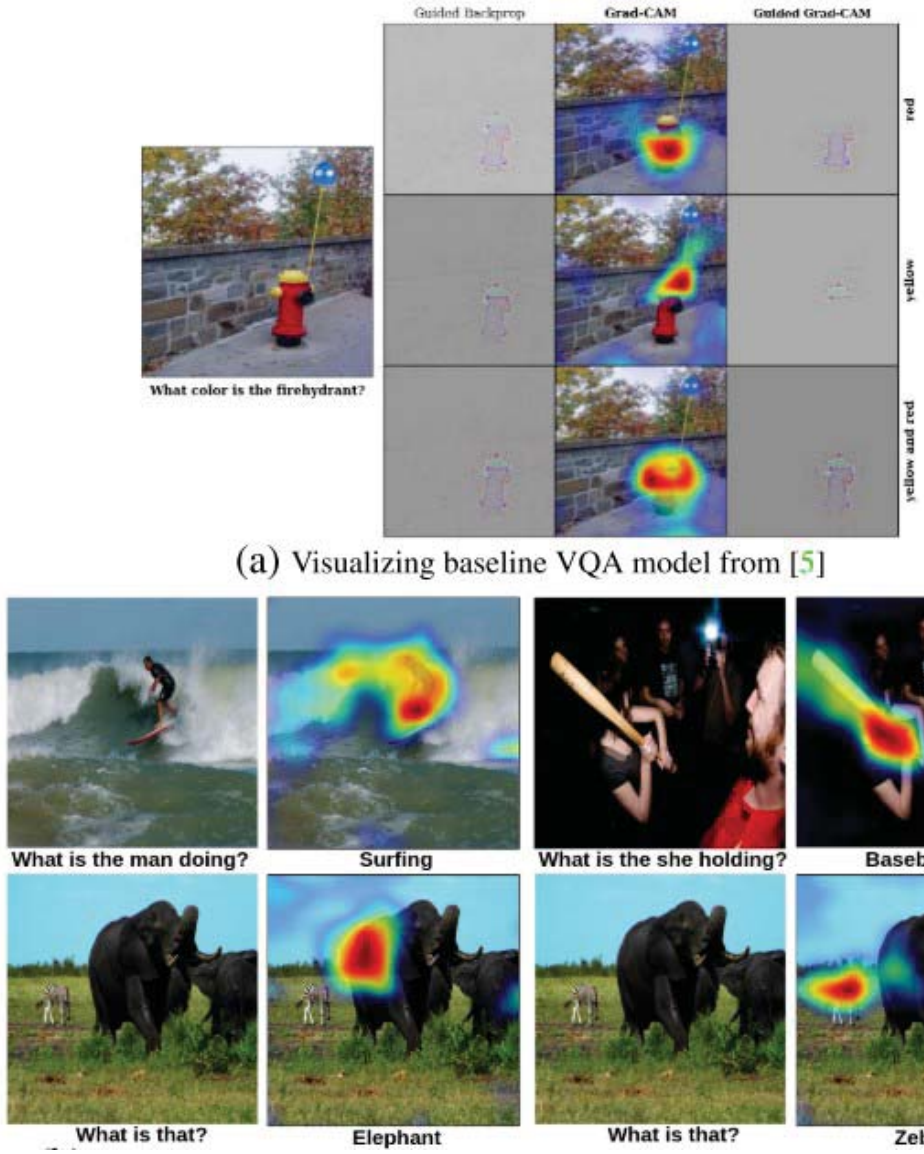


Fig. 4.9: Reconstruction of navigational-affordance maps reprinted from [5] with permission

The original authors who introduced Grad-CAM, conducted a deeper study [6] to help humans establish trust in the generated image by the Grad-CAM. To build this trust, authors conduct Visual Question Answering(VQA)[47]. In the AI field of Visual Question Answering (VQA) , a system responds to open-ended questions about an image by combining computer vision to "see" the image with natural language processing to comprehend the question and produce a human-like response. Such questions require deep visual understanding and reasoning, such as "What color is the fire hydrant?" or "What animal is in the picture?"[1]. Through VQA, human users gain confidence in the model's predictions. By seeing why the model made a certain prediction (i.e., by localizing the relevant evidence in the image), users can better trust the system. This is illustrated in the paper [47], as shown in Figure 4.10. It is visible that in Figure 4.10a, the attention of the network varies with the color patterns. This shows that, the network focuses on the most relevant pixel values corresponding to that class.

As illustrated in the Figure 4.6, we can observe that left, right and forward navigability visual cues have been identified by the network. The Grad-CAM heatmap for the "left" output highlights exactly those pixels most influencing the left-turn prediction. Similarly the "right" heatmap reflects right-turn cues, and "forward" reflects center-path cues. The overlap represents the general understanding of the intersection. Because convolutional layers preserve spatial layout, features on the left side of the

[1] https://github.com/ramprs/grad-cam/



(a) Visualizing baseline VQA model from [5]

Fig. 4.10: VQA Visualizations reprinted from[6]. Copyright © 2017, IEEE

image mainly influence the left-turn neuron. Grad-CAM backpropagates the gradient of the left output through the last convolutional maps, producing a heatmap over the input that shows where in the image the network "looked" to decide "left". This is close to what is illustrated in Figure 4.10. We provide an ablation study to further verify our study in Appendix B 2. From a cognitive science perspective, Humans similarly perceive scenes in terms of possible paths or "navigational affordances" as described above. Vision science research shows that our brain automatically encodes where we can go in a scene. For instance, the occipital place area (OPA) in the human visual cortex represents possible movement routes through a scene, invariant to low-level changes. We observe a notable resemblance between the navigational affordances represented in the human brain's occipital place area (OPA) and the Grad-CAM

visualizations shown in Figure 4.6, suggesting a similarity in the encoded navigational representations. In particular, this correspondence is evident in the emphasis on navigable open regions: the Grad-CAM maps focus on free space along navigable directions, mirroring the affordance structures reconstructed from neural representations in the human brain. Furthermore, we observe a resemblance between the right grad-cam visualizations of the digital twin last two method in Table 4.4 and the way how human perceive an intersection which is represented by the annotation in Figure 4.11 from the paper[7].

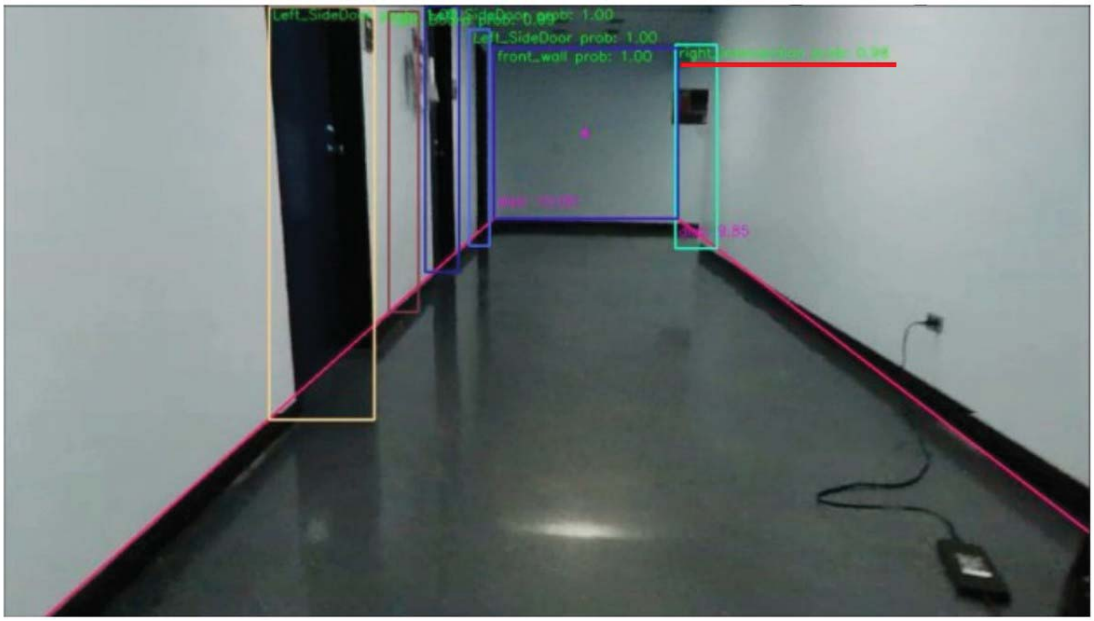


Fig. 4.11: Example of how human annotates intersections, adapted from the [7]. Copyright © 2019, IEEE. The green box highlights the human perception of a right-bend intersection(red underline).

In conclusion, the Grad-CAM heatmaps are explained by both deep learning and cognitive science aspects. We conclude that the Grad-CAM visualization is not random, it correctly identifies the left/right boundaries for side-paths and the central aisle for forward, which are precisely the critical cues for deciding possible directions.

4.4 Inference results

As the final set of experiments, we evaluated our model in different environmental conditions. Figure 4.12 illustrates the predicted probability values in different intersection types. Probabilities greater than 0.6 are marked in green, while values below 0.6, marked as red. The variation of probabilities over the complete video stream is available at the following link[2].

[2] https://drive.google.com/file/d/1lEl8f97Kkx-0TZptM8fY_96pLP2oxexn/view?usp=sharing

We also evaluated the memory consumption associated with model quantization in Appendix 4D. Table 4.6 shows that the model deployed onboard on the GAP8 microcontroller has a parameter size of 1.28 MB. Additionally, as reported in Table D.2, we achieved a parameter size of 2.6 MB for the quantized MobileNetV2 model with almost the same accuracy on test data in Table3.1.

TABLE 4.6: MODEL COMPLEXITY AND RESOURCE COMPARISON

Parameter	**ResNet50 (Ours)**	**MobileNetV2 (Ours)**	**PULP-DroNetv3 (Quantized)**
FLOPs	4,109,470,720	312,917,056	50,445,696
Parameter size (MB)	94.06	8.91	1.28
Forward/backward pass size (MB)	4173.73	1709.60	136.48

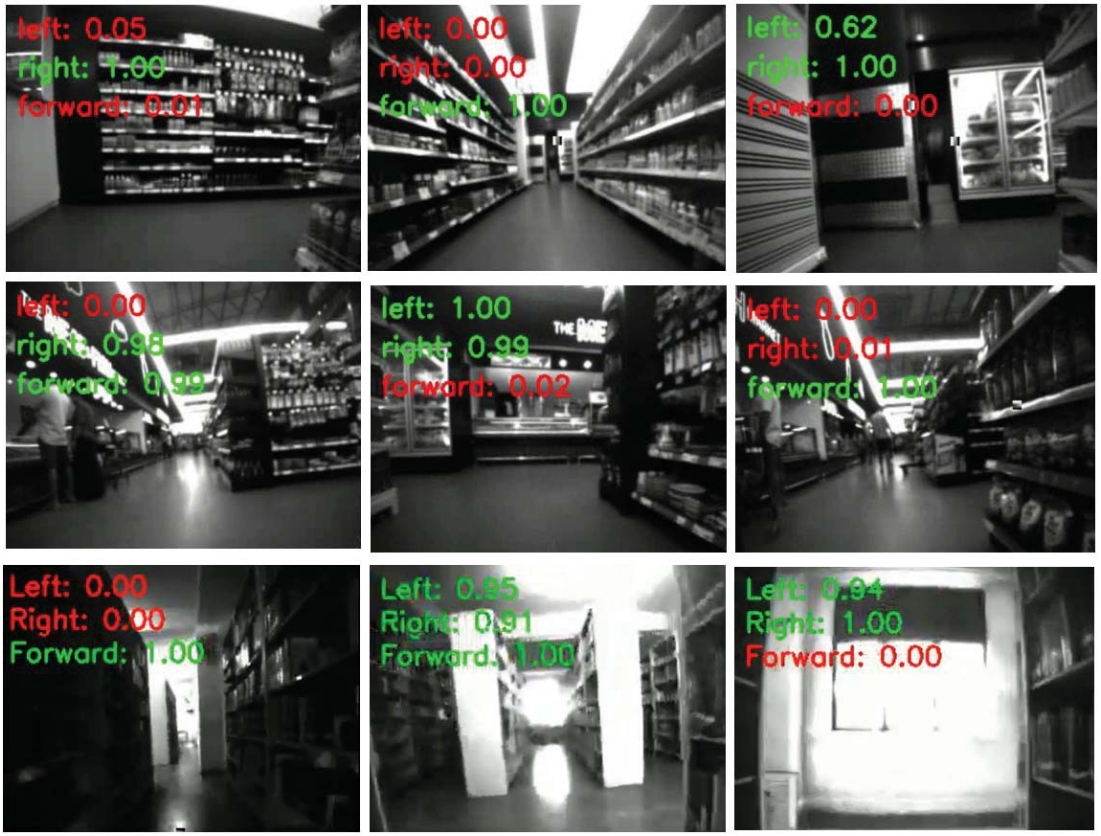


Fig. 4.12: Left, Right, and Forward inference probabilities in supermarket and library environments, illustrated using randomly selected snapshots from different timestamps

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

By creating a model that can understand high-level intersection concepts and use them as visual navigational waypoints, this thesis examined vision-based navigation for nano-scale drones in GPS-denied environments. To determine the best formulation for intersection identification, several categorization techniques were systematically assessed. Explainable AI approaches were used to test the model's conceptual comprehension of visual signals in order to guarantee that the learned representations were meaningful and interpretable. Lastly, the suggested method's capacity for generalization was evaluated in unseen real-world settings, indicating its potential for reliable use outside of the training environment.

Current state-of-the-art navigation techniques for nano drones, like PULP-DroNet, mostly concentrate on low-level reactive control and lack an explicit understanding of high-level scene semantics, as covered in Chapter 1. We developed an egocentric visual waypoint-based navigation framework that makes use of vision-based intersection recognition in order to overcome these constraints. The suggested method enables higher-level decision-making beyond collision avoidance by allowing the drone to identify intersections and bends as significant navigational cues. This concept enhances navigational flexibility, interpretability, and generalization in complicated, GPS-denied situations by enabling the drone to consider several viable pathways at decision points.

As described in Section 3.4.5, we first looked into several ways to formulate the intersection identification task and found limitations in traditional multi-class classification. In order to get around these limitations, we broke down high-level intersecting categories into directional labels and reconstructed the problem as a multi-label classification task. This method captures common visual characteristics across many junction kinds more successfully and performs better than multi-class formulations.

With Grad-CAM activation maps that resemble occipital place area-like navigational affordance representations seen in the human brain, we show that the model pays attention to semantically significant junction aspects. This claim is supported in detail in Section 4.3, where the interpretability of the learnt representations is highlighted by the strong similarities between the reconstructed OPA maps and the localization maps produced by Grad-CAM.

We argue that a model trained solely on digital twin data which achieves a real-world F1-score of 0.6 substantially surpassing a random classifier with an F1-score of 0.5. Experiment E, which is described in Section 3.4.6 and shown in Figure 3.15,

was used to specifically assess this capacity. These results demonstrate the robust sim-to-real transfer capability of the suggested methodology, even in the absence of real-world training data, as summarized in Figure 4.8, highlighting the efficacy of digital twin-based learning when real data is limited. The Digital Twin (Last 2) and Digital Twin (All) configurations in Figure 4.8 demonstrate that by incorporating real-world data during training can further improves the performance of the model.

5.2 Limitations and Future Work

Despite the promising results demonstrated in this research, there are few limitations to acknowledge. The performance of the proposed algorithm degrades in low-light environments due to the reliance on visual features, indicating the need for enhanced illumination robustness. The maneuvering strategy described in Section 3.4 requires fine-tuning based on the width of the intersection and is currently best suited for sharp 90 degree junctions. In this research, we chose locations with the aisle widths varying from 1-1.5m. Furthermore, our algorithm is primarily effective in supermarket and library environments because of its repetitive structure(literature shows that supermarkets have repetitive structures[48]).

One possible direction for future research is to examine the connection between the visual characteristics learned by the suggested models, as shown by Grad-CAM analysis, and navigational affordance maps shown in the human brain, namely inside the occipital place area (OPA). To improve the digital twin architecture for safer, more efficient, and privacy-preserving training and validation, an automated pipeline to convert real-world video into high-fidelity 3D simulation environments can be created. To evaluate robustness under difficult navigational conditions, the work can be expanded to more complicated and ambiguous intersection geometries, like Y-intersections. Another potential extension is to examine the effectiveness of zero-shot learning, where the model will be trained only using orthogonal images(left only, right only and forward only) and to evaluate whether it can generalize this knowledge to interpret and explain unseen intersection configurations. Furthermore, through model quantization(int-4) and pruning, the framework can be extended toward real-time on-board deployment, allowing effective vision-based navigation on resource-constrained aerial platforms in GPS-denied conditions. In this way, the transmission is not required and will increase the frames per second. Although the system is resilient to artefacts generated by the gray-scale camera due to packet loss, these artefacts can be mitigated while doing the on board execution.

A possible approach to enhance maneuvering efficiency in situations with multiple paths for navigation, like complex junctions, is to incorporate temporal information. Temporal context has been explored in related domains, including UAV tracking [49] and recurrent neural network–based navigation for room crossing with obstacle avoid-

ance [50]. However, the viability of traditional sequential models is limited by the tight memory and processing resource constraints imposed by nano-scale drones. Thus, lightweight temporal modeling techniques provide a useful substitute. For instance, the Temporal Shift Module (TSM) [51] enables joint spatial–temporal modeling with negligible computational overhead, making it well suited for resource-constrained platforms. It is a plug-and-play module that allows temporal reasoning without requiring any additional FLOPs or parameters. Integrating such mechanisms would allow for systematic performance and accuracy comparisons between spatial-only and spatial-temporal models.

To summarize, in future the following aspects can be explored further to extend the work presented in this thesis:

- The connection between the visual characteristics learned by the suggested models, as shown by Grad-CAM analysis, and navigational affordance maps shown in the human brain.
- The identification of complicated and ambiguous intersection geometries, like Y-intersections.
- The effectiveness of zero-shot learning, where the model will be trained only using orthogonal images.
- The real-time onboard deployment.
- The Incorporation of lightweight temporal modeling to enhance maneuvering in situations where there are multiple navigable paths at intersections. This allows for direct accuracy comparisons between spatial-only and spatial-temporal models without the need for extra maneuvering fine-tuning.

REFERENCES

[1] M. Y. Arafat, M. M. Alam, and S. Moh, "Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges," *Drones*, vol. 7, no. 2, 2023.

[2] A. Elamin, A. El-rabbany, and S. Jacob, "Event-Based Visual / Inertial Odometry for UAV Indoor Navigation," 2025.

[3] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi, "Improving Autonomous Nano-Drones Performance via Automated End-to-End Optimization and Deployment of DNNs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 548–562, 2021.

[4] L. Lamberti, "ENABLING MULTI-TASKING AI-BASED PERCEPTION ON AUTONOMOUS NANO-UAVS," Ph.D. dissertation, 2024. [Online]. Available: https://amsdottorato.unibo.it/id/eprint/11643/3/lamberti_lorenzo_tesi.pdf

[5] M. F. Bonner and R. A. Epstein, "Coding of navigational affordances in the human visual system," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, no. 18, pp. 4793–4798, 2017.

[6] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that?" pp. 1–4, 2017. [Online]. Available: http://arxiv.org/abs/1611.07450

[7] A. Garcia, S. S. Mittal, E. Kiewra, and K. Ghose, "A Convolutional Neural Network Feature Detection Approach to Autonomous Quadrotor Indoor Navigation," *IEEE International Conference on Intelligent Robots and Systems*, pp. 74–81, 2019.

[8] Y. Lu, Z. Xue, G. S. Xia, and L. Zhang, "A survey on vision-based UAV navigation," *Geo-Spatial Information Science*, vol. 21, no. 1, pp. 21–32, 2018. [Online]. Available: http://doi.org/10.1080/10095020.2017.1420509

[9] M. Y. Arafat and S. Moh, "Bio-inspired approaches for energy-efficient localization and clustering in uav networks for monitoring wildfires in remote areas," *IEEE Access*, vol. 9, no. civil, pp. 18 649–18 669, 2021.

[10] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "UAV autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, vol. 7, pp. 117 227–117 245, 2019.

[11] T. C. Knowles, A. G. Summerton, J. G. Whiting, and M. J. Pearson, "Ring Attractors as the Basis of a Biomimetic Navigation System," *Biomimetics*, vol. 8, no. 5, pp. 1–16, 2023.

[12] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 21–28, 2010.

[13] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[14] I. Sa, M. Kamel, M. Burri, M. Bloesch, R. Khanna, M. Popovic, J. Nieto, and R. Siegwart, "Build Your Own Visual-Inertial Drone," *IEEE Robotics & Automation Magazine*, no. March 2018, pp. 89–103, 2017.

[15] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, Control, and Planning for Aggressive Flight with a Small Quadrotor with a Single Camera and IMU," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.

[16] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, 2022.

[17] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64mW DNN-based Visual Navigation Engine for Autonomous Nano-Drones," 5 2018. [Online]. Available: http://arxiv.org/abs/1805.01831http://dx.doi.org/10.1109/JIOT.2019.2917066

[18] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Muller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi, "Fully Onboard AI-Powered Human-Drone Pose Estimation on Ultralow-Power Autonomous Flying Nano-UAVs," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1913–1929, 2022.

[19] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[20] L. Lamberti, L. Bellone, L. Macan, E. Natalizio, F. Conti, D. Palossi, and L. Benini, "Distilling Tiny and Ultrafast Deep Neural Networks for Autonomous

Navigation on Nano-UAVs," *IEEE Internet of Things Journal*, vol. 11, no. 20, pp. 33 269–33 281, 2024.

[21] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mW DNN-Based Visual Navigation Engine for Autonomous Nano-Drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[22] S. N. Fry and R. Wehner, "Honey bees store landmarks in an egocentric frame of reference," *Journal of Comparative Physiology - A Sensory, Neural, and Behavioral Physiology*, vol. 187, no. 12, pp. 1009–1016, 2002.

[23] M. Kutbi, H. Li, Y. Chang, B. Sun, X. Li, C. Cai, N. Agadakos, G. Hua, and P. Mordohai, "Egocentric Computer Vision for Hands-Free Robotic Wheelchair Navigation," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 107, no. 1, 2023.

[24] B. Pan, A. W. Harley, C. K. Liu, and L. J. Guibas, "LookOut: Real-World Humanoid Egocentric Navigation," 2025. [Online]. Available: http://arxiv.org/abs/2508.14466

[25] A. Prakash, R. Tu, M. Chang, and S. Gupta, "3D Hand Pose Estimation in Everyday Egocentric Images," pp. 183–202, 2025.

[26] C. Plizzari, G. Goletto, A. Furnari, S. Bansal, F. Ragusa, G. M. Farinella, D. Damen, and T. Tommasi, "An Outlook into the Future of Egocentric Vision," *International Journal of Computer Vision*, vol. 132, no. 11, pp. 4880–4936, 2024. [Online]. Available: https://doi.org/10.1007/s11263-024-02095-7

[27] Z. Qiu, Z. Liu, W. Niu, T. Bhattacharjee, and S. Kalantari, "EgoCogNav: Cognition-aware Human Egocentric Navigation," 2025. [Online]. Available: http://arxiv.org/abs/2511.17581

[28] W. Wang, C. K. Liu, and M. Kennedy, "Egocentric Scene-aware Human Trajectory Prediction," 2024. [Online]. Available: http://arxiv.org/abs/2403.19026

[29] J. Krantz, A. Gokaslan, D. Batra, S. Lee, and O. Maksymets, "Waypoint Models for Instruction-guided Navigation in Continuous Environments," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 15 142–15 151, 2021.

[30] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sunderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments," *Proceed-*

ings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3674–3683, 2018.

[31] S. Liu, H. Zhang, Y. Qi, P. Wang, Y. Zhang, and Q. Wu, "AerialVLN: Vision-and-Language Navigation for UAVs," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 15 338–15 348, 2023.

[32] C. Chen, S. Majumder, Z. Al-Halah, R. Gao, S. K. Ramakrishnan, and K. Grauman, "Learning To Set Waypoints for Audio-Visual Navigation," *ICLR 2021 - 9th International Conference on Learning Representations*, pp. 1–16, 2021.

[33] B. R. Mantha and B. Garcia de Soto, "Designing a reliable fiducial marker network for autonomous indoor robot navigation," *Proceedings of the 36th International Symposium on Automation and Robotics in Construction, ISARC 2019*, no. Isarc, pp. 74–81, 2019.

[34] A. Giusti, J. Guzzi, D. C. Ciresan, F. L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[35] A. Garcia, E. Mattison, and K. Ghose, "High-speed vision-based autonomous indoor navigation of a quadcopter," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pp. 338–347, 2015.

[36] A. Garcia, S. S. Mittal, E. Kiewra, and K. Ghose, "A convolutional neural network vision system approach to indoor autonomous quadrotor navigation," *2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019*, no. May, pp. 1344–1352, 2019.

[37] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa, "Deep Neural Network for Autonomous UAV Navigation in Indoor Corridor Environments," *Procedia Computer Science*, vol. 133, pp. 643–650, 2018. [Online]. Available: https://doi.org/10.1016/j.procs.2018.07.099

[38] S. S. Mansouri, P. Karvelis, C. Kanellakis, A. Koval, and G. Nikolakopoulos, "Visual Subterranean Junction Recognition for MAVs based on Convolutional Neural Networks," *IECON Proceedings (Industrial Electronics Conference)*, vol. 2019-Octob, pp. 192–197, 2019.

[39] E. Dack, L. Brigato, M. McMurray, M. Fontanellaz, T. Frauenfelder, H. Hoppe, A. Exadaktylos, T. Geiser, M. Funke-Chambour, A. Christel, L. Ebner, and

S. Mougiakakou, “An Empirical Analysis for Zero-Shot Multi-Label Classification on COVID-19 CT Scans and Uncurated Reports,” *Proceedings - 2023 IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2023*, pp. 2626–2635, 2023.

[40] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, 2020.

[41] Z. C. Lipton, “The mythos of model interpretability,” *Communications of the ACM*, vol. 61, no. 10, pp. 35–43, 2018.

[42] L. V. Haar, T. Elvira, and O. Ochoa, “An analysis of explainability methods for convolutional neural networks,” *Engineering Applications of Artificial Intelligence*, vol. 117, no. October 2022, p. 105606, 2023. [Online]. Available: https://doi.org/10.1016/j.engappai.2022.105606

[43] D. Palossi, F. Conti, and L. Benini, “An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs,” *Proceedings - 15th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2019*, pp. 604–611, 2019.

[44] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, “GAP-8: A RISC-V SoC for AI at the Edge of the IoT,” *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, vol. 2018-July, pp. 1–4, 2018.

[45] K. He, “Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification,” 2014.

[46] E. Spera, A. Furnari, S. Battiato, and G. M. Farinella, “Egocart: A benchmark dataset for large-scale indoor image-based localization in retail stores,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1253–1267, 2021.

[47] A. Nada and M. Chen, “Visual Question Answering,” *2024 International Conference on Computing, Networking and Communications, ICNC 2024*, pp. 6–10, 2024.

[48] K. Schlegel, P. Neubert, and P. Protzel, “Building a Navigation System for a Shopping Assistant Robot from Off-the-Shelf Components,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12228 LNAI, pp. 103–115, 2020.

[49] X. Yuan, T. Xu, X. Liu, Y. Wang, H. Qin, Y. Fang, and J. Li, "Multi-Step Temporal Modeling for UAV Tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 8, pp. 7216–7230, 2024.

[50] K. Kelchtermans and T. Tuytelaars, "How hard is it to cross the room? – Training (Recurrent) Neural Networks to steer a UAV," 2017. [Online]. Available: http://arxiv.org/abs/1702.07600

[51] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7082–7092, 2019.

[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014.

[53] Z. Li, F. Li, W. Zhang, Z. Zheng, X. Liu, Y. Liu, and L. Zeng, "THUD++: Large-Scale Dynamic Indoor Scene Dataset and Benchmark for Mobile Robots," pp. 1–11, 2024. [Online]. Available: http://arxiv.org/abs/2412.08096

[54] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 2242–2251, 2017.

APPENDIX A

MOBILENETV2 RESULTS

The K-Fold performance of the model is tabulated in the Table A.1. We can observe that Kaiming He and ImageNet weight initialization methods converge faster compared to digital twin setting. However, the highest accuracy was achieved by the digital twin weight initialization method trained on whole network. The Grad-CAM visualization tables are illustrated in Table A.2 and Table A.3. The forward Grad-CAM heatmaps attention can be considered as meaningful, however the left and right navigability heatmaps are not consistent across the same label's different images. Both ImageNet configuration and digital twin configuration have higher potential of generalizability as they achieve approximately the same highest values in Figure A.2. We show that even for light weight MobileNetV2 model trained solely on digital twin data achieves a real-world F1 score of 0.51, which is slightly better than that of a random classifier(0.5).

TABLE A.1: K-FOLD PERFORMANCE COMPARISON ACROSS EXPERIMENTS A–D (MOBILENETV2)

	Kaiming He		**ImageNet**		**Digital Twin (Last 2)**		**Digital Twin (All)**	
Fold	**Best Val Acc**	**Epochs**	**Best Val Acc**	**Epochs**	**Best Val Acc**	**Epochs**	**Best Val Acc**	**Epochs**
1	0.9987	9	0.9987	10	0.8015	13	1.0000	14
2	0.9987	8	0.9962	11	0.8435	16	1.0000	13
3	0.9987	10	0.9975	11	0.8181	19	0.9975	11
4	1.0000	14	0.9987	10	0.8321	19	1.0000	11
5	0.9990	10	1.0000	11	0.8510	14	0.9975	7

Note. Comparison of K-fold validation accuracy and convergence epochs for MobileNetV2 under different weight initialization and domain adaptation strategies.

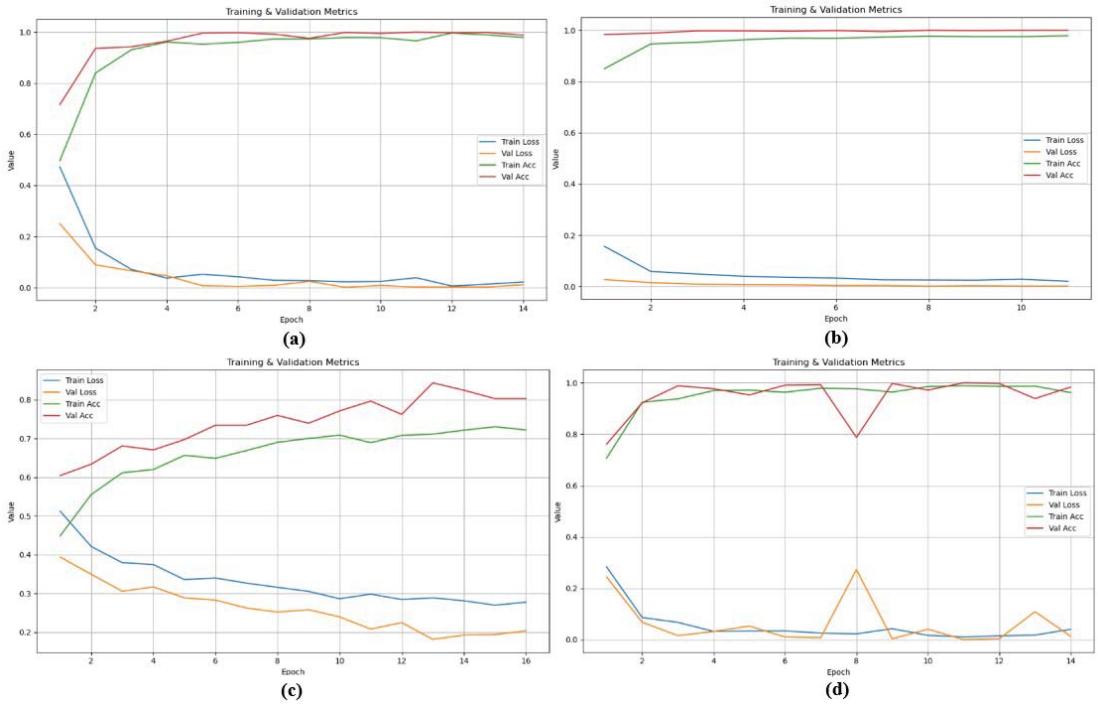


Fig. A.1: Training graphs of different configurations for MobileNetV2 model(a) Kaiming He Initialization (b) ImageNet Initialization (c) Synthetic last two (d) Synthetic all

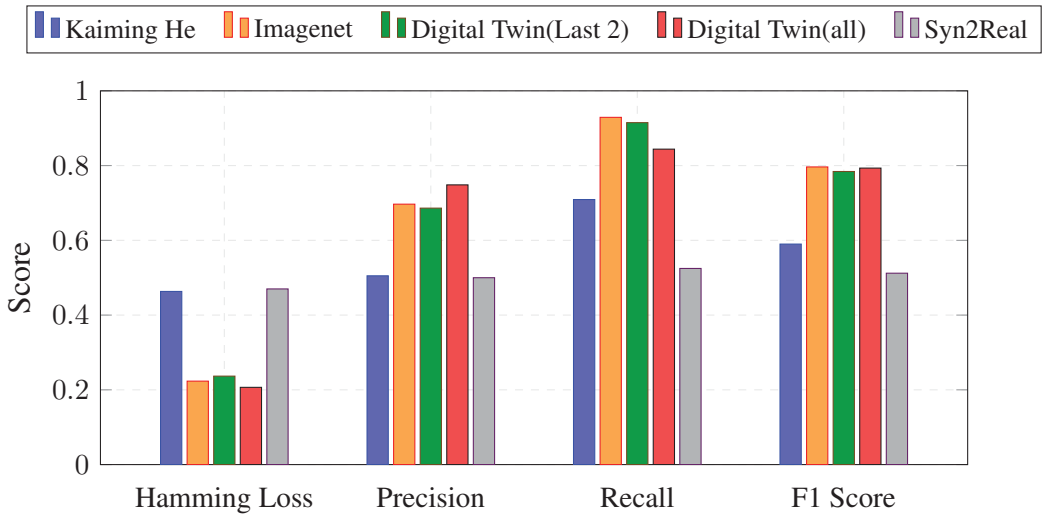
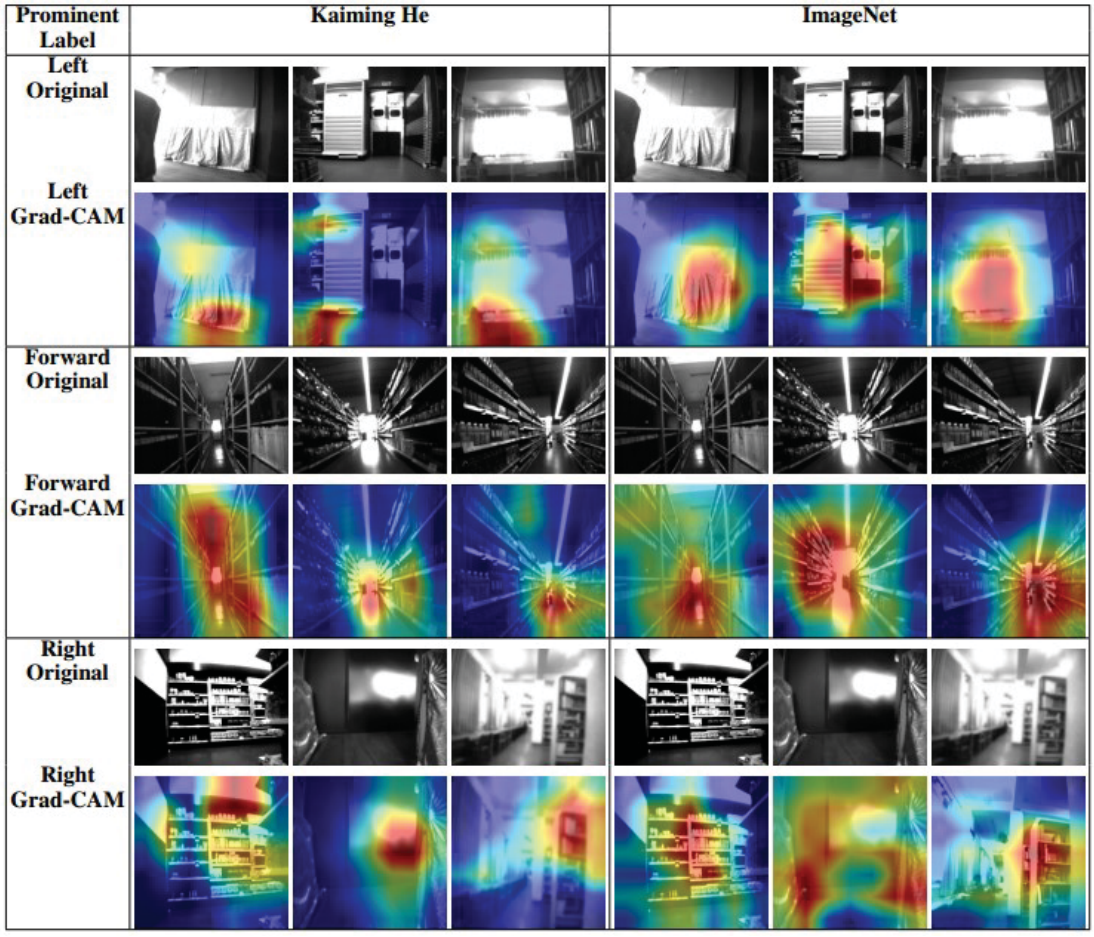


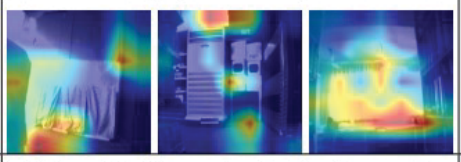
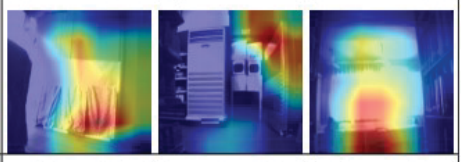
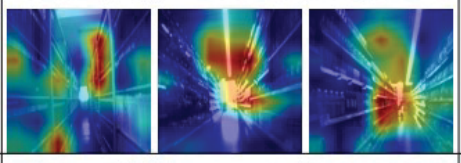
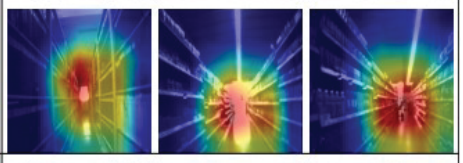
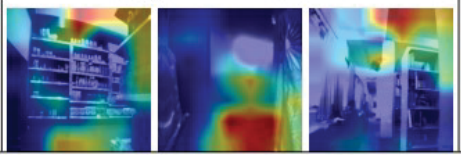
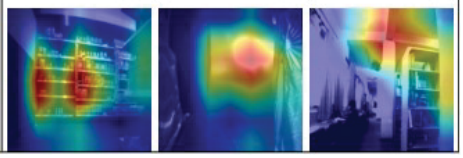
Fig. A.2: Comparison of four experiments across Hamming Loss, Precision, Recall, and F1 Score on test data(MobileNetV2).

TABLE A.2: GRAD-CAM VISUALIZATION RESULTS FOR MULTI-LABEL CONFIGURATION ON THE HE AND IMAGENET SETTINGS USING THREE DIFFERENT SAMPLES (MOBILENETV2)



Note. Distribution of samples across different classes used in the study.

TABLE A.3: GRAD-CAM VISUALIZATION RESULTS FOR MULTI-LABEL CONFIGURATION ON THE DIGITAL TWIN SETTINGS USING THREE DIFFERENT SAMPLES (MOBILENETV2)

Prominent Label	Digital Twin (Last 2)	Digital Twin (All)
Left Original		
Left Grad-CAM		
Forward Original		
Forward Grad-CAM		
Right Original		
Right Grad-CAM		

Note. Distribution of samples across different classes used in the study.

APPENDIX B

GRAD-CAM ABLATION STUDY

To study the verification in Grad-CAM visualizations, we conduct an explanation sanity check which is similar to the experiments presented in the paper [52]. Figure B.1a shows that after applying dotted patches near the edge the attention does not change which indicates that those edges provide a tighter correspondence to the model's prediction. However, we can observe that towards the bottom region of the edge, the attention has been reduced.

Figure B.1b represents another study where we compare original image and its occluded image with left,right and forward heatmaps. Here, we can observe how the predicted probability shifts after occluding the images. For example in the top library image, we can observe that the after occluding the highlighted area in the original image, the left probability image passes the previously top right label image(positions swapped).

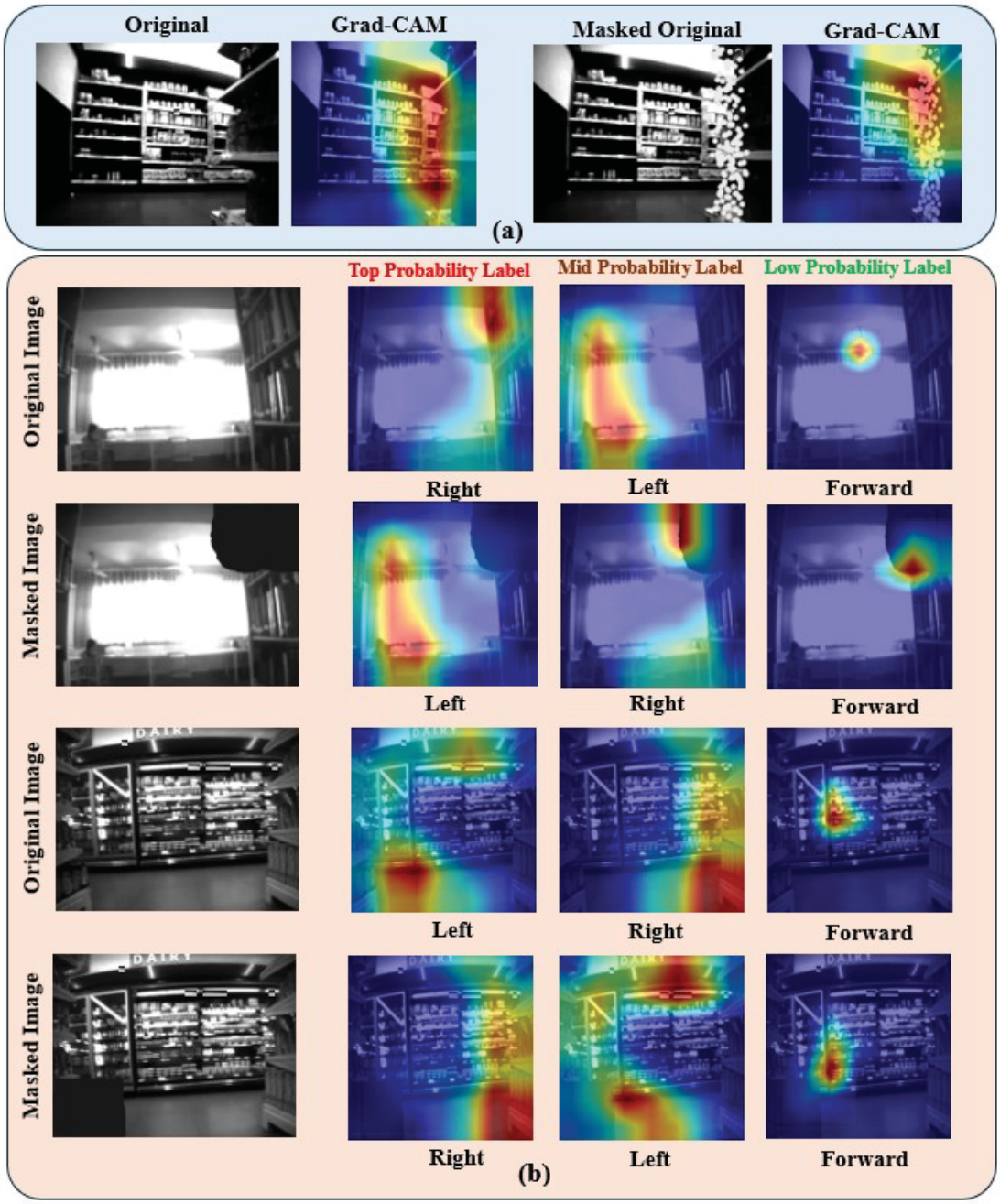


Fig. B.1: (a)Representation of heatmap attention reduction after masking using white dots in the Masked Original Image. (b) Comparison of how the probability of the label position changes after masking is performed.

APPENDIX C

VIDEO TO SIMULATION TOOL

One of the major limitations as discussed in the Section 2.1.3 is the limitation of categorized intersection data. Although there are some datasets capturing supermarket features [46, 53], the data is not categorized according to the intersection type and doing it manually takes a lot of time. Therefore, we tried several techniques to address this problem.

C.1 Simulation to Real Image Generation

Transferring images from one domain to another domain is an interesting research problem and many researchers have introduced different tools for this. CycleGAN [54] is a tool for image-to-image translation from one domain to another in the absence of paired images. We incorporated this tool to generate real world images from synthetic data with the goal of later converting a categorized synthetic image to a real world scene after the model training is completed.

Fig. C.1: Synthetic(image of epoch 1) to Real(image of epoch 30) translation

Figure C.1 illustrates how the conversion occurs with the training starting from epoch 1 to epoch 30. Though the GAN converted synthetic images to real-world scenes, they were not good at generating features in the intersections.

C.2 Pipeline for Model Generation

As we discussed in the Section 3.4.1, collecting data in real-world environment data is tedious, entirely manual, may damage the drone while it is in use and a privacy breach. To provide a solution for this, we experimented to create a pipeline as shown in Figure C.2. We took a video from the phone camera in a supermarket aisle environment and converted that video into a 3d model. The goal is to export this 3D model to Webots platform so that we can use it to simulate.

We tried several tools to generate a 3D model. Colmap, Hunyuan, metashape were the software that we experimented with. Out of these software, metashape was the

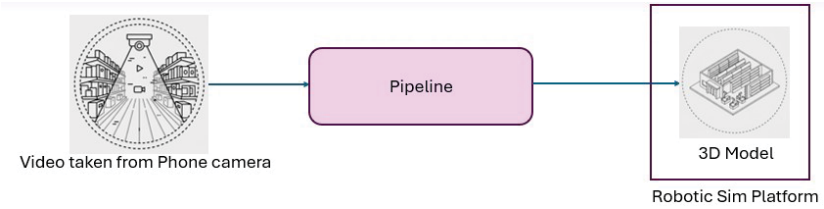


Fig. C.2: Video to simulation pipeline

best candidate for our task. However, there were some issues with scaling and texture reconstruction of this method as illustrated in Figure C.3.

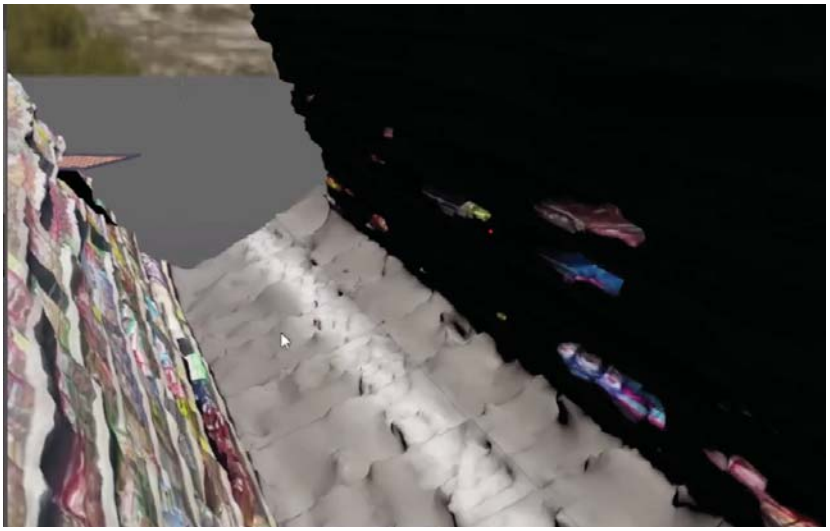


Fig. C.3: Imported 3D model to Webots showing scaling issues and only one sided aisle construction

APPENDIX D

MODEL QUANTIZATION

In intersection identification framework, we consider two processing options: which are on board processing and off board processing as we discussed in the Section 3.4. For on board processing, the first step is to quantize the model.We use pyTorch's torchao and performed the static quantization which is the recommended technique for CNN quantization. We quantized two models one from each ResNet50 and MobilenetV2 to perform this task and evaluated them on the test data.

TABLE D.1: PERFORMANCE ON TEST DATA BY RESNET50(LAST 2 CONFIG)

Parameter	**FP-32**	**INT-8**
Hamming Loss	0.2500	0.2533(1.32%)
Precision	0.6667	0.6633(-0.51%)
Recall	0.9362	0.9362(0%)
F1-score	0.7788	0.7765(-0.3%)
Model size(MB)	94.06	24.1(-74.4%)

TABLE D.2: PERFORMANCE ON TEST DATA BY MOBILENETV2(LAST 2 CONFIG)

Parameter	**FP-32**	**INT-8**
Hamming Loss	0.2367	0.2367(0%)
Precision	0.6862	0.6966(+1.5%)
Recall	0.9149	0.8794(-3.8%)
F1-score	0.7842	0.7774(-0.8%)
Model size(MB)	9.2	2.6(-71.7%)

As shown in the Table D.1 and Table D.2, it can be seen that we are able to successfully shrink the Resnet50 model by 73% of the original size without dropping the accuracy less than 1% on the F1-score. A similar trend can be observed in the MobilentV2 models also.

APPENDIX E

SAFE LANDING MECHANISM

One of the drawbacks of Crazyflie is that its instability when hovering in low textured, low light conditions and flying over its own shadow[1]. On some occasions, the Crazyflie tends to drift over these low textured floors in an unsafe manner. This is because the flow deck's optical flow sensor is unstable for low light conditions. We conducted several experiments to check the sensor reading values on different floors by flying the drone in the floor for a known time period. The goal was to trigger a landing sequence when the sensor readings get noisy and fall below the pre-defined threshold.

E.1 Flow Deck working principle

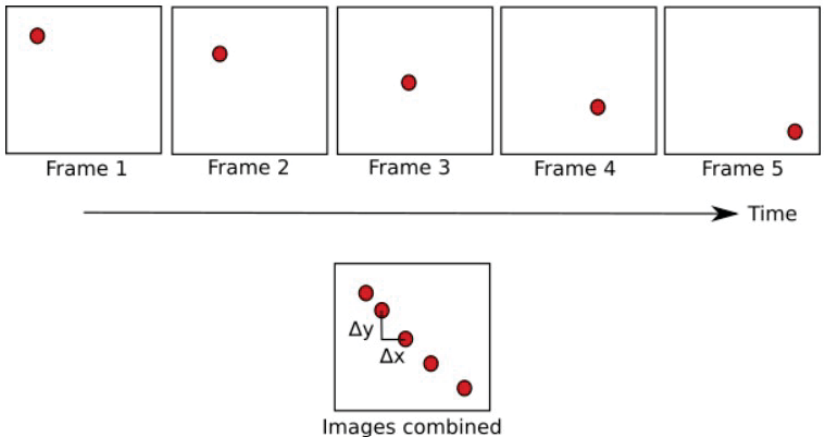


Fig. E.1: Feature tracking with the time(Reprinted from Bitcraze Documentation)

Figure E.1 shows how the features move from left to right when the sensor is moved from right to left[2]. The feature is monitored from one frame to the next, and the output shows how far it has moved since then. Knowing the distance to the features is one of the challenges of employing optical tracking from a flying platform. Measuring height is crucial since movement in relation to the ground must be taken into account. For this purpose, the VL53L0x ToF distance sensor has been added to Bitcraze's Flow deck to measure the distance to the surface being monitored.

E.2 Flow Deck's Sensor readings in different floors

Following are the parameters values and thier descriptions read from the PMW3901 optical flow sensor.

[1] https://www.bitcraze.io/tag/flow-deck/
[2] https://www.bitcraze.io/2017/11/optical-flow/

- deltaX = Flow X measurement (flow/fr)
- deltaY = Flow Y measurement (flow/fr)
- squal = Count of surface feature
- rawDatasum = Average raw data value

The following figures(Figure E.2, E.3, E.4, E.5) illustrate the different parameters of the sensor readings. It can be seen that squal value is best when it's flying over a wooden floor and worst over the green plain mat. Based on experimental evaluation, a threshold of 120 was selected, if the squal value is less than that value, the drone's landing sequence is executed preserving the crashing.

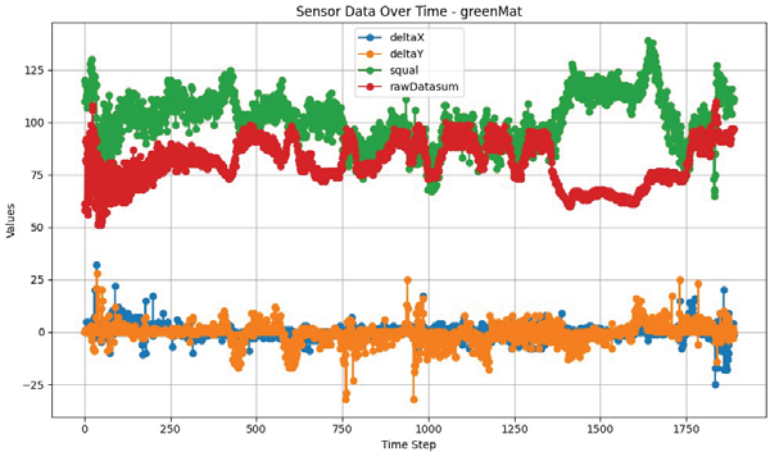


Fig. E.2: Flow deck sensor values on a Green Mat

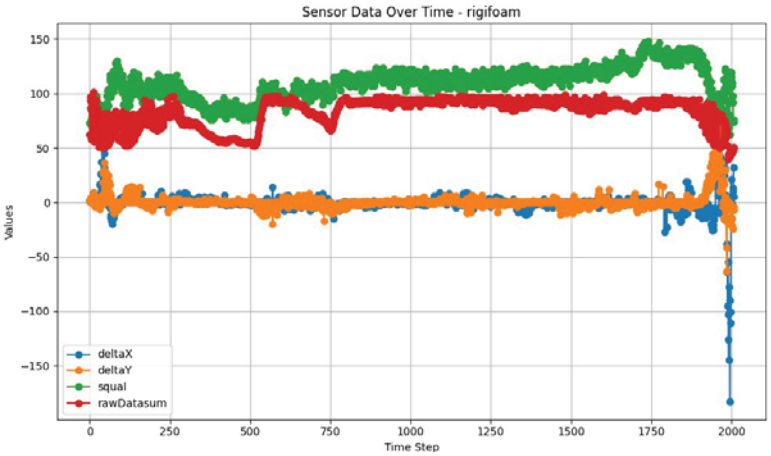


Fig. E.3: Flow deck sensor values on Rigifoam Floor

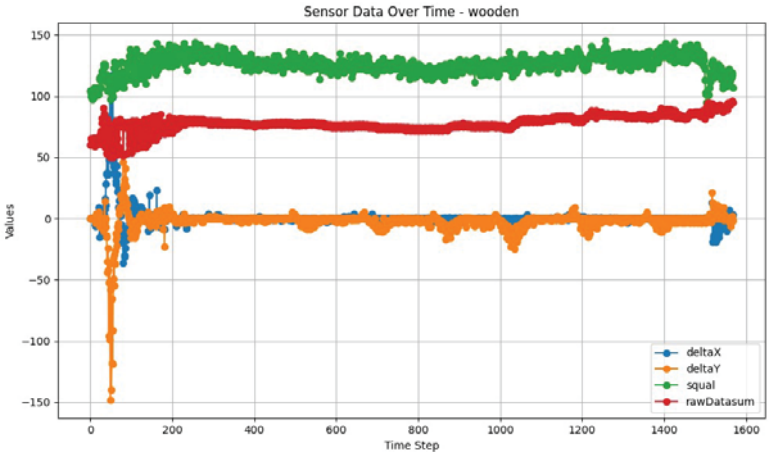


Fig. E.4: Flow deck sensor values on Wooden Floor

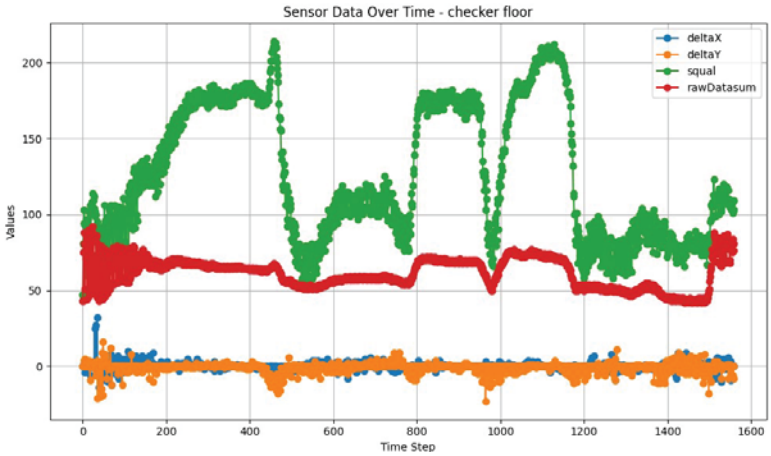


Fig. E.5: Flow deck sensor values on Checkered floor