

UNIVERSITY OF BAHRAIN

College of Engineering



**Department of
Electrical & Electronics
Engineering**

ROUTE PLANNING USING ARTIFICIAL INTELLIGENCE

A Thesis Submitted in Partial Fulfilment of the Requirements for the
M.Sc.Degree in Artificial Intelligence Systems

Submitted by

Ahmed Mohammed Majid Al-Khalidi

ID: 20180414

Supervised by

Dr. Ayman Al-Khazraji

University of Bahrain

Kingdom of Bahrain

November 2025

UNIVERSITY OF BAHRAIN

College of Engineering



Department of
Electrical & Electronics
Engineering

Defence committee's approval

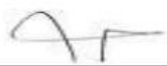
The student Ahmed Mohammed Al-Khalidi, ID 20180414, has defended the thesis by which he fulfilled the requirement for the degree of Master of Science entitled "Route Planning Using Artificial Intelligence".

Name:

Dr. Ayman Al-Khazraji (Supervisor)



Prof. Medhat Awadalla (External Examiner)



Prof. Mohab Mangoud (Internal Examiner)



Academic Department Stamp



Abstract

This study presents an independent Route Planning System using Artificial Intelligence that monitors small-scale environments, such as warehouses, retail stores, and car parking areas, to generate routes for ground robots. The system enables robots to navigate and avoid collisions by providing optimal paths without direct control. It employs semantic segmentation models—Fully Convolutional Network (FCN), U-Net, SegNet, and PSPNet—to transform top-view images into binary grids distinguishing free and occupied spaces. These grids are processed by route planning algorithms—A*, Dijkstra's, and Bidirectional—to compute efficient routes. Implemented in Python using PyTorch, the system demonstrates robust performance across 800 training and 200 testing images from the iSAID dataset. Results show U-Net consistently outperforming other models, achieving the lowest segmentation loss (0.0699) and highest scores across multiple metrics including Pixel Accuracy (0.970), Jaccard Index (0.948), and Recall (0.993). SegNet emerged as a strong competitor, while PSPNet demonstrated notable limitations. Route planning performed most efficiently on U-Net and SegNet-generated grids, with A* achieving the fastest execution times (0.15-0.18 ms) and lowest memory usage (0.024-0.025 MB), while Bidirectional yielded the shortest paths (7.05-7.57 units). In contrast, PSPNet-generated grids led to significantly longer paths (20.8-21.8 units) and higher computational demands. The system enhances navigation efficiency, safety, and cost-effectiveness for logistics applications in small-scale environments, with U-Net and A* identified as the optimal combination for real-time performance.

Publications

Route Planning Using Artificial Intelligence for Robotic Navigation in Dynamic Environment Submitted to The Journal of Engineering Research on November 12, 2025. Journal website: <https://tjer.squ.edu.om>

Dedication

I dedicate this work first to my parents, who always supported me, and second to the University of Bahrain, which provided my education and made this possible.

Acknowledgements

I sincerely thank my supervisor, Dr. Ayman Al Khazraji, for his help and support in completing my master's thesis on the Route Planning System using Artificial Intelligence. His advice and ideas during meetings were invaluable and made this work possible.

Table of Contents

Defence committee’s approval	A
Abstract	B
Publications	C
Dedication	D
Acknowledgements	E
Table of Contents	F
List of Tables	I
List of Figures	J
Abbreviations	L
Chapter One	1
Introduction	1
1.1 Introduction	2
1.2 Research Problem	3
1.3 Research Questions	4
1.4 Research Objectives	4
1.5 Research Significance	5
1.6 Study Setting	5
1.7 Research Terminologies	6
Chapter Two	8
Literature Review	8
2.1 Theoretical Framework	10
2.1.1 Fully Convolutional Network (FCN)	10
2.1.2 FCN Mathematical Formula	11
2.1.3 FCN Loss Function	11
2.1.4 U-Net	12
2.1.5 U-Net Mathematical Formula:	12
2.1.6 U-Net Loss Function:	13
2.1.7 Pyramid Scene Parsing Network (PSPNet)	13
2.1.8 PSPNet Mathematical Formula	14
2.1.9 PSPNet Loss Function	14
2.1.10 SegNet	15

2.1.11 SegNet Mathematical Formula	15
2.1.12 SegNet Loss Function	16
2.1.13 Optimization with Adam.....	16
2.1.14 Route Planning Algorithms	17
2.2 Literature Review.....	21
2.3 Chapter Conclusion.....	49
Chapter Three.....	51
Methods and Procedures	51
3.1 Route Planning with AI System Overview	52
3.2 Segmentation Models Design	55
3.2.1 U-Net Model	55
3.2.2 Description of the U-Net Mathematical Model	56
3.2.3 FCN Model	57
3.2.4 Description of the FCN Mathematical Model	59
3.2.5 PSPNET Model.....	60
3.2.6 Description of PSPNET Mathematical Model.....	62
3.3 SegNet Model	65
3.3.1 Description of SEGNET Mathematical Model.....	66
3.4 Dataset Preparation Process	70
3.4.1 The Dataset	70
3.4.2 Dataset Preparation Process Steps	71
3.5 Training and Testing for Segmentation Models	74
3.5.1 Overview of Training and Testing Workflow	74
3.5.2 Description of the Training and Testing Workflow.....	75
3.5.3 Project File Division	76
3.5.4 Training Process.....	77
3.5.5 Description of the Training Loop:	79
3.5.6 Performance Evaluation Metrics.....	80
3.5.7 Overview of Performance Evaluation.....	81
3.6 Route Planning Evaluation Process	87
3.6.1 Execution Time.....	88
3.6.2 Memory Usage.....	88
3.6.3 Path Length	89

3.7 Software Tools	90
3.7.1 Microsoft Paint.....	90
3.7.2 Visual Studio Code	91
3.8 Hardware.....	94
Chapter Four	95
Results, Discussion and Recommendations.....	95
4.1 Segmentation Models Training Result Analysis.....	96
4.2 Segmentation Models Performance Metrics Result Analysis.....	99
4.2.1 Pixel Accuracy Metric Results Analysis.....	101
4.2.2 Per-Class Accuracy Metric Results Analysis	102
4.2.3 Dice Coefficient Metric results analysis	103
4.2.4 Jaccard Index Metric results analysis	104
4.2.5 F1 Score Metric results analysis	105
4.2.6 Hausdorff Distance Metric results analysis	106
4.2.7 ASSD Metric results analysis	107
4.2.8 Precision Metric results analysis.....	108
4.2.9 Recall Metric results analysis	109
4.2.10 Frequency-Weighted IoU Metric results analysis.....	110
4.2.11 Boundary F1 Metric results analysis.....	111
4.3 Segmentation Models Visual Results Analysis	112
4.4 Route Planning Algorithms Results Analysis.....	113
4.5 Path Route Visual results	116
4.6 Discussion	120
4.7 Future Work	124
References.....	125
المخلص	i

List of Tables

Table (1): Training Configurations for each Segmentation Models	79
Table (2): Confusion Matrix for Binary Segmentation	81
Table (3): Path Planning (Mean $\pm$ SD) Results Across Models and Algorithms	115

List of Figures

Fig. 1: A* Algorithm Flowchart	18
Fig. 2: Dijkstra's Algorithm	19
Fig. 3: Bidirectional Search Flowchart	21
Fig. 4: A datapoint from turkey earthquake dataset (Gomroki, Hasanlou, et al., 2025)	22
Fig. 5: Architecture of two phase model (Li, He, et al., 2022)	24
Fig. 6: Segmentation of burned areas in Central Africa (Brand, Manandhar, et al., 2021)	25
Fig. 7: Classification accuracy results for each tree species in each model (Cui, Sun, et al., 2025)	26
Fig. 8: Model segmentation result (Ragunthar, Reddy, et al., 2025)	27
Fig. 9: UOSAM model structure (Xu, Chen, et al., 2025)	28
Fig. 10: Predicted PV area and energy and the ground truth (Jurakuziev, et al., 2023)	30
Fig. 11: BE-SegModel model architecture (Wang, Gao, et al., 2025)	32
Figure 12: Model development workflow (Ismael, Sadeq, et al., 2025)	33
Fig. 13: Seg–Unet model architecture (Abdollahi, Pradhan, et al., 2020)	35
Fig. 14: Spatial-Channel Attention UNet Model diagram (Anitha et al., 2024)	36
Fig. 15: EDFF-Unet model architecture (Wang, Fan, et al., 2025)	38
Figure 16: Hybrid A* Bidirectional Search flowchart (Jin, Li, et al., 2024)	40
Fig. 17: BiA*-ACO route recommendation system (Xia, Shen, et al., 2023)	43
Fig. 18: Hybrid LSTM and Dijkstra’s algorithm (Praveen Kumar, Hariharan, et al., 2023)	44
Fig. 19: Ant Colony and the A* Search hybrid model flowchart (Zhang, Wen, et al., 2023)	46
Fig. 20: Image Segmentation	53
Fig. 21: Segmented image to Binary Grid	54
Fig. 22: Route Planning	55
Fig. 23: U-Net Model Architecture	56
Fig. 24: FCN Model Architecture	58
Fig. 25: PSPNET Architecture	61

Fig. 26: SEGNET Architecture.....	66
Fig. 27: Dataset Preparation Process	73
Fig. 28: Training and Testing Workflow	75
Fig. 29: Project File Division.....	77
Fig. 30: Training Loop.....	78
Fig. 31: Performance Evaluation Overview.....	80
Fig. 32: Microsoft Paint Screenshot.....	91
Fig. 33: Visual Studio Code Screenshot	91
Fig. 34: Training Loss graphs per epoch for FCN Model.....	97
Fig. 35: Training Loss graphs per epoch for Unet Model.....	97
Fig. 36: Training Loss graphs per epoch for PSPNET Model	98
Fig. 37: Training Loss graphs per epoch for Segnet Model	98
Fig. 38: confusion matrices for all Segmentation models.....	100
Fig. 39: Pixel Accuracy Metric results	101
Fig. 40: Per- Class Accuracy Metric Results	102
Fig.41: Dice Coefficient Metric results	103
Fig. 42: Jaccard Index Metric results	104
Fig. 43: F1 Score Metric results.....	105
Fig. 44: Hausdorff Distance Metric results.....	106
Fig. 45: ASSD Metric results.....	107
Fig. 46: Precision Metric results	108
Fig. 47: Recall Metric results.....	109
Fig. 48: Frequency-Weighted IoU Metric results	110
Fig. 49: Boundary F1 Metric results	111
Fig. 50: Segmentation Models Visual Results	113
Fig. 51: Execution time results	114
Fig. 52: Memory Usage Results	114
Fig. 53: Path Length Results.....	115
Fig. 54: Route on Binary grid representations for all scenarios	118
Fig.55: Route on Actual environment region for all scenarios.....	119

Abbreviations

AI	Artificial Intelligence
ASSD	Average Symmetric Surface Distance
BCE	Binary Cross-Entropy
BCEWithLogitsLoss	Binary Cross-Entropy With Logits Loss
BN	Batch Normalization
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
FW-IoU	Frequency-Weighted Intersection over Union
GF-2	Gaofen-2 (Chinese satellite)
GPU	Graphics Processing Unit
IoU	Intersection over Union
iSAID	ISPRS Aerial Image Segmentation Dataset
JL-1	Jilin-1 (Chinese satellite constellation)
LiDAR	Light Detection and Ranging
MB	Megabytes
mIoU	Mean Intersection over Union
PIL	Python Imaging Library
PPM	Pyramid Pooling Module
PSPNet	Pyramid Scene Parsing Network
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
SegNet	Segmentation Network
TN	True Negative
TP	True Positive
U-Net	U-shaped Network
VGG16	Visual Geometry Group 16-layer network

Chapter One

Introduction

1.1 Introduction.....	2
1.2 Research Problem	3
1.3 Research Questions	4
1.4 Research Objectives.....	4
1.5 Research Significance.....	5
1.6 Study Setting.....	5
1.7 Research Terminologies.....	6

Chapter one

Introduction

1.1 Introduction

The importance of route planning within the contemporary economic framework is extensively recognized, as it constitutes a fundamental component of logistics operations across terrestrial, aerial, and maritime domains. Recent advancements in robotics technology, combined with increasing of elderly individuals, have necessitated the integration of robotic and drone-based solutions in confined environments, including warehouses, retail establishments, and manufacturing facilities. These technological and societal shifts have underscored the requirement for navigation systems specifically designed for small-scale settings. To navigate efficiently through the provision of real-time route optimization, autonomous agents such as robots and drones have been widely deployed.

To achieve effective navigation, the environment must be represented as a grid-based data structure, which is compatible with established algorithmic approaches, such as the A* and Dijkstra's algorithms. Google, for example, depends on complex infrastructural frameworks to process environmental data, as demonstrated by Google Maps application. The resource-intensive nature of such application highlights the necessity for a lightweight, small-scale route planning framework. A system is designed to capture top-view images of an environment using camera systems, which are then combined into a single image that represents the environment's layout at a specific moment in time. This composite image is then converted into a grid-based data structure to facilitate processing route planning algorithms. The

development of this system is motivated by the need to enhance operational efficacy in spatially limited environments, where conventional large-scale navigation infrastructure may prove impractical. The proposed framework uses image-based environmental representation and grid transformation to provide an optimized approach for robotic navigation. It specifically addresses the unique challenges faced in small-scale environments.

1.2 Research Problem

The challenge addressed by the proposed system, as outlined in the introduction, pertains to the transformation of an environmental state image into a grid-based representation. In this research, the environmental state is sought to be converted into a binary grid representation, wherein the environment is delineated into two distinct regions: occupied regions, denoted by 1s, and free regions, represented by 0s. This process necessitates the analysis and interpretation of shapes and textures within the top-view image, a task acknowledged as complex. In addition, advancements in deep learning technology have facilitated the resolution of this challenge through the application of segmentation models, which serve as the cornerstone of the proposed system. Consequently, a hybrid framework is established, integrating deep learning, image processing, and route planning algorithms. Within this study, four segmentation models, namely the Fully Convolutional Network (FCN), U-Net, SegNet, and PSPNet are employed. For route planning, three algorithms are utilized: A*, Dijkstra's, and bidirectional route planning algorithms. Through this approach, the effectiveness of the integrated system in enabling real-time navigation for small-scale environments is evaluated.

1.3 Research Questions

The following research questions were formulated to guide the investigation, as outlined in the introductory, which emphasizes the need for efficient navigation in small-scale environments using a hybrid framework of deep learning and route planning:

1. How effectively can environmental state images be transformed into binary grid representations? This includes deep learning segmentation models; the Fully Convolutional Network (FCN), U-Net, SegNet, and PSPNet.
2. What is the comparative performance of the FCN, U-Net, SegNet, and PSPNet models in generating accurate binary grids, as measured by various segmentation metrics?
3. How do the route planning algorithms A*, Dijkstra's, and Bidirectional perform when applied to binary grids generated by FCN, U-Net, SegNet, and PSPNet in terms of execution time, memory usage, and path length?
4. Which combination of segmentation model and route planning algorithm yields the most efficient and reliable?

1.4 Research Objectives

The objectives of this study were established to align with the research problem and questions, focusing on the development and evaluation of a hybrid navigation framework:

1. To develop a system that transforms top-view environmental images into binary grid representations using the semantic segmentation models FCN, U-Net, SegNet, and PSPNet.

2. To evaluate the performance of FCN, U-Net, SegNet, and PSPNet in generating accurate binary grids by employing a benchmark dataset and a comprehensive set of segmentation metrics.
3. To assess the efficiency of three route planning algorithms—A*, Dijkstra's, and Bidirectional—when applied to the binary grids produced by FCN, U-Net, SegNet, and PSPNet through metrics such as execution time, memory usage, and path length.
4. To determine the optimal combination of segmentation model and route planning algorithm for achieving efficient real-time navigation in small-scale environments.

1.5 Research Significance

1. A real-time route planning system analyses the environment to guide robots smoothly in warehouses, retail stores, and parks.
2. Costs drop as AI robots navigate efficiently, saving time and energy for tasks in warehouses, retail stores, and parks.
3. Short paths are made by route planning algorithms to navigate faster for doing different tasks.
4. Safety improves as the Route Planning system reduces collisions by avoiding obstacles like shelves or people.

1.6 Study Setting

This study was done to build and test an AI-based navigation system for robots in small areas like warehouses, retail stores, and parks. The work included image processing, deep segmentation models (Fully Convolutional Networks, U-Net, SegNet, and PSPNet) and route planning (A*, Dijkstra's, and Bidirectional search), all coded in Python. Anaconda was used to set up the Python environment and manage tools like NumPy, OpenCV, and PyTorch for handling data and building models. Visual

Studio Code was the main tool for writing, checking, and running the code. To create binary masks for image segmentation, Microsoft Paint was used to manually draw and edit top-view images, making data for training and testing. The experiments and model training ran on a gaming laptop, an ASUS TUF Gaming F15 (model FX507ZC), which greatly sped up the model's calculations. The system was equipped with a 12th Gen Intel Core i7-12700H processor, an NVIDIA GeForce RTX 3050 GPU, 16.0 GB of RAM, and a 64-bit version of Windows 11 Home.

1.7 Research Terminologies

1. Image Processing: Any mathematical operation on the images such as cropping, rotating etc...
2. Deep Learning: A machine learning model with huge number of trainable weights.
3. Segmentation Model: A machine learning model that assigns each pixel in an image to a specific class, supporting binary segmentation (two classes) or multi-class segmentation (multiple classes).
4. Route Planning: The process to find the point in the graph that connect the starting point with the targeting point.
5. Small-Scale Environment: Environment like Warehouse's, car parks, supermarkets etc...
6. Real-Time Navigation: A route planning system that keep updating the route if any change happens in the environment.

7. Environment Analysis: Simplifying the environment into a binary grid representation that can be passed to route planning algorithms.
8. Dynamic Obstacle: A moving object such as human robot.
9. Hybrid System: A complex system which combine multiple algorithms that work together to form a complex system.
10. Confusion Matrix: A diagram used in classification application to calculate performance metrics such as F1.
11. Performance Metrics: Mathematical equations used to measure the permanence of a model.

Chapter Two

Literature Review

2.1 Theoretical Framework.....	10
2.1.1 Fully Convolutional Network (FCN).....	10
2.1.2 FCN Mathematical Formula	11
2.1.3 FCN Loss Function	11
2.1.4 U-Net.....	12
2.1.5 U-Net Mathematical Formula:	12
2.1.6 U-Net Loss Function:.....	13
2.1.7 Pyramid Scene Parsing Network (PSPNet)	13
2.1.8 PSPNet Mathematical Formula.....	14
2.1.9 PSPNet Loss Function	14
2.1.10 SegNet.....	15
2.1.11 SegNet Mathematical Formula	15
2.1.12 SegNet Loss Function	16
2.1.13 Optimization with Adam.....	16
2.1.14 Route Planning Algorithms	17
2.2 Literature Review.....	21
2.3 Chapter Conclusion.....	49

Chapter Two

Literature Review

This chapter covers the theory for semantic segmentation models FCN, U-Net, SegNet, and PSPNet and route planning algorithms. It helps understand the key ideas behind the route planning system for robots in small-scale environments. The chapter is split into two sections. The first section explains the theory of models and algorithms. The second section reviews recent studies to show how this work connects to the field.

2.1 Theoretical Framework

This section discusses the theory behind semantic segmentation models and route planning algorithms used in the route planning system. It explains how FCN, U-Net, SegNet, and PSPNet process images to separate free and occupied spaces for mapping. It also describes how A*, Dijkstra's, and Bidirectional search algorithms create paths for robots. These concepts support the system's design and testing in small-scale environments.

2.1.1 Fully Convolutional Network (FCN)

It is a supervised machine learning semantic segmentation model which has this architecture: a series of convolutional layers that scan images to label each pixel for creating navigation maps. These layers shrink the image to find key patterns, then upscale it to produce a detailed map of free and occupied spaces. The design skips traditional dense layers, allowing it to process images of any size for the route planning system. This setup helps robots navigate by mapping complex environments accurately.

2.1.2 FCN Mathematical Formula

Let $(I \in R^{H \times W \times C})$ be an input image with height (H), width (W), and channels (C). The FCN model (F_{FCN}) transforms (I) into a segmentation map $(S \in R^{H \times W \times K})$ see equation 1, where (S) is the predicted output $((\hat{y}))$ labeling each pixel into (K) classes (e.g., free space, occupied space).

$$S = F_{FCN}(I) = U \left(C_i \left(\dots C_2(C_1(I)) \right) \right) \quad (1)$$

Here (C_i) represents the $(i) - th$ block of convolutional layers (with convolution, activation, and pooling), which shrinks the image, and (U) represents upsampling layers (with deconvolution) that restore the map to the original size. This formula captures how FCN processes the image through layered scans to label pixels for navigation maps.

2.1.3 FCN Loss Function

The predicted output $\hat{Y}$ is compared with the ground truth using a loss function. For binary classification, FCNs often use the Binary Cross-Entropy (BCE) loss:

$$BE = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2)$$

This formulation computes the loss over all $N = H \times W$ pixels, encouraging the predicted values $\hat{y}_i$ to match the true labels y_i . However, due to the encoder-only structure, FCNs often produce less precise segmentation boundaries because high-resolution details are lost in deep layers.

2.1.4 U-Net

It is a supervised machine learning semantic segmentation model which has this architecture: a U-shaped network with contracting layers, expanding layers, and skipping connections to build precise navigation maps. The contracting layers shrink the image to capture important details, while the expanding layers rebuild it into a clear map of free paths and obstacles. Skip connections link early and later layers to keep fine details, improving accuracy with limited data. This makes U-Net ideal for the route planning system in small-scale environments.

2.1.5 U-Net Mathematical Formula:

Let $(I \in R^{H \times W \times C})$ be an input image with height (H), width (W), and channels (C). The U-Net model (F_{U-Net}) transforms (I) into a segmentation map $(S \in R^{H \times W \times K})$, where (S) is the predicted output $((\hat{y}))$ labeling each pixel into (K) classes (e.g., free space, obstacle) for navigation maps. The general system is:

$$S = F_{U-Net}(I) = E_m \left(\dots E_2(E_1(I)) \right) + \sum_{i=1}^m SC_i \left(D_i \left(\dots D_1 \left(E_m(\dots E_1(I)) \right) \right) \right) \quad (3)$$

Here, (E_i) represents contracting layers (convolution and pooling) that shrink the image, (D_i) represents expanding layers (deconvolution) that rebuild the map, and (SC_i) represents skip connections that combine early layer outputs with later ones. The (+) denotes concatenation, showing how U-Net builds the predicted map (S) (or $(\hat{y}))$ for the route planning system.

2.1.6 U-Net Loss Function:

To train U-Net, a stable loss function called BCEWithLogitsLoss is used, combining sigmoid activation and binary cross-entropy for binary segmentation (e.g., free vs. occupied spaces). Equation 4 represents the loss function:

$$L_{U-Net} = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W \left[\max(z_{i,j}, 0) - z_{i,j} \cdot y_{i,j} + \log(1 + e^{-|z_{i,j}|}) \right] \quad (4)$$

(L_{U-Net}) is the total loss; (H) and (W) are the image's height and width; ($z_{i,j}$) is the model's raw prediction for pixel ((i,j)); ($y_{i,j}$) is the true label (0 or 1); $\max(z_{i,j}, 0)$ stabilizes positive logits; ($-z_{i,j} \cdot y_{i,j}$) aligns predictions with labels; ($\log(1 + e^{-|z_{i,j}|})$) ensures stable sigmoid computation. This formula avoids separate sigmoid calculations and prevents gradient issues, making it ideal for U-Net's deep architecture.

2.1.7 Pyramid Scene Parsing Network (PSPNet)

It is a supervised machine learning semantic segmentation model which has this architecture: a deep convolutional network with a pyramid pooling module to capture multi-scale context for precise scene parsing. The model uses a pre-trained ResNet-50 backbone to extract hierarchical features through five progressive stages, followed by a pyramid pooling module that aggregates contextual information at multiple scales. These multi-scale features are fused and upsampled to produce detailed segmentation maps that accurately delineate free spaces and obstacles. The design enables comprehensive scene understanding by integrating both local details and global context, making it highly effective for complex environment navigation in route planning systems.

2.1.8 PSPNet Mathematical Formula

Let I be an input image with height H , width W , and channels C . The PSPNet model F_PSP transforms I into a segmentation map S , where S is the predicted output $F_PSP(I)$ labeling each pixel into K classes (e.g., free space, occupied space). The general system is:

$$S = U \left(C \left(P(B(I)) \right) \right) \quad (5)$$

Here, B represents the ResNet-50 backbone that extracts hierarchical features through convolutional layers and residual blocks, P represents the pyramid pooling module that captures multi-scale context through parallel pooling branches, C represents the feature concatenation operation that fuses multi-scale representations, and U represents the upsampling layers that restore spatial resolution. This formula captures how PSPNet processes images through deep feature extraction and multi-scale context aggregation to generate precise navigation maps.

2.1.9 PSPNet Loss Function

To train PSPNet, we use the BCEWithLogitsLoss function, which combines sigmoid activation and binary cross-entropy for efficient binary segmentation. The loss function is defined as:

$$\mathcal{L}_{PSP} = -\frac{1}{N} \sum_{i=1}^N [y_i \cdot \log(\sigma(\hat{y}_i)) + (1 - y_i) \cdot \log(1 - \sigma(\hat{y}_i))] \quad (6)$$

where $\hat{y}_i$ represents the model's raw output for pixel i , y_i denotes the ground truth label (0 or 1), σ is the sigmoid function that converts logits to probabilities, and N is the total number of pixels. This formulation ensures numerical stability during training by combining the sigmoid and cross-entropy operations, preventing gradient saturation issues while effectively

optimizing the model parameters for accurate free space segmentation in navigation tasks.

2.1.10 SegNet

It is a supervised machine learning semantic segmentation model which has this architecture: a symmetric encoder-decoder network with pooling indices for precise spatial recovery in navigation mapping. The model uses a VGG16-based encoder that progressively extracts hierarchical features through five convolutional blocks, each followed by max pooling operations that store location indices. The decoder mirrors this structure with five corresponding blocks that use unpooling with stored indices to restore spatial resolution while refining features. This design enables accurate boundary delineation for free spaces and obstacles by maintaining spatial information through index-based upsampling rather than learnable deconvolution, making it computationally efficient for real-time route planning applications.

2.1.11 SegNet Mathematical Formula

Let I be an input image with height H , width W , and channels C . The SegNet model F_{SegNet} transforms I into a segmentation map S , where S is the predicted output $F_{\text{SegNet}}(I)$ labeling each pixel into K classes (e.g., free space, occupied space). The general system is:

$$S = D(E(I)) \quad (7)$$

Here, E represents the encoder function that processes the input through five convolutional blocks with max pooling and index storage: $E(I) = \text{pool}_5(\text{conv}_5(\dots \text{pool}_1(\text{conv}_1(I)) \dots))$, where each conv_i consists of multiple 3×3 convolutional layers with batch normalization and ReLU activation. D represents the decoder function that reconstructs the

segmentation map through five symmetric blocks with unpooling using stored indices:

$D(E(I)) = \text{conv_1}'(\text{unpool_1}(\dots \text{conv_5}'(\text{unpool_5}(E(I))) \dots)$, where each $\text{conv_i}'$ contains corresponding convolutional layers that progressively reduce channel dimensions while restoring spatial resolution.

2.1.12 SegNet Loss Function

SegNet utilizes the same BCEWithLogitsLoss function as described in section 2.1.9 for PSPNet, which combines sigmoid activation and binary cross-entropy in a numerically stable manner for binary segmentation tasks. This loss function effectively optimizes the network parameters by comparing the model's raw predictions with ground truth labels while preventing gradient saturation issues during training.

2.1.13 Optimization with Adam

All four segmentation models use the Adam optimizer for training, which adapts the learning rate for each parameter using first and second moments of the gradient. The update rule is given by:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon} \quad (8)$$

Here, η is the learning rate, $\widehat{m}_t$ and $\widehat{v}_t$ are the bias-corrected first and second moment estimates, and ϵ is a small constant for numerical stability. This optimizer helps accelerate convergence and stabilize training, especially for deep models like UNet.

2.1.14 Route Planning Algorithms

The A* algorithm starts by loading the environment and identifying the start and end positions. It first checks whether the grid is empty or if the path is completely blocked; if it is, the process stops. If the grid is valid, the algorithm copies it and creates a list of points to explore. This list begins with the start point and is sorted based on a score that adds the actual cost to reach a point and a guessed distance (heuristic) to the goal. As the algorithm runs, it picks the point with the lowest total score—this score is a sum of the distance already traveled and an estimated distance remaining. This helps A* focus its search on the goal instead of exploring blindly. For each selected point, it checks if it has reached the end. If not, it examines neighboring points. If a neighbor is valid and not blocked, it calculates the cost to reach it. If the new cost is better than a previously known one, it updates the path and includes the point in the list. This continues until the end point is found or all possible paths are checked. Once the goal is reached, A* traces back the shortest path and marks it on the environment grid. By combining known and estimated costs, A* is often faster than other methods in finding a path, especially in large or complex environments, “Fig. 1” shows the A star algorithm Flowchart.

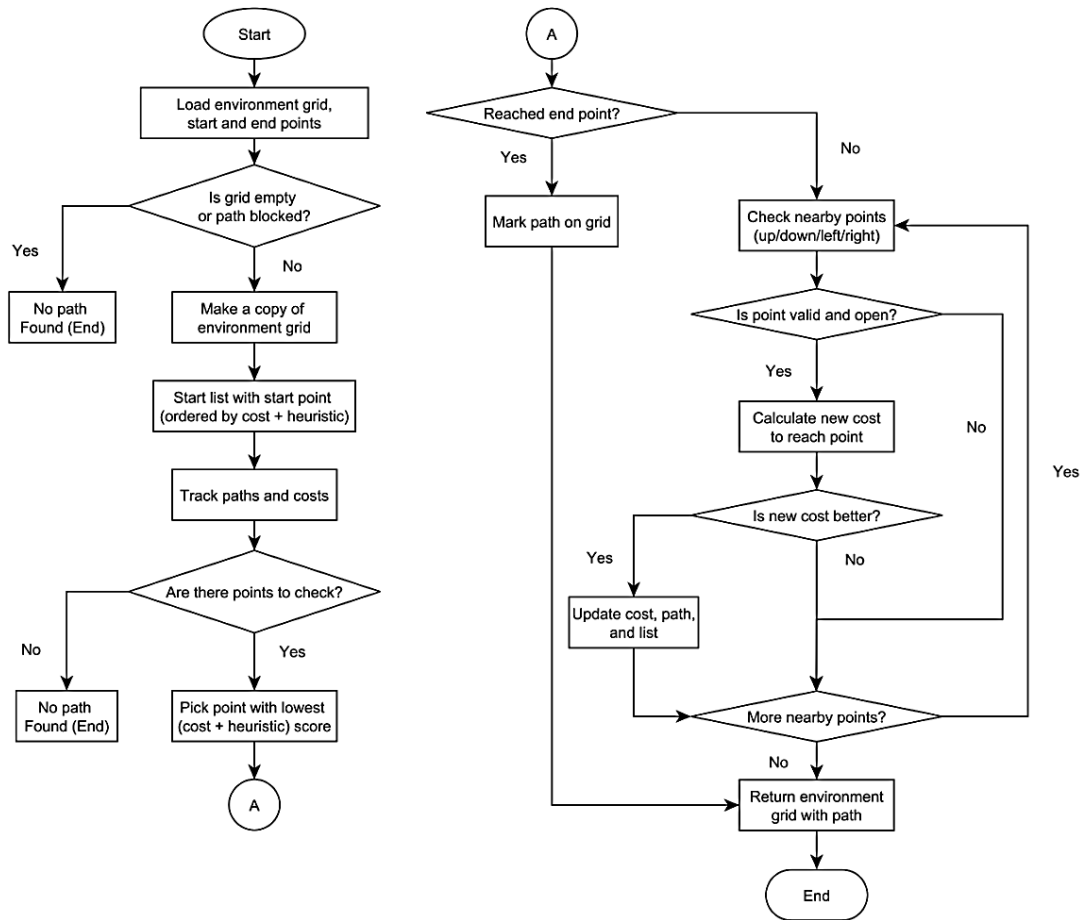


Fig. 1: A* Algorithm Flowchart

Dijkstra's algorithm also begins by loading the environment and checking the validity of the grid. If the path is blocked, it ends immediately. Otherwise, it copies the grid and starts a list with the start point. Unlike A*, Dijkstra's list is ordered only by the actual cost from the start point to each point explored. It does not use any estimated distance to the goal. At each step, the algorithm picks the point with the lowest accumulated cost and checks whether it has reached the end. If not, it explores all neighboring points in the four directions. For each valid neighbor, it calculates the total cost to get there. If this cost is lower than what was known before, it updates the list and path information. This ensures that the shortest known path is always pursued. The process repeats until the goal is reached or all paths are exhausted. When the shortest path is found, it is marked on the

grid and returned. Dijkstra's method is thorough and guarantees the shortest path, but it may take more time than A* because it explores points evenly in all directions without aiming toward the goal, “Fig. 2” shows the Dijkstra Flowchart.

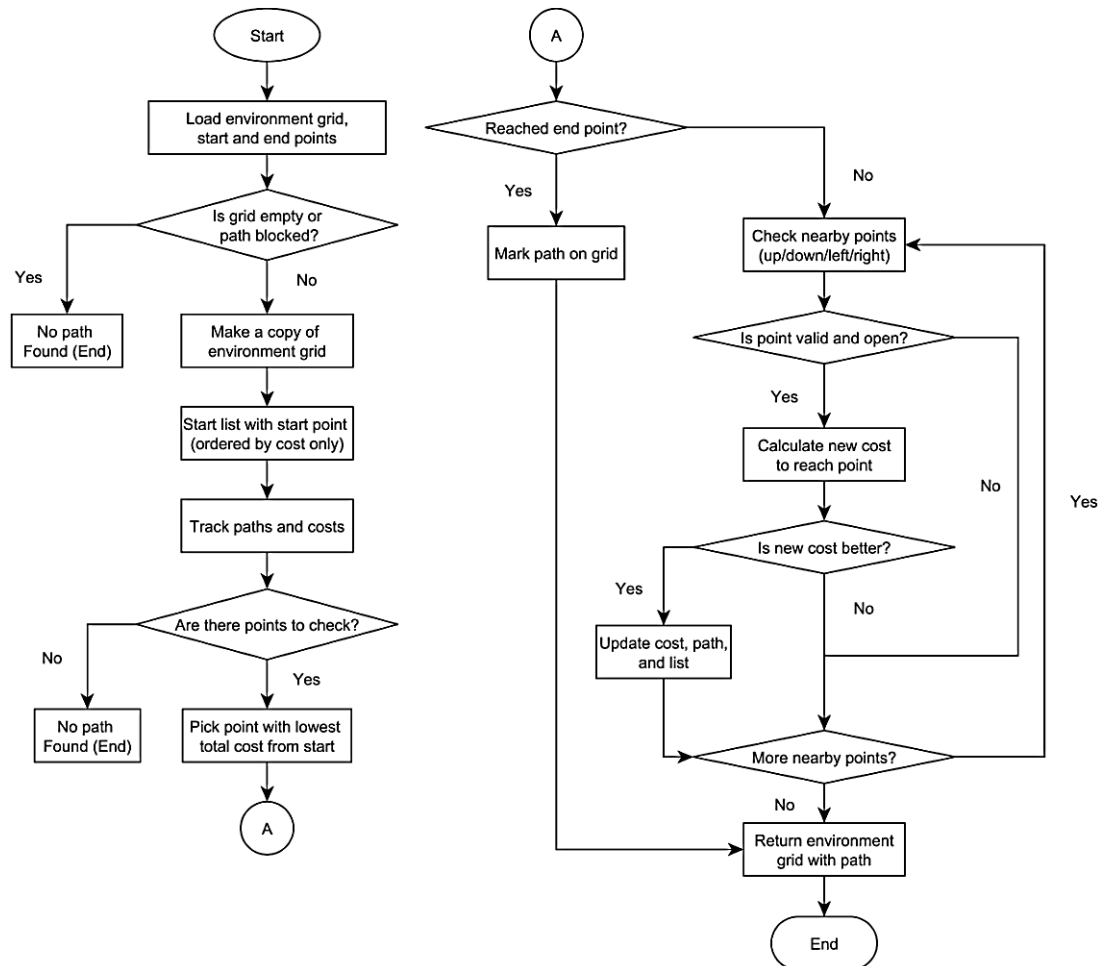


Fig. 2: Dijkstra's Algorithm

Bidirectional Search begins by loading the environment grid and defining both the start and end points. It first checks if the grid is valid and whether a path is possible. If not, it stops early. If the setup is valid, the algorithm starts two search fronts—one from the start and one from the end. This dual approach is key to its efficiency. Both lists keep track of their paths and travel costs independently, reducing the number of nodes each side needs to explore. In each iteration, the algorithm expands one

node from each search front. First, it selects the next best point from the start list and checks if that point has already been visited by the search from the end. If so, a meeting point is found, and the paths can be joined. If not, it continues to explore nearby points and updates the list. Then, the process is repeated for the search front coming from the end point. Each valid move updates the corresponding list with cost and path details. The biggest strength of Bidirectional Search is how it narrows the search space. Instead of exploring the entire path from one direction, it brings two search waves toward each other, meeting somewhere in the middle. This usually results in significantly fewer explored nodes, making the search faster, especially in large environments. Once a connection is made, the algorithm combines both partial paths, marks the complete route on the grid, and returns it. This makes Bidirectional Search ideal when both the start and goal are known ahead of time and the space is large or symmetric. When both searches meet at a common point, the paths are connected, marked on the grid, and returned. Bidirectional search can be much faster than A* or Dijkstra in large spaces because it cuts the search area nearly in half by working from both directions at once, “Fig. 3” shows the Bidirectional Search flowchart.

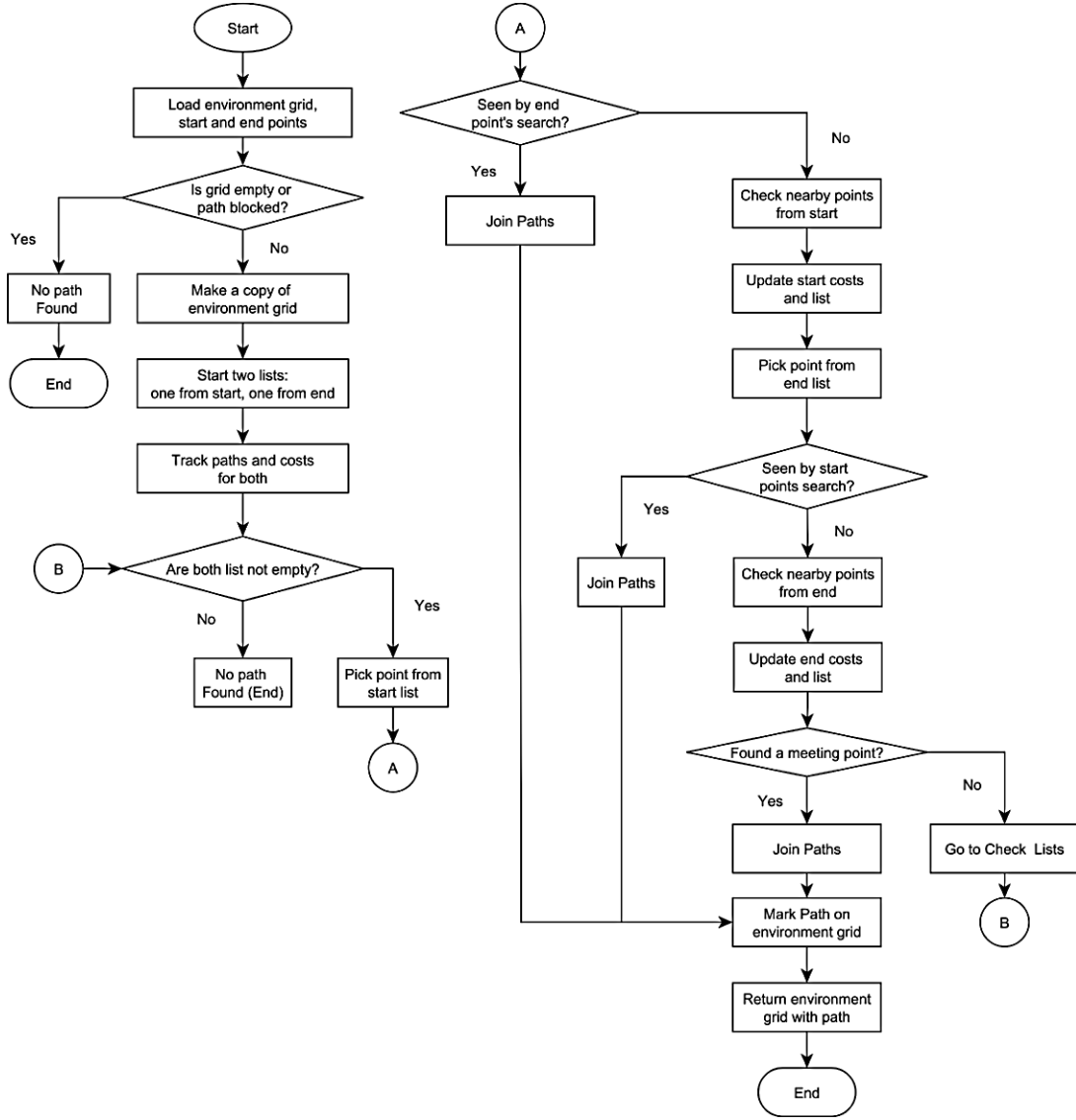


Fig. 3: Bidirectional Search Flowchart

2.2 Literature Review

This section reviews recent studies on map generation with semantic segmentation and route planning algorithms for robotics and different tasks.

The work by Gomroki, Hasanlou, Chanussot, and Hong, "UNet-GCViT: A UNet-Based Framework with Global Context Vision Transformer Blocks for Building Damage Detection" (2025), unveils

UNet-GCViT, a deep learning model for detecting building damage in post-disaster satellite imagery, crucial for fast disaster response. It adds Vision Transformer blocks to U-Net, which avoids problems like registration errors that come up with traditional methods. The model gets the data ready, trains it, makes damage maps, and checks how accurate they are. It has been tested on the 2023 Turkey earthquake, the 2021 Bata explosion, and the 2011 Haiti earthquake, and it is more than 96% accurate, which is better than other methods. Its design make it able to deal with images that contains complicated damage patterns. UNet-GCViT is a tool for real-time disaster recovery that can grow as needed. “Fig. 4” visualizes Turkey earthquake dataset where (a) represent RGB image and (b) is the ground truth (Gomroki, Hasanlou, et al., 2025).

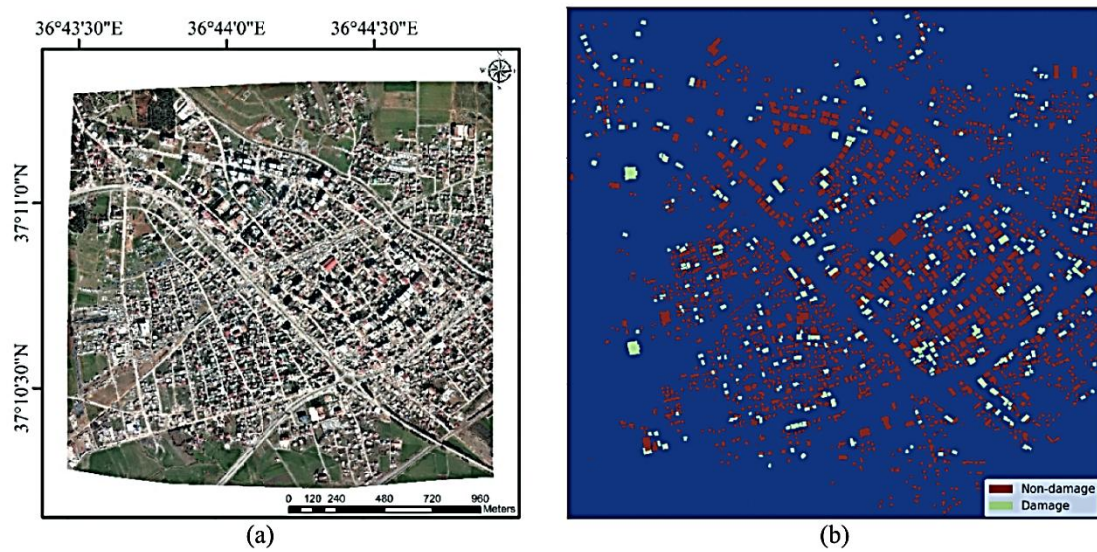


Fig. 4: A datapoint from turkey earthquake dataset (Gomroki, Hasanlou, et al., 2025)

The study by Bhardwaj, Nagabhooshanam, Singh, Selvalakshmi, Angadi, Shargunam, Guha, Singh, and Rajaram, "Enhanced Satellite Imagery Analysis for Post-Disaster Building Damage Assessment Using Integrated ResNet-U-Net Model" (2025), introduces a ResNet-U-Net ensemble model for fast, accurate building damage segmentation in post-

disaster satellite imagery. The model combine ResNet's feature extraction with U-Net's localization and attention mechanisms, it achieves 96.91% accuracy on unbalanced data and 89.35% on balanced data, exceeding other methods. Using post-Hurricane Iota images, the efficient model excels at spotting damage patterns. This model provides a tool for timely disaster recovery (Bhardwaj, Nagabhooshanam, et al., 2025).

The work by Feng, Qiu, Wen, Zhang, Yang, Lyu, Liu, and Fang, "U3UNet: An Accurate and Reliable Segmentation Model for Forest Fire Monitoring Based on UAV Vision" (2025), unveils U3UNet, a deep learning model for real-time forest fire monitoring via UAV imagery, crucial for early fire detection. Unlike limited traditional methods, U3UNet uses a nested U-shaped structure to fuse multi-scale features, ensuring precise fire source mapping. Tested on Unreal Engine and real fire scenes, it scores 71.44% in static and 80.53% in dynamic scenarios, outperforming models like U-Net and DeeplabV3+. Its lightweight design enables edge device use. U3UNet is a scalable model good for fire response and environmental protection (Feng, Qiu, et al., 2025).

The result of Li, He, Xu, Deng, Li, and Wei, "Detection and Segmentation of Loess Landslides via Satellite Images: A Two-Phase Framework" (2022), presents a two-phase framework for detecting loess landslides in satellite images, key for disaster risk assessment. Phase I uses Faster-RCNN to pinpoint landslides with bounding boxes. Phase II crops images and employs boundary detection with an upgraded U-Net, using inception blocks for accurate segmentation. The model was tested on 150 northern China landslides; it surpasses standard U-Net and other methods. This model is suitable to automates fast landslide detection task. "Fig. 5" illustrates the architecture of two phase model (Li, He, et al., 2022).

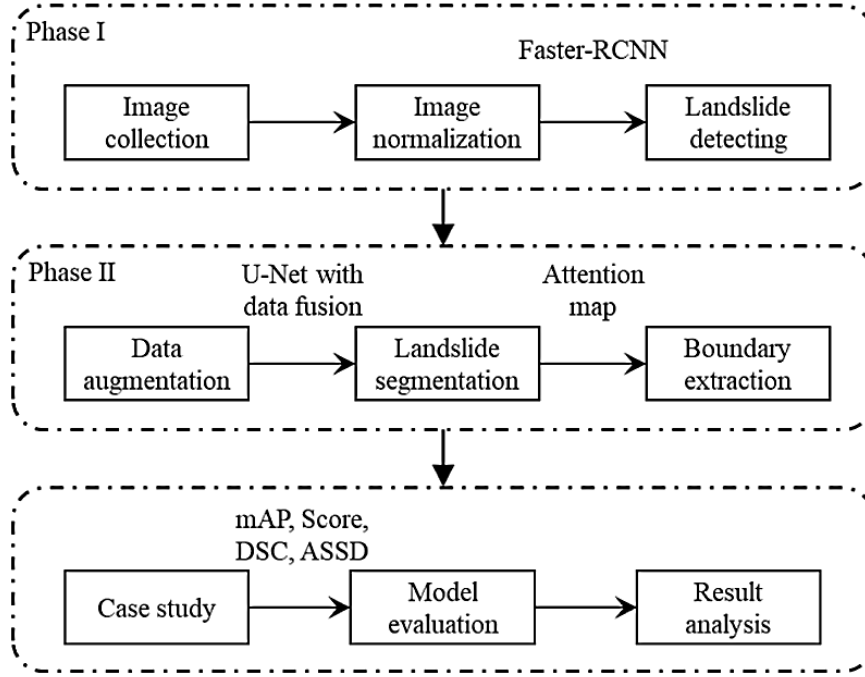


Fig. 5: Architecture of two phase model (Li, He, et al., 2022)

In this work "Detection of Flooded Regions from Satellite Images Using Modified UNET," Jaisakthi, Dhanya, and Kumar (2021) present a modified U-Net model for segmenting flood-affected areas in satellite datasets, crucial for disaster response. Using the MediaEval 2017 dataset, the model preprocesses images and achieves an IoU of 99.46% and 99.41% accuracy, surpassing traditional methods like Mean Shift clustering. Its efficient design ensures fast, precise flood detection. Further improvements could enhance adaptability to many datasets (Jaisakthi, Dhanya, et al., 2021).

A study by Brand and Manandhar, "Semantic Segmentation of Burned Areas in Satellite Images Using a U-Net-Based Convolutional Neural Network" (2021), introduces a U-Net variant for mapping burned areas in Sentinel-2 satellite imagery, surpassing traditional methods needing manual feature engineering. The model compares local and global

models in Indonesia and Central Africa and was tested in two stages. With 94% accuracy, the local model in Indonesia achieves a macro-F1 of 92%, while the global model achieves a macro-F1 of 91%. Both achieve 84% accuracy and 82% macro-F1 in Central Africa. “Fig. 6” shows Segmentation of burned areas in Central Africa (Brand, Manandhar, et al., 2021).

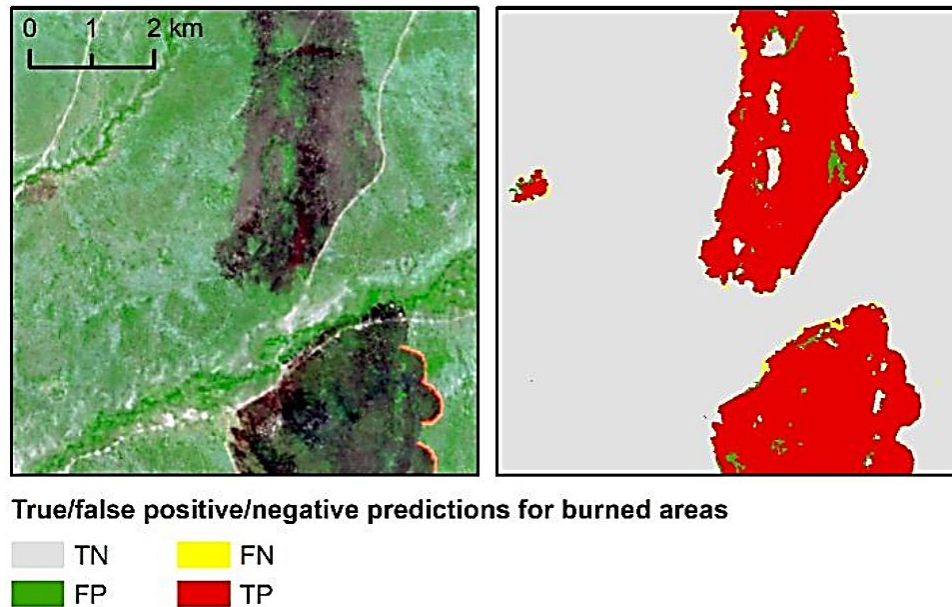


Fig. 6: Segmentation of burned areas in Central Africa (Brand, Manandhar, et al., 2021)

The research conducted by Hakim and Tsai, "Deep Learning-Based Land Cover Extraction from Very-High-Resolution Satellite Imagery for Assisting Large-Scale Topographic Map Production" (2025), presents a deep learning model to speed up topographic map creation in Indonesia, suitable for development planning. This model works on extracting land cover from high-resolution satellite imagery, surpassing other methods limited to 13,000 km²/year. The model results reaches a Kappa value of 0.81 and 86% accuracy, classifying complex land cover types precisely. This model is a great tool for efficient mapping and national growth (Hakim, Tsai, et al., 2025).

The research done by Cui, Sun, Chen, Li, and Zhang, "Deep Learning-Based Urban Tree Species Mapping with High-Resolution Pléiades Imagery in Nanjing, China" (2025), introduces a deep learning model for mapping urban tree species in Nanjing, vital for tackling urban heat and flooding. Using high-resolution Pléiades imagery and field-verified data of eight tree species, it employs transfer learning with segmentation networks (DeeplabV3+, UNet, HRNet, PSPNet) for accurate, efficient classification. "Fig. 7" clarifies Classification accuracy results found by each of four deep learning models for each tree species (Cui, Sun, et al., 2025).

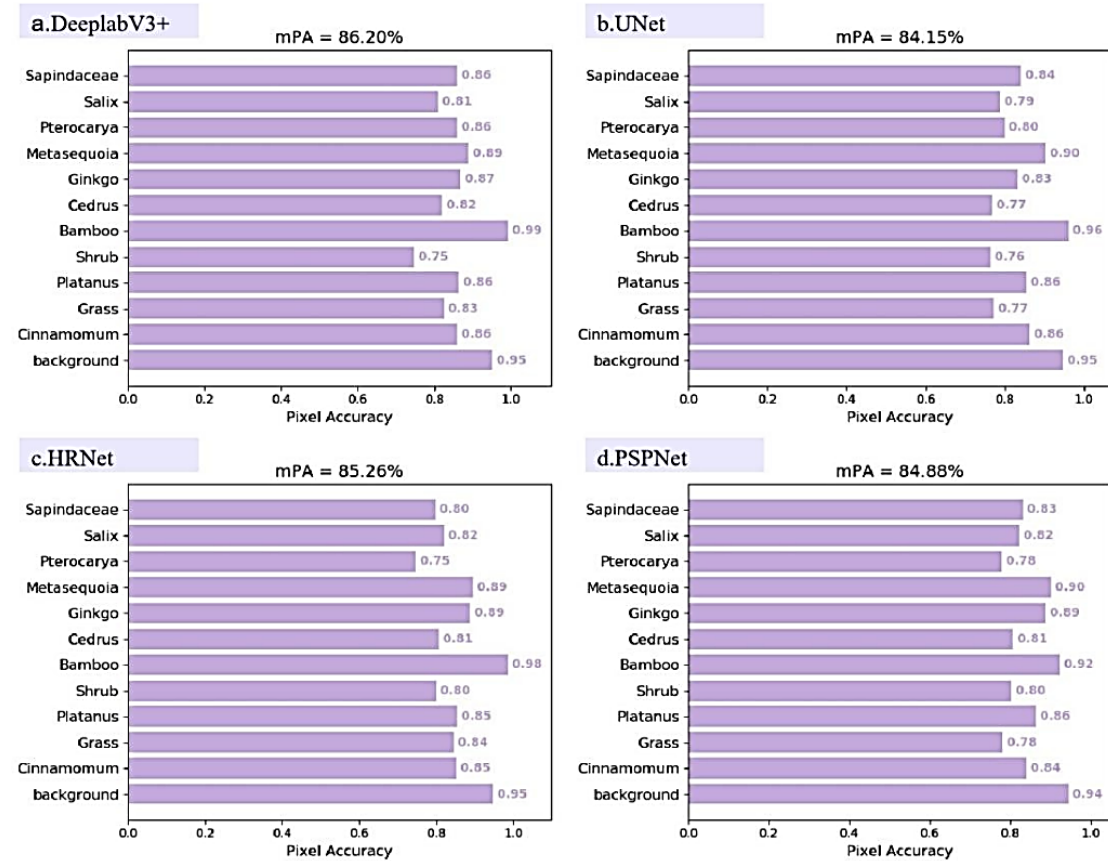


Fig. 7: Classification accuracy results for each tree species in each model (Cui, Sun, et al., 2025)

The work by Ragunthar, Reddy, and Pavan, "Semantic Segmentation of Land Cover Dataset" (2025), presents a U-Net-based

approach for accurate land cover segmentation in satellite imagery, essential for urban planning and environmental monitoring. It tackles challenges like varying scales and weather, achieving reliable feature delineation across multiple datasets. The streamlined U-Net design ensures efficient processing for large-scale remote sensing. This framework is a precise tool for sustainable land management. “Fig. 8” represents model segmentation result (Ragunthar, Reddy, et al., 2025).

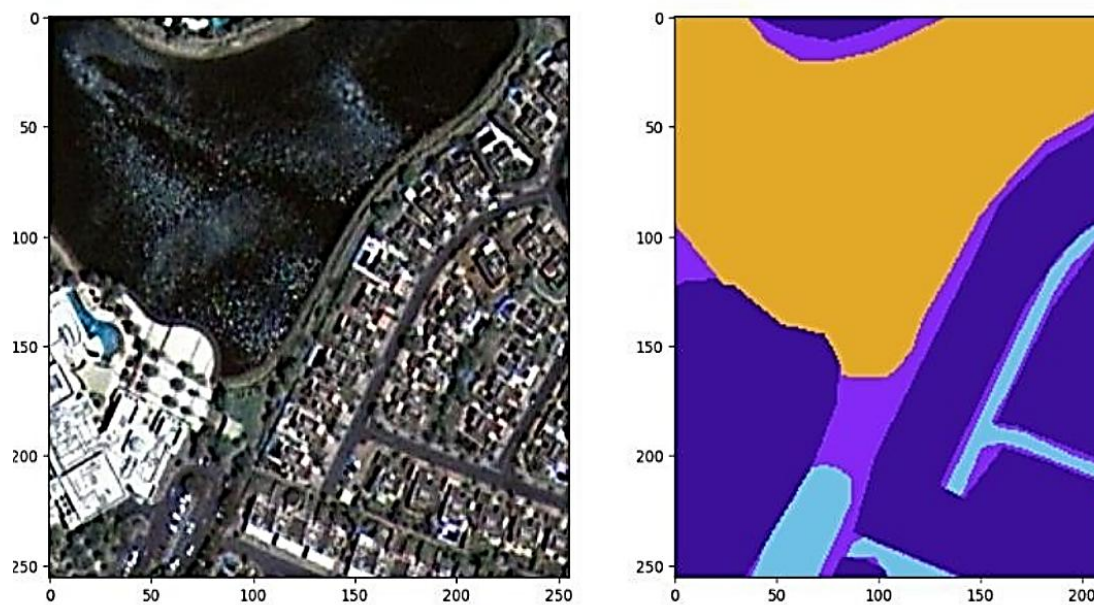


Fig. 8: Model segmentation result (Ragunthar, Reddy, et al., 2025)

The study by Xu, Chen, Niu, Fan, Lu, and Feng, "Boosting Urban Openspace Mapping with the Enhancement Feature Fusion of Object Geometry Prior Information from Vision Foundation Model" (2025), introduces UOSAM, a deep learning model for mapping urban open spaces using high-resolution remote sensing imagery, crucial for urban planning. UOSAM leverages the Segment Anything Model for accurate object segmentation, tackling complex urban settings. A pyramid transformer captures multi-scale features, while a feature fusion module boosts precision. Tested on the UOCTC dataset from ten Chinese cities, it

outperforms other methods. The efficient design suits large-scale use. UOSAM is a versatile tool for sustainable urban management. “Fig. 9” visualizes the UOSAM model structure (Xu, Chen, et al., 2025).

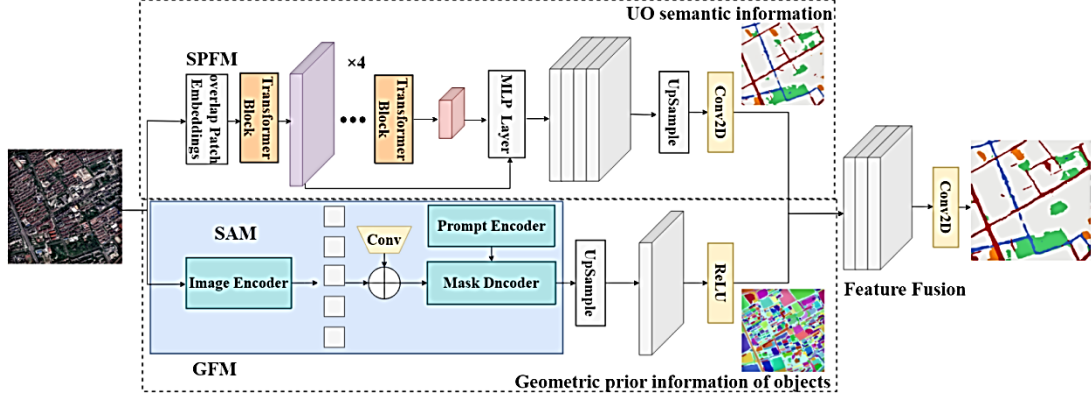


Fig. 9: UOSAM model structure (Xu, Chen, et al., 2025)

The work of Sierra, Ramo, Padilla, and Cobo, "Optimizing Deep Neural Networks for High-Resolution Land Cover Classification Through Data Augmentation" (2025), presents a deep learning method for land cover classification using 25 cm resolution imagery from Cantabria, Spain, vital for environmental monitoring. It tests eight augmentation techniques across U-Net, DeepLabv3+, FCN, and PSPNet, creating 19 training sets and 72 models, boosting performance by up to 30%. The best model, DeepLabV3+ with flip, contrast, and brightness tweaks, achieves 0.89 accuracy and 0.78 IoU. It maps land cover for 2014, 2017, and 2019, hitting 87.2% accuracy. The efficient design is scalable. This method is a practical tool for precise land cover mapping (Sierra, Ramo, et al., 2025).

This research "A Framework to Estimate Generating Capacities of PV Systems Using Satellite Imagery Segmentation," done by Jurakuziev, Jumaboev, and Lee (2023) introduce a deep learning model to estimate solar energy from satellite images. The model calculates PV panel areas on

rooftops and ground, using UNet which which use an EfficientNetB7 as encoder. It reaches a Dice Coefficient of 0.9982 and a MAPE of 3.5%. The model converts pixel data to square meters, considering panel slope and efficiency. It outperforms UNet++, FPN, PSPNet, and DeepLabV3. The efficient design suits large-scale use. More improvements could improve diverse climate handling. This method is a cost-effective tool for global solar energy planning. “Fig. 10” shows the predicted PV area and energy and the ground truth (Jurakuziev, Jumaboev, et al., 2023).

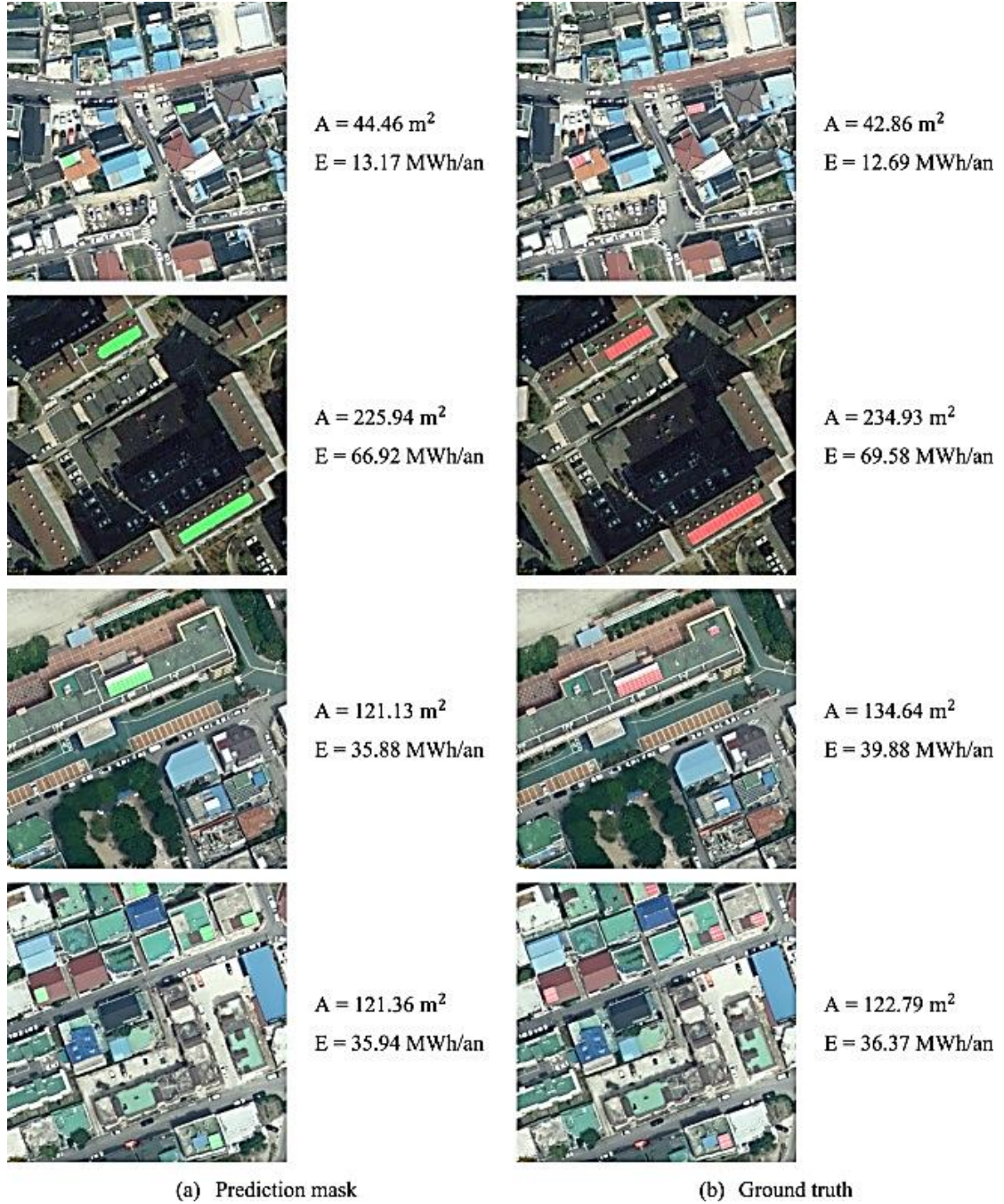


Fig. 10: Predicted PV area and energy and the ground truth (Jurakuziev, et al., 2023)

The research done by Safarov, Temurbek, Jamoljon, Temur, Chedjou, Abdusalomov, and Cho, titled "Improved Agricultural Field Segmentation in Satellite Imagery Using TL-ResUNet Architecture" (2022), unveils TL-ResUNet, a deep learning model for precise

agricultural field segmentation in satellite images, crucial for food security and land use. The TL-ResUNet model uses transfer learning, residual networks, and UNet's encoder-decoder to improve feature extraction using pre-trained ImageNet weights. The model results reaches an IoU of 0.81 on the DeepGlobe dataset, exceeding other segmentation models. This model is suitable for high accuracy agriculture tasks. This model is a key tool for sustainable agriculture (Safarov, Temurbek, et al., 2022).

The work done by Huerta, Yépez, Lozano-García, Guerra Cobián, Ferriño Fierro, de León Gómez, Cavazos González, and Vargas-Martínez, "Mapping Urban Green Spaces at the Metropolitan Level Using Very High Resolution Satellite Imagery and Deep Learning Techniques for Semantic Segmentation" (2021), unveils a deep learning approach to identify urban green spaces (UGSs) in fast-growing cities, crucial for healthy ecosystems and city planning. It uses high-resolution satellite images with a U-Net model and two neural network encoders to map UGS shapes accurately. The top model scores a Dice coefficient of 0.57, IoU of 0.75, and 97% accuracy, adeptly recognizing varied green space patterns. Its lean design builds a unified UGS database by city area, aiding planners. This method provides a practical, large-scale solution for UGS oversight (Huerta, Yépez, et al., 2021).

This research conducted by Wang, Gao, Zhao, Xu, and Ma, "A Boundary-Enhanced Semantic Segmentation Model for Buildings" (2025), presents BE-SegModel, a deep learning model for precise building extraction from high-resolution satellite imagery, key for surveying and mapping. It improves accuracy by extracting building masks and boundary details together, using heatmaps and feature maps. Tested on xBD, WHU Global, and WHU East Asia datasets, BE-SegModel surpasses models like

ResNet34 and Deeplabv3+, improving accuracy by 6.9%. Its efficient design suits large-scale use. This model is a strong tool for automated building mapping. “Fig. 11” demonstrates BE-SegModel model architecture (Wang, Gao, et al., 2025).

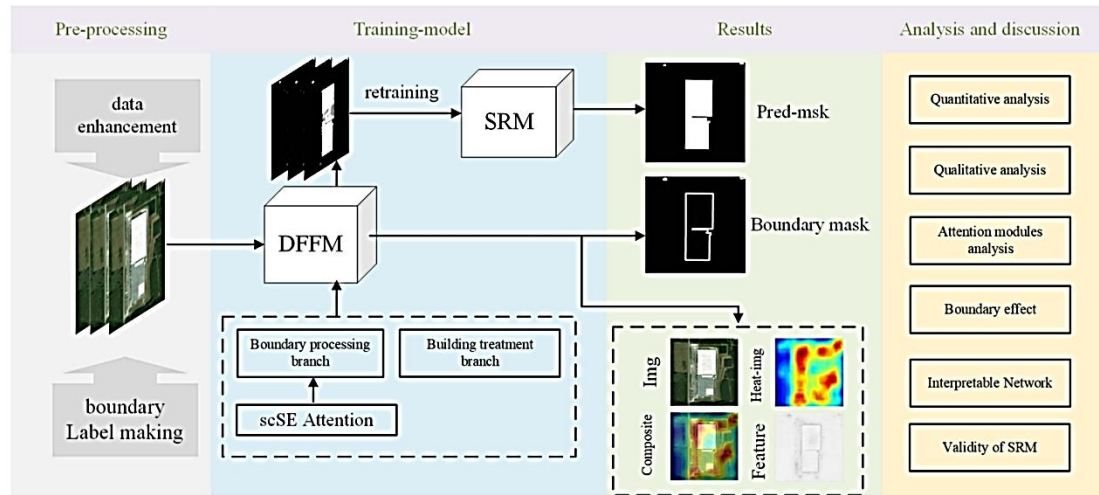


Fig. 11: BE-SegModel model architecture (Wang, Gao, et al., 2025)

A paper by Ismael and Sadeq, "LoD2 Building Reconstruction from Stereo Satellite Imagery Using Deep Learning and Model-Driven Approach" (2025), presents a hybrid framework for LoD2 building reconstruction from stereo-satellite imagery, key for 3D urban modeling. It blends U-Net, FCN, and Mask R-CNN for precise boundary segmentation with digital surface models for geometric accuracy, refined by OpenStreetMap and graph cut optimization. Tested on GeoEye-1 imagery of Erbil City, it achieves a mean absolute error of 0.70 m in open areas and 1.0 m in occluded ones, with reliable roof classification. The efficient design ensures scalability. This model is a powerful tool for urban planning. “Fig. 12” illustrate the model development workflow (Ismael, Sadeq, et al., 2025).

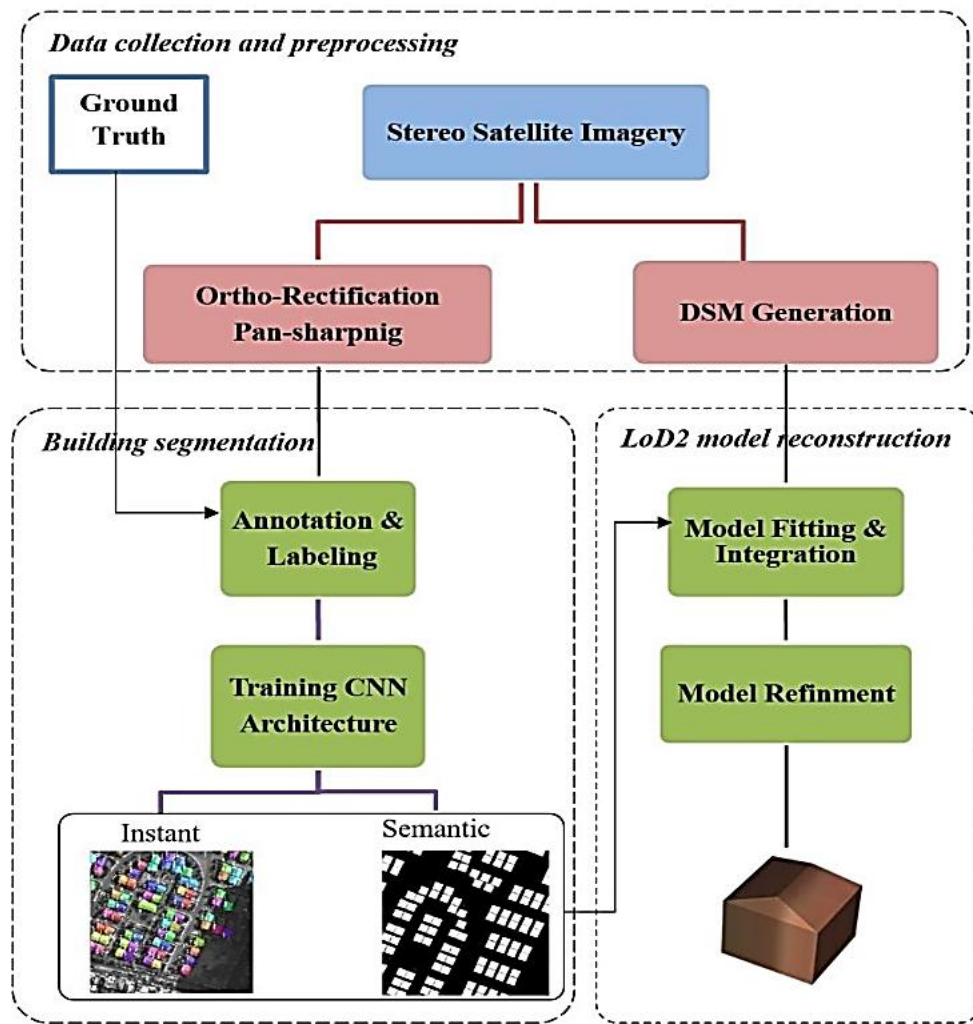


Figure 12: Model development workflow (Ismael, Sadeq, et al., 2025)

This research titled "Automatic Building Extraction on Satellite Images Using Unet and ResNet50," Alsabhan and Alotaiby (2022) propose an AI model to automate building segmentation from high-resolution satellite imagery, good for urban planning in fast growing cities. By integrating the U-Net architecture with a ResNet50 encoder, the model segments buildings in the Massachusetts dataset. It reaches an IoU of 82.2%, an F1 score of 0.9, and 90% accuracy, Suitable in complex urban areas. The efficient design ensures accurate results with low computational cost. more improvements could increase the adaptability to many datasets (Alsabhan, Alotaiby, et al., 2022).

The paper done by Dey, Chaudhuri, Banerjee, and Bhattacharya, "Dual-Path Morph-UNet for Road and Building Segmentation From Satellite Images" (2021), introduces Dual-Path Morph-UNet, a framework for segmenting roads and buildings in satellite imagery. It uses 2D morphological operations in a U-Net model to optimize structural element sizes and improve segmentation. The dual-path encoder-decoder with residual and dense pathways produce high accuracy with 10 times fewer parameters comparing it to top methods. Tested on the Massachusetts dataset, it accurately identifies roads and buildings. The lightweight design increase efficiency. This model offers a precise, efficient solution for geospatial tasks (Dey, Chaudhuri, et al., 2021).

The research conducted by Abdollahi, Pradhan, and Alamri, titled "An Ensemble Architecture of Deep Convolutional Segnet and Unet Networks for Building Semantic Segmentation from High-Resolution Aerial Images" (2020), unveils Seg-UNet, a deep learning model for precise building segmentation in aerial imagery, vital for geospatial database updates. The Seg-UNet model architecture merges Segnet and UNet strengths to tackle obstacles like shadows and trees. Tested on the Massachusetts dataset, it achieves 92.73% accuracy, surpassing FCN, Segnet, and UNet. The design complexity give the model the ability to accurate building delineation in urban scenes. "Fig. 13" represents Seg-UNet model architecture (Abdollahi, Pradhan, et al., 2020).

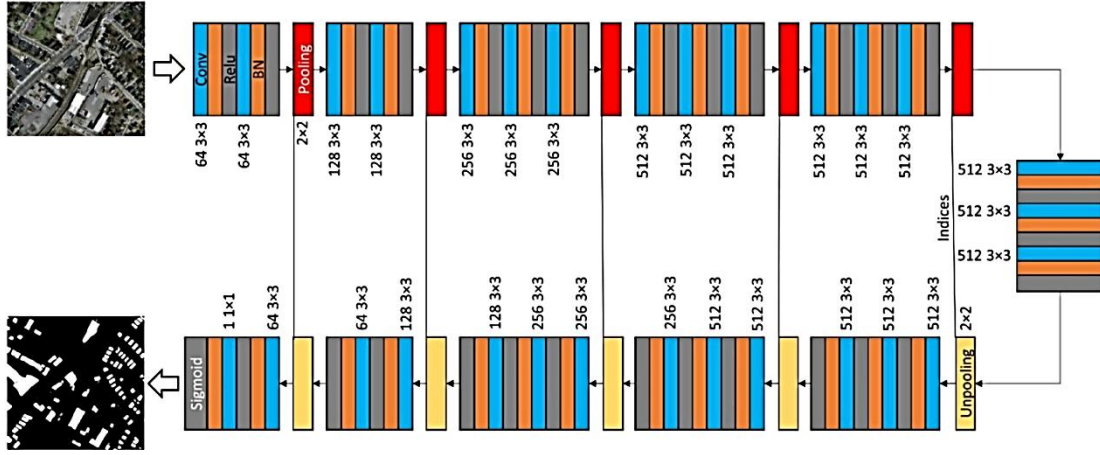


Fig. 13: Seg-UNet model architecture (Abdollahi, Pradhan, et al., 2020)

The study by Yang, Wei, Liu, Yang, He, and Li, "Segmentation of Urban Green Space and Water Body Based on High-Resolution Remote Sensing Images" (2025), introduces a deep learning approach to segment urban green spaces and water bodies from high-resolution satellite images, vital for urban planning and ecological protection. Using the novel NEU-RS1 dataset, it unifies diverse green and water features. A data augmentation strategy and an adaptive threshold algorithm with Jaccard distance and differential evolution enhance segmentation. Tested with models like UNet and DeepLabv3+ on NEU-RS1, it achieves 85.37% mIoU and 91.18% mAcc. The efficient design scales well. This model is a powerful tool for sustainable urban mapping (Yang, Wei, et al., 2025).

A paper done by Anitha, Chitra, and Meenaragavi, "Segmentation of Water Bodies from Satellite Images Using Spatial-Channel Attention UNet" (2024), presents a deep learning framework for accurate water body segmentation in Sentinel-2 satellite imagery, designed for environmental monitoring. The research compares UNet, Deeplabs, and SegNet models. The Spatial-Channel Attention UNet—boosted by its built-in attention mechanisms—hits 96.7% accuracy, outperforming other models on key

metrics such as mIoU and cross-entropy loss. It produces precise segmentation even on tricky water imagery, offering a scalable fix for real-world water management challenges. “Fig. 14” demonstrates model diagram (Anitha et al., 2024).

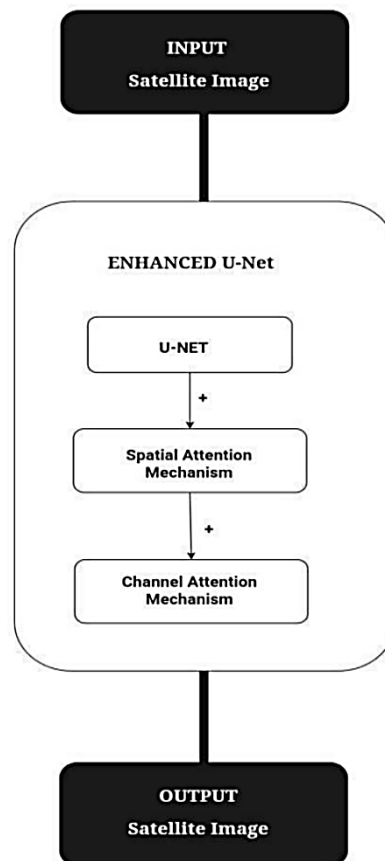


Fig. 14: Spatial-Channel Attention UNet Model diagram (Anitha et al., 2024)

The research by Jonnala, Bheemana, Prakash, Bansal, Jain, Pandey, Faruque, and Al-mugren, "DSIA U-Net: Deep Shallow Interaction with Attention Mechanism UNet for Remote Sensing Satellite Images" (2025), presents DSIA U-Net, a lightweight deep learning model for water body segmentation in high-resolution satellite imagery. It uses a U-Net framework with a deep-shallow interaction and attention module to make edge detection and feature integration better. It captures both local and

global details better than standard models. The model is 95% accurate, which is better than other methods. Its slim shape makes it good for monitoring the environment and planning cities. DSIA U-Net is a reliable, scalable tool for use in the real world. (Jonnala, Bheemana, et al., 2025).

The result of Adhikary, Singh, Dey, Ghosh, and Sahu, "Satellite Image Segmentation Using Deep Learning" (2025), introduces a deep learning framework using a modified U-Net to segment satellite imagery, crucial for urban planning and environmental monitoring. It classifies six land-use types in a Dubai dataset, using patch creation, normalization, and data augmentation for accuracy. Combining categorical focal loss and dice loss, it tackles class imbalance, achieving robust segmentation. The efficient model excels in diverse applications. This approach is a scalable tool for real-time geospatial analysis (Adhikary, Singh, et al., 2025).

The work done by Wang, Fan, Jiang, Yan, and Yang, "EDFF-Unet: An Improved Unet-Based Method for Cloud and Cloud Shadow Segmentation in Remote Sensing Images" (2025), introduces EDFF-Unet, a deep learning model for segmenting clouds and shadows in remote sensing imagery, key for climate monitoring and surface analysis. Built on U-Net, it combines a semantic sub-network with attention and spatial pyramid pooling to focus on key features, and an edge sub-network for clear contours. Tested on the GF1_WHU dataset, it achieves a 92.87% mIoU, beating others by 0.67%. The efficient design handles complex data well. This model is a reliable tool for remote sensing tasks. "Fig. 15" illustrate model architecture (Wang, Fan, et al., 2025).

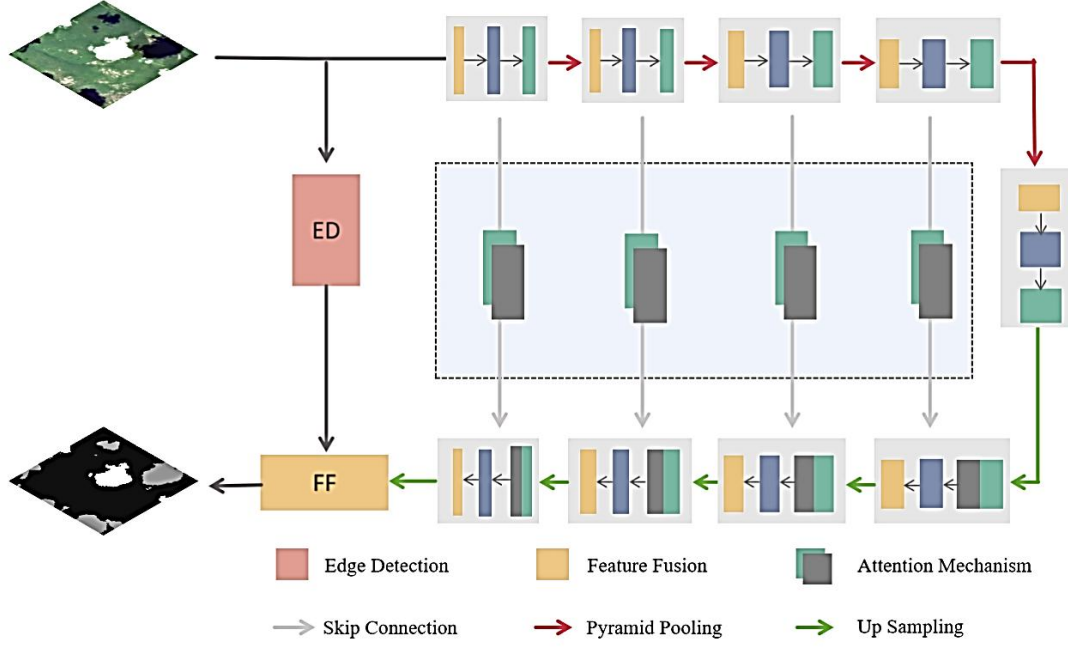


Fig. 15: EDFF-Unet model architecture (Wang, Fan, et al., 2025)

A study titled "Satellite Image Segmentation via Image Quality Enhancement and Modified Unet Architecture," Pathak, Kamani, and Priyanka (2024) propose a framework for segmenting high-resolution satellite images, key for urban planning and environmental monitoring. They use an enhanced ESRGAN to improve image quality, avoiding pixelation. A modified U-Net with a ResNet50 encoder and Tversky loss achieves 94% accuracy and 83% mIoU, accurately identifying buildings, forests, and roads. The efficient design handles large and complex datasets well. More improvements could enhance complex scene segmentation. (Pathak, Kamani, et al., 2024).

In this study "Semantic Segmentation of Satellite Images Using Deep-Unet," Singh and Nongmeikapam (2023) introduce Deep-UNet, an modified U-Net model optimized for satellite image segmentation, important for urban planning, agriculture, and environmental monitoring. Deep-UNet use FAAGKFCM and SLIC Superpixel preprocessing to

improve classification of features like roads, buildings, and vegetation. It reaches a high mIoU of 89.51% and global accuracy of 90.6%, surpassing SegNet and standard UNet. The model design ensures high-precision segmentation of complex satellite data. More improvement could make the model more flexib to diverse datasets and challenging scenarios (Singh, Nongmeikapam, et al., 2023).

A study done by Wang, Yang, Liang, Zheng, Yin, Tian, and Cui in "Semantic Segmentation and Analysis on Sensitive Parameters of Forest Fire Smoke Using Smoke-Unet and Landsat-8 Imagery" (2022) reveal Smoke-UNet, a deep learning model for detecting early forest fire smoke in Landsat-8 satellite imagery, vital for wildfire monitoring. Smoke-UNet is an improved U-Net model with attention mechanisms and residual blocks it uses RGB, SWIR2, and AOD bands for high accuracy smoke segmentation. The model results achieves 3.1% higher pixel accuracy than standard U-Net on a custom complex dataset. The model design ensures fast detection. This model ia scalable and suitable for real-time wildfire warning systems (Wang, Yang, et al., 2022).

In their research titled "M-UNet: Modified U-Net Segmentation Framework with Satellite Imagery," Soni et al. (2020) introduce M-UNet, an advanced U-Net model for satellite image segmentation. M-UNet uses a deeper architecture with DenseNet in downsampling to reuse feature maps and long-range skip connections To maintain fine details. It achieves an IOU of 73.02% and OA of 96.02%, outperforming standard methods. The compact design ensures efficiency while excelling at complex pattern recognition. More enhancement could enhance diverse data handling, making M-UNet a reliable tool for real-world remote sensing applications (Soni, Koner, et al., 2020).

The research by Jin, Li, Lv, and Zhang, "An Optimized Path Planning Approach for Automatic Parking Using Hybrid A* Bidirectional Search" (2024), unveils a bidirectional hybrid A* algorithm for automatic parking in tight, obstacle-heavy spaces. It uses rectangular expansion for safe, collision-free navigation and adds a safety cost to keep clear of obstacles. A bidirectional search with a cohesive cost speeds up planning, cutting search time and nodes. Simulations for vertical and parallel parking show better safety, speed, and stability than other methods. The efficient design suits urban constraints. This model is a reliable tool for smart parking solutions. “Fig. 16” represents algorithm flowchart (Jin, Li, et al., 2024).

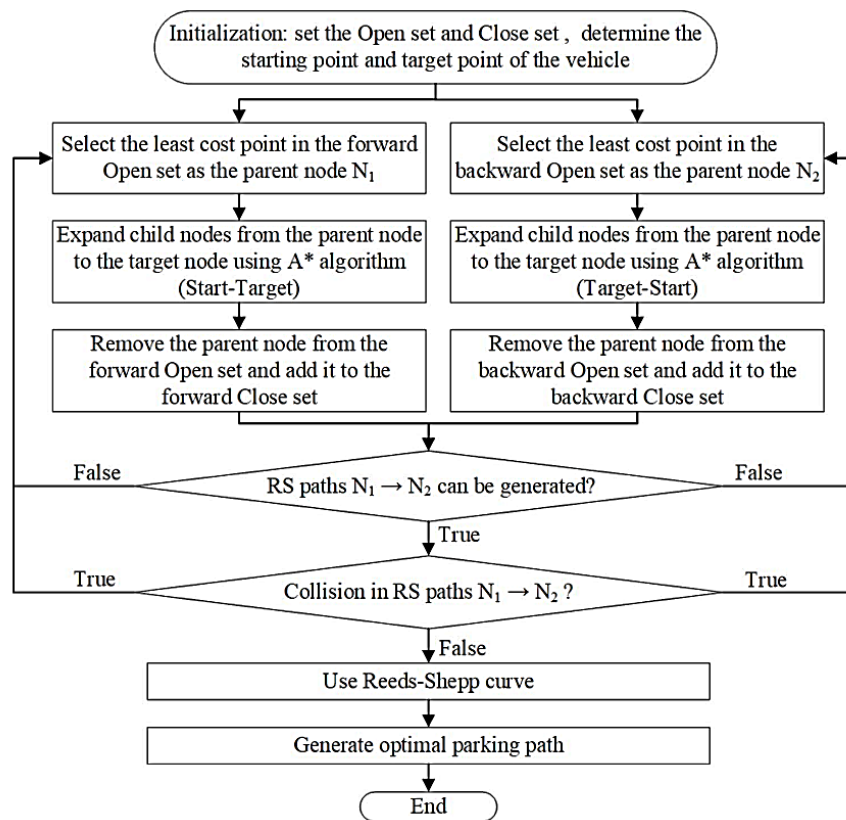


Figure 16: Hybrid A* Bidirectional Search flowchart (Jin, Li, et al., 2024)

The work by Zhang, Chen, and Zhang, "AGV Path Planning Based on Improved A-Star Algorithm" (2024), presents an enhanced A-star

algorithm for optimizing Automated Guided Vehicle (AGV) paths, addressing issues like excessive turns and obstacle proximity. It uses a refined cost function with dynamic weights to reduce turns and integrates an artificial potential field with a penalty term for safer paths. Simulations show fewer turning points, smoother routes, and better obstacle avoidance than traditional A-star. The efficient design suits complex environments. This model is a reliable tool for energy-efficient AGV navigation (Zhang, Chen, et al., 2024).

The work by Lin, "Dijkstra and A* Algorithms in Automated Vehicle Driving" (2024), examines Dijkstra and A* algorithms for path planning in autonomous vehicles, crucial for safe navigation. It reviews their history, strengths, and limits. Dijkstra ensures optimal paths but is slow, while A* is faster with heuristics, though less optimal at times. Compared for intelligent driving, both excel in different scenarios. The efficient analysis highlights their role in dynamic urban settings. This study offers a solid benchmark for next-generation autonomous driving solutions (Lin, et al., 2024).

The study by Wang, Huang, Li, Jiang, Li, and Jiang, "AGV Path Planning Algorithm Based on Improved Dijkstra Algorithm" (2024), unveils a dynamic heuristic A* algorithm for optimizing Automated Guided Vehicle (AGV) paths in smart warehouses, overcoming static heuristic limits. It adjusts its heuristic function in real-time for dynamic settings, outperforming traditional A* and Dijkstra methods. Experiments show it cuts path costs by 20%, reduces expanded nodes by 15%, and speeds up runtime by 7.5 milliseconds. The efficient design ensures high-quality, adaptable paths. This model is a responsive tool for efficient warehouse navigation. (Wang, Huang, et al., 2024).

The research by He, Fan, Wu, and Zhang, "Research on Path Planning Based on Bidirectional A* Algorithm" (2024), presents a bidirectional A* algorithm for efficient path planning of unmanned mining vehicles in open-pit mines. It adds a gradient factor to the cost function for obstacle avoidance and a weighted heuristic blending Euclidean and Manhattan distances for accurate routing. Exploring from both start and end points, it speeds up searches. MATLAB simulations show faster, higher-quality paths than standard methods. The lean design suits rugged terrains. This model is a fast, precise tool for autonomous mining navigation (He, Fan, et al., 2024).

The work by Xia, Shen, Zheng, Zhang, Bai, Hu, and Li, "A Bidirectional-A-Star-Based Ant Colony Optimization Algorithm for Big-Data-Driven Taxi Route Recommendation" (2023), introduces BiA*-ACO, a hybrid algorithm for fast taxi route planning in urban areas, cutting fuel use and congestion. It merges Bidirectional A-star's cost estimation with Ant Colony Optimization to improve search efficiency and uses cycle-optimal routes for faster convergence. Tested with real taxi GPS data, it boosts efficiency by 47.05% for small datasets and 49.81% for large ones, outperforming ACO, Dijkstra, and others, while shortening routes compared to A-star, Acyclic, and Gurobi. The lean design handles big data well. This model is a smart tool for eco-friendly taxi navigation. "Fig. 17" shows the route recommendation system (Xia, Shen, et al., 2023).

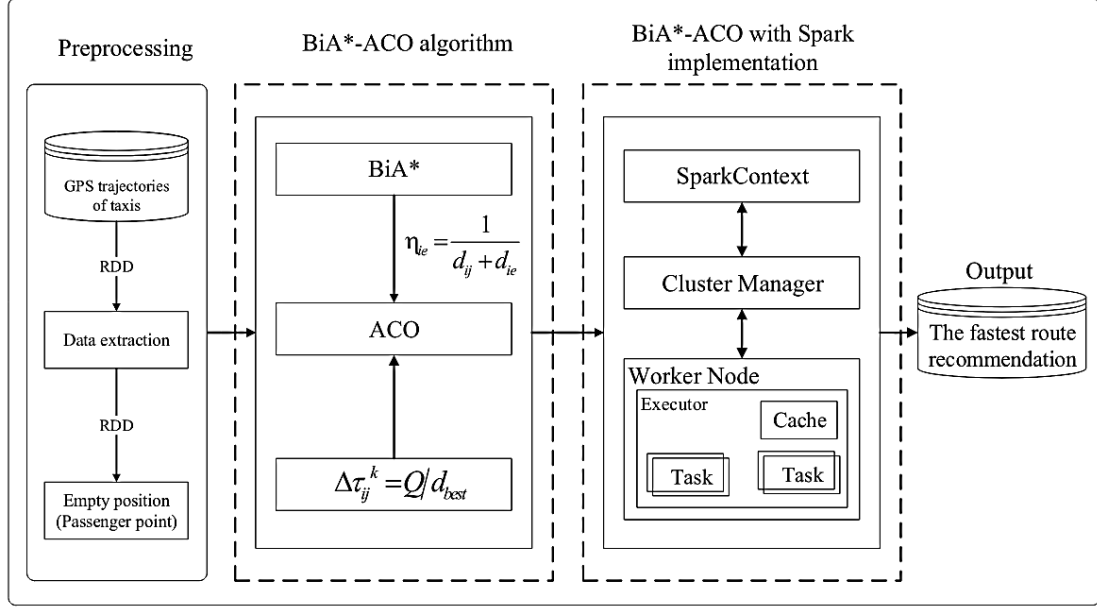


Fig. 17: BiA*-ACO route recommendation system (Xia, Shen, et al., 2023)

The study by Sundarraj, Reddy, Basam, Kumar, Flammini, and Natarajan, "Route Planning for an Autonomous Robotic Vehicle Employing a Weight-Controlled Particle Swarm-Optimized Dijkstra Algorithm" (2023), presents WCPsODA, a route planning model for autonomous robotic vehicles to navigate crowded spaces safely, ideal for logistics and exploration. It combines Particle Swarm Optimization's weight-adjusted flock-like behavior with Dijkstra's shortest-path algorithm for efficient, secure routes. MATLAB simulations confirm its strong path reliability compared to standard tools. The lean design ensures real-world usability. This model is a powerful tool for safe robotic navigation (Sundarraj, Reddy, et al., 2023).

The work by Praveen Kumar, Hariharan, and Manikandan, "Hybrid Long Short-Term Memory Deep Learning Model and Dijkstra's Algorithm for Fastest Travel Route Recommendation Considering Eco-Routing Factors" (2023), presents a hybrid model blending LSTM and Dijkstra's

algorithm for fast, eco-friendly route planning in urban areas, tackling traffic and emissions. LSTM predicts traffic patterns, while Dijkstra's computes optimal paths, factoring in fuel use and emissions like CO, NO_x, and HC. Compared to shortest-path methods, it cuts fuel by 54.5%, CO by 61.5%, NO_x by 64.2%, and HC by 81.9%. The efficient design suits city traffic systems. This model is a sustainable tool for smart urban mobility. “Fig. 18” show the model flowchart (Praveen Kumar, Hariharan, et al., 2023).

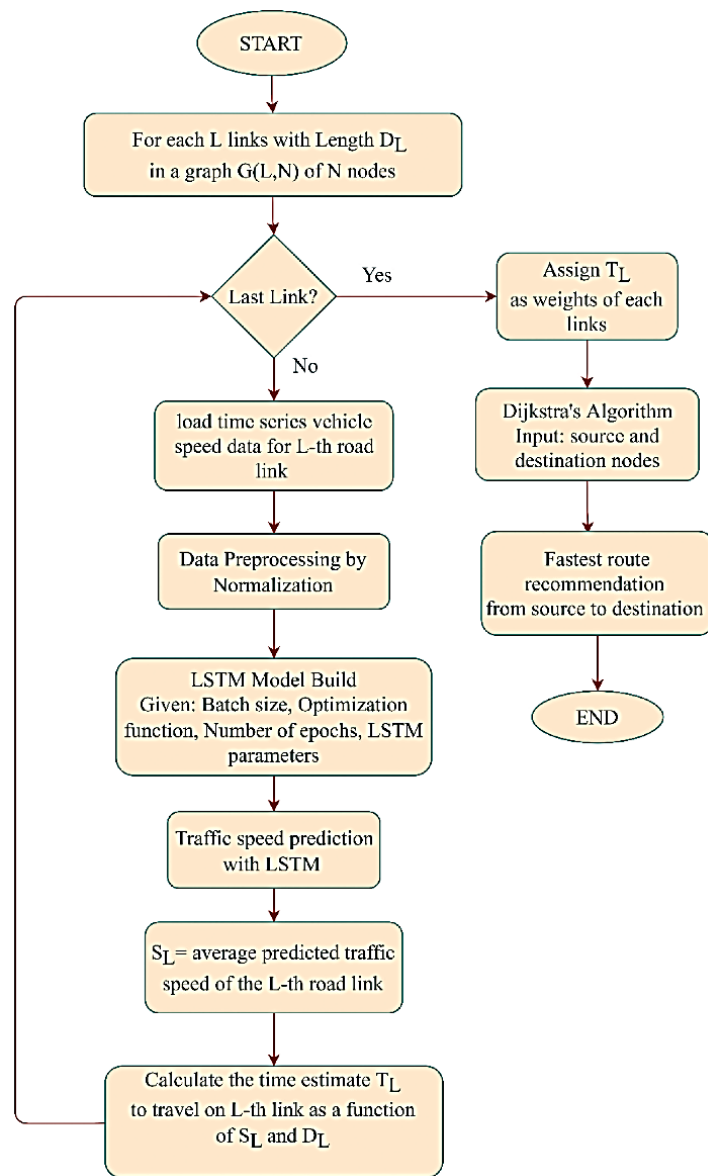


Fig. 18: Hybrid LSTM and Dijkstra's algorithm (Praveen Kumar, Hariharan, et al., 2023)

The study by Chen, Hu, Luo, Wu, Biancardo, Zheng, and Xian, "Intelligent Ship Route Planning via an A* Search Model Enhanced Double-Deep Q-Network" (2025), presents a ship route planning method combining A-star and Double Deep Q-Network (DDQN) for safe, efficient navigation under weather and emission constraints. A convolutional neural network models maritime conditions. The A-DDQN merges A-star's precise pathfinding with DDQN's adaptive learning, using a reward function to optimize routes. It cuts fuel use by 11.93% and emissions by 12.16% compared to standard DDQN. The efficient design suits real-time use. This model is a smart, green tool for maritime routing (Chen, Hu, et al., 2025).

The study by Zhen, Gu, Shi, and Suo, "An Improved A-Star Ship Path-Planning Algorithm Considering Current, Water Depth, and Traffic Separation Rules" (2023), presents an upgraded A-star algorithm for ship path planning, vital for safe maritime navigation. It includes a risk model for currents, water depth, and obstacles to reduce collision risks. A traffic model ensures compliance with separation rules, while a turning model and smoothing method improve path control. Validated in three simulated scenarios, it excels in obstacle avoidance and rule adherence. The efficient design scales well. This model is a strong tool for safe ship routing (Zhen, Gu, et al., 2023).

The research by Zhang, Wen, and Tu, "A Method for Ship Route Planning Fusing the Ant Colony Algorithm and the A* Search Algorithm" (2023), presents a hybrid method for ship route planning using AIS data, vital for smart navigation. It merges A-star's global search with the ant colony algorithm's optimization, refining paths with Bézier curve smoothing for practical use. Tested on a bulk carrier, it delivers shorter,

faster routes than other methods. The efficient design meets maritime needs. This model is a strong tool for safe, intelligent ship routing. “Fig. 19” demonstrates model flowchart (Zhang, Wen, et al., 2023).

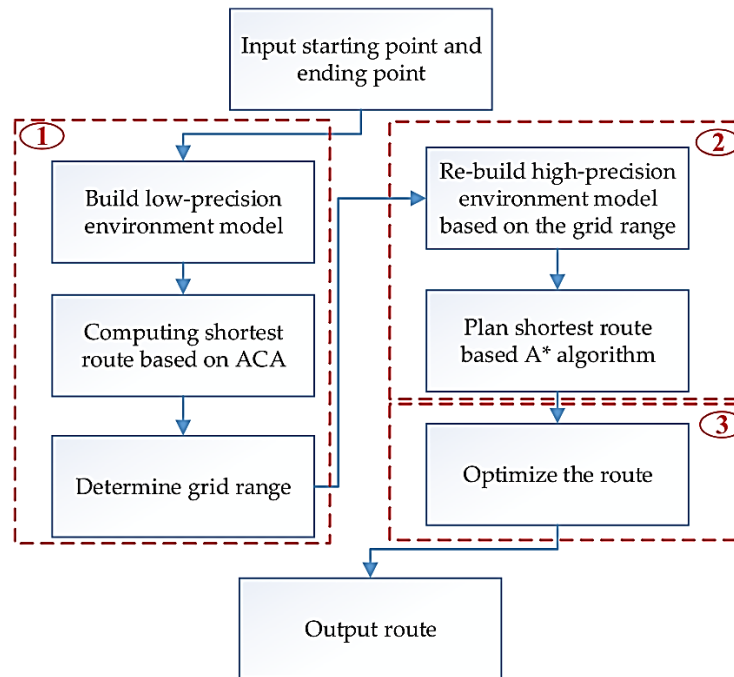


Fig. 19: Ant Colony and the A* Search hybrid model flowchart (Zhang, Wen, et al., 2023)

The research by Miyombo, Liu, Mulenga, Siamulonga, Kabanda, Shaba, Xi, and Ayodeji, "Optimal Path Planning in a Real-World Radioactive Environment: A Comparative Study of A-Star and Dijkstra Algorithms" (2024), presents a real-world path planning experiment at Zambia’s radioactive Slag Dump No. 48 to reduce worker radiation exposure. Using the Radiation Detection Backpack System and Geolocation Application for Radiation Monitoring, it applies Dijkstra and A-star algorithms with gamma dose rates to find safe walking paths. The study tests both algorithms on dose- and distance-weighted graphs, showing effective path optimization. Its efficient design suits hazardous

settings. This approach is a practical tool for safe nuclear navigation (Miyombo, Liu, et al., 2024).

The study by Choi and Lee, "Application of an Advanced Dijkstra Algorithm for Automation Systems of 3D Piping Routing" (2024), unveils an automated pipeline routing model for complex EPC plant projects, crucial for cost and time-saving design. It uses an Advanced Dijkstra Algorithm with digital image processing and 3D equipment models. The algorithm finds the shortest paths, optimized by a Routing Priority Algorithm and avoids collisions with an Interference Avoidance Algorithm. Tested on Korea's 'D' petrochemical plant data, it achieves 98.9% pipe weight accuracy and 84.9% length accuracy. The lean design stores project data for future use. This model is a precise tool for 3D piping design (Choi, Lee, et al., 2024).

The study by Huang, Gao, Guo, Qian, and Chen, "An Adaptive Bidirectional Quick Optimal Rapidly-Exploring Random Tree Algorithm for Path Planning" (2024), presents ABQ-RRT*, an enhanced algorithm for UAV path planning in narrow spaces, improving on RRT*'s slow convergence and weak performance in tight areas. It adjusts the cost function with Euclidean distance and turning angles for smoother routes. Adaptive sampling and local obstacle checks, paired with a repulsion field, ensure efficient, safe paths. Optimized parent node selection boosts path quality. Simulations show it outperforms other RRT variants. The lean design excels in complex settings. This model is a precise tool for UAV navigation (Huang, Gao, et al., 2024).

The work by Ait Ben Mouh, Ouhda, El Mourabit, and Baslam, "A Hybrid Approach of Dijkstra's Algorithm and A* Search, with an Optional

Adaptive Threshold Heuristic" (2023), presents a hybrid Dijkstra-A* algorithm for efficient path planning in robotic crop harvesting, crucial for navigating dynamic fields. It uses Dijkstra's accurate cost mapping and A*'s heuristic speed to find low-cost paths, refined by an optional adaptive heuristic for complex settings. Tests across graph sizes show it outperforms standalone Dijkstra and A* in speed and path quality. The lean design suits real-time farming tasks. This model is a practical tool for precise robotic harvesting (Ait Ben Mouh, Ouhda, et al., 2023).

The study by Fransen and van Eekelen, "Efficient Path Planning for Automated Guided Vehicles Using A* (Astar) Algorithm Incorporating Turning Costs in Search Heuristic" (2021), introduces an improved A-star algorithm for Automated Guided Vehicle (AGV) path planning in production facilities, crucial for saving time and energy. It adds a heuristic that accounts for turning costs, reducing turns compared to traditional methods. The monotone, admissible heuristic ensures faster pathfinding with fewer iterations. Comparative tests show it creates smoother, efficient paths in geometric graphs. The lean design suits industrial use. This model is a practical tool for AGV navigation (Fransen, van Eekelen, et al., 2021).

The work by Parimala, Jafari, Riaz, and Aslam, "Applying the Dijkstra Algorithm to Solve a Linear Diophantine Fuzzy Environment" (2021), adapts the Dijkstra algorithm to find the shortest path in linear Diophantine fuzzy graphs, enhancing fuzzy set applications. It uses Linear Diophantine Fuzzy Numbers for arc weights and a new score function to handle uncertainty. The modified Dijkstra algorithm computes optimal fuzzy paths accurately. Tested on a small telecommunication network, it outperforms existing methods. The efficient design suits uncertain

scenarios. This model is a versatile tool for path planning in fuzzy systems like telecommunications (Parimala, Jafari, et al., 2021).

The research by Liu, Jiang, Chen, Ye, He, and Sun, "Integrating Dijkstra's Algorithm into Deep Inverse Reinforcement Learning for Food Delivery Route Planning" (2020), introduces a deep inverse reinforcement learning model to optimize e-bike food delivery routes in China. It learns rider preferences from GPS trajectories, addressing mismatches between suggested and actual routes. By using Dijkstra's algorithm instead of value iteration, it improves route accuracy, even with incomplete road data. Tested with Meituan-Dianping Group data, it enhances route planning efficiency. The lean design suits dynamic urban settings. This model is a practical tool for efficient delivery logistics (Liu, Jiang, et al., 2020).

2.3 Chapter Conclusion

This chapter explained the core ideas behind the route planning system for robots in small-scale environments. The first section covered the basics of semantic segmentation models, U-Net, FCN, PSPNet, and SegNet which map free and occupied spaces from images. It also described route planning algorithms—A*, Dijkstra's, and Bidirectional search—that find efficient paths for robots. The second section reviewed recent studies showing U-Net and FCN used in map generation for applications like satellite image segmentation for urban planning, building extraction, flooded region detection, agricultural field segmentation, water body segmentation, forest fire smoke detection, urban green space mapping, burned area mapping, landslide detection, and tree species mapping. Recent studies also show A*, Dijkstra's, and Bidirectional search used in route planning for food delivery, ship navigation, automated guided vehicle movement, autonomous driving, taxi route recommendation,

unmanned aerial vehicle navigation, automatic parking, robotic harvesting, nuclear environment navigation, and 3D piping routing. These recent studies from high-ranking journals such as IEEE, Springer and Elsevier show that U-Net, FCN, A*, Dijkstra's, and Bidirectional search are excellent choices for building a route planning system for small-scale environment applications.

Chapter Three

Methods and Procedures

3.1 Route Planning with AI System Overview	52
3.2 Segmentation Models Design	55
3.2.1 U-Net Model	55
3.2.2 Description of the U-Net Mathematical Model	56
3.2.3 FCN Model	57
3.2.4 Description of the FCN Mathematical Model	59
3.2.5 PSPNET Model.....	60
3.2.6 Description of PSPNET Mathematical Model.....	62
3.3 SegNet Model	65
3.3.1 Description of SEGNET Mathematical Model.....	66
3.4 Dataset Preparation Process	70
3.4.1 The Dataset	70
3.4.2 Dataset Preparation Process Steps	71
3.5 Training and Testing for Segmentation Models	74
3.5.1 Overview of Training and Testing Workflow	74
3.5.2 Description of the Training and Testing Workflow.....	75
3.5.3 Project File Division	76
3.5.4 Training Process.....	77
3.5.5 Description of the Training Loop:	79
3.5.6 Performance Evaluation Metrics.....	80
3.5.7 Overview of Performance Evaluation.....	81
3.6 Route Planning Evaluation Process	87
3.6.1 Execution Time.....	88
3.6.2 Memory Usage.....	88
3.6.3 Path Length	89
3.7 Software Tools	90
3.7.1 Microsoft Paint.....	90
3.7.2 Visual Studio Code	91
3.8 Hardware.....	94

Chapter Three

Methods and Procedures

This chapter comprehensively describes the research methods, including the AI model's structure, training and testing procedures, experimental setup, and implementation process.

3.1 Route Planning with AI System Overview

This section provides an overview of the route planning system, an independent framework designed to generate optimal paths for a robot operating in a small-scale environment. Utilizing a top-view camera, the system captures real-time data, processing each frame to adapt to dynamic conditions. The primary focus is on creating a reliable path that a ground robot can follow, without directly controlling its movements. The process begins with image input and progresses through segmentation, grid conversion, and path computation, forming the backbone of the system's functionality.

“Fig. 20” illustrates the initial stage of the route planning system, capturing a new state with each frame per second. In this case, we have a parking setting, where the system processes an input RGB image showing a top-down view of a building and several parked vehicles, representing the environment where a robot on the ground will operate. This image is fed into a segmentation model, which forms the core of the route planning system and can utilize architectures like FCN, U-Net, PSPNet, and SegNet. The model converts the RGB image into a grayscale representation, highlighting areas suitable for the robot's path. This grayscale output is

critical as it isolates navigable regions from obstacles, such as vehicles and structures, laying the groundwork for further processing. The route planning system focuses on generating a path that a ground robot can later follow, without directly controlling the robot itself. The segmentation step ensures that the system can handle the dynamic nature of the small-scale environment, where objects may shift or change position, by providing an adaptable foundation for path generation.

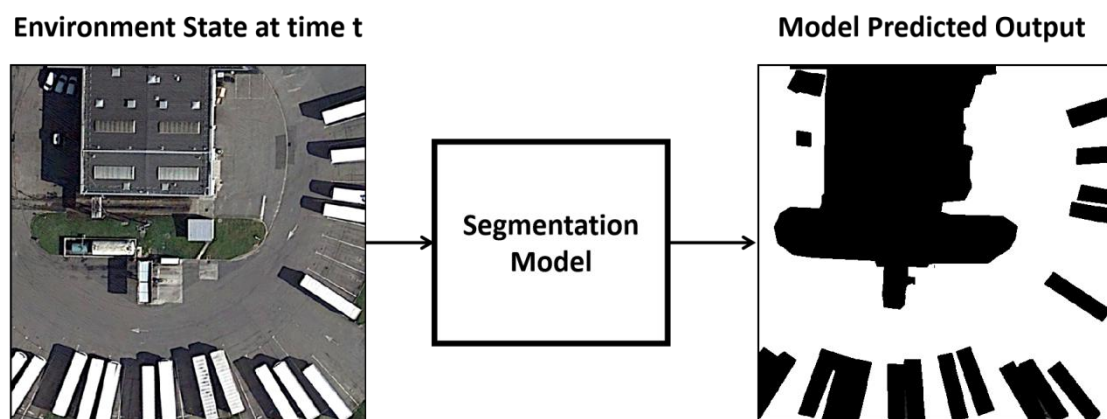


Fig. 20: Image Segmentation

“Fig. 21” illustrates the transformation of the grayscale image within the route planning system, converting it into a structured format for generating paths. The process starts with the "Image to Grid" function, which turns the grayscale output from the segmentation model into a grid data structure. This grid breaks down the environment into cells, each representing a small section of the space, allowing for systematic analysis of the layout. Following this, the "Grid to Binary Matrix" function checks each cell: cells dominated by occupied pixels, such as those representing vehicles or buildings, are labeled as 1, while cells with mostly free pixels are marked as 0. This binary grid distinguishes between navigable and obstructed zones, setting the stage for the final path computation phase.

The figure uses a grid overlay to visually demonstrate how the grayscale data is interpreted into a binary format, highlighting the conversion process. The resolution of the grid will control the path accuracy, ensuring the system can produce a precise route.

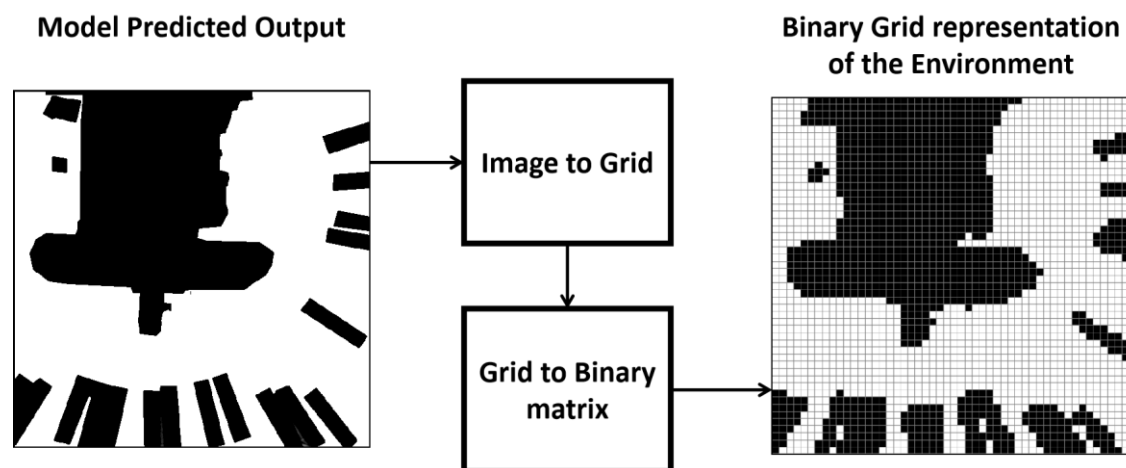


Fig. 21: Segmented image to Binary Grid

“Fig. 22” illustrates the final stage of the route planning system, where a binary grid is used to generate an optimal path for a robot on the ground. The "Route Planning Algorithm" section shows the start point at coordinates $(x=3, y=10)$ and the target point at $(x=40, y=5)$ within the grid, based on the top-down perspective. Algorithms such as A*, Dijkstra, or bidirectional search are applied to analyze the binary grid, finding a path that avoids occupied areas (marked as 1) and navigates through free spaces (marked as 0). The optimal path is depicted with a red line, while black lines represent obstacles, offering a clear visualization of the route from start to target. The chosen algorithm optimizes the path for efficiency, balancing factors like distance or time based on the system’s design goals.

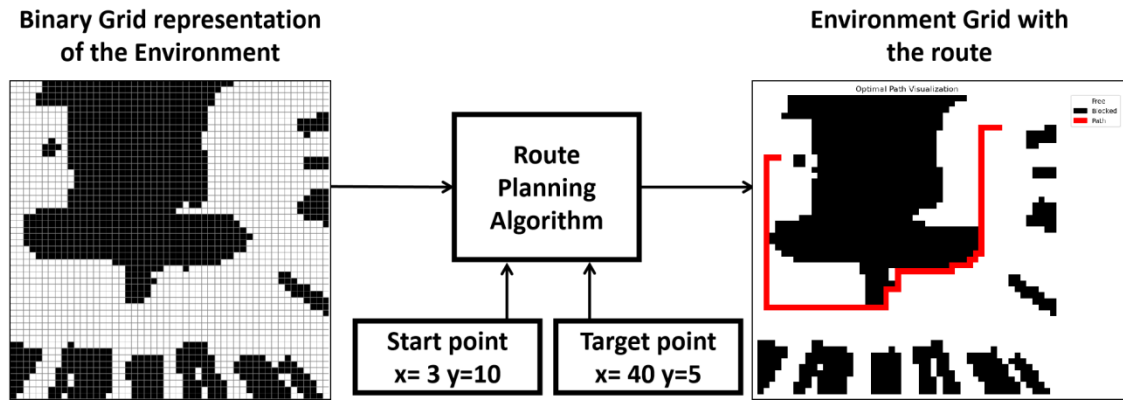


Fig. 22: Route Planning

3.2 Segmentation Models Design

This section details the design of the segmentation models used in the route planning system to process input images and generate binary masks for path identification in a small-scale environment. Four models are explored: U-Net, FCN, PSPNet, and SegNet each with distinct architectures tailored for the task. The subsections below provide an in-depth look at each model, including their architecture diagrams, mathematical formulations, and operational flow, supported by figures for clarity.

3.2.1 U-Net Model

This subsection presents the U-Net model architecture employed in the route planning system. “Fig. 23” illustrates the U-Net architecture, detailing the flow of data through its encoder-decoder structure with skip connections.

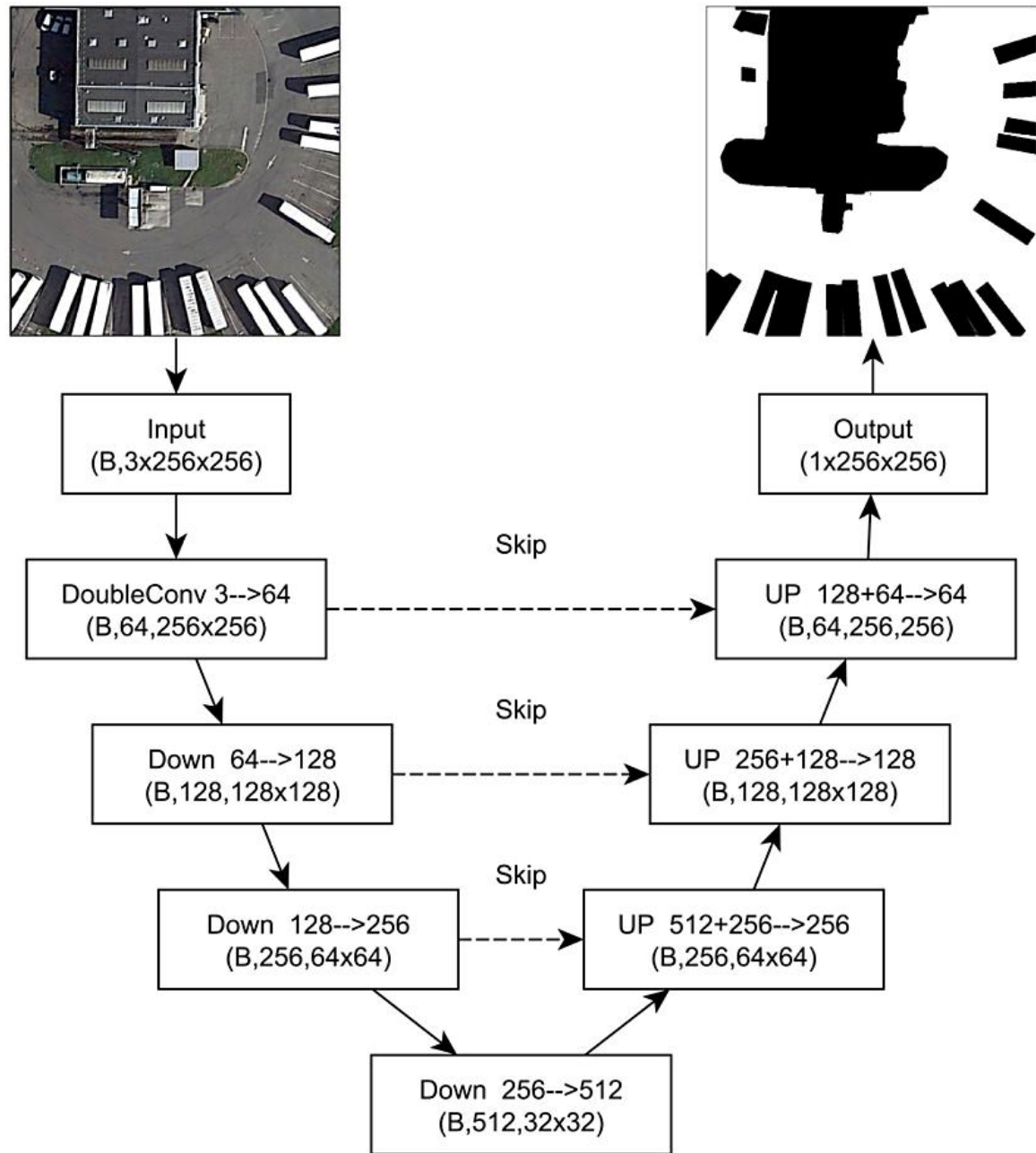


Fig. 23: U-Net Model Architecture

3.2.2 Description of the U-Net Mathematical Model

The U-Net mathematical model follows an encoder-decoder structure tailored for segmentation. It begins with an input image of shape $(B, 3, 256 \times 256)$, where B is the batch size, processed by a DoubleConv block that applies two 3×3 convolutions, each followed by batch

normalization (BN) and ReLU activation. The encoder consists of three Down blocks, each applying 2x2 max pooling with stride 2 followed by a DoubleConv, progressively reducing spatial dimensions while increasing channel depth. The decoder comprises three Up blocks, each performing bilinear upsampling, concatenation with skip connections from the encoder, and a DoubleConv. A final 1x1 convolution maps the output to a single channel, followed by a sigmoid activation and thresholding at 0.5 to produce the binary mask for path planning. The formulas for each block are listed below:

DoubleConv Block:

$$\text{DoubleConv}(x) = \text{ReLU}\left(\text{BN}\left(W_2 * \text{ReLU}\left(\text{BN}(W_1 * x)\right)\right)\right) \quad (9)$$

Down Block:

$$\text{Down}(x) = \text{DoubleConv}(\text{MaxPool}(x)) \quad (10)$$

Up Block:

$$\text{Up}(x_i, x_{\text{skip}}) = \text{DoubleConv}\left(\text{Concat}(\text{Upsample}(x_i), x_{\text{skip}})\right) \quad (11)$$

Output Layer:

$$x_{\text{out}} = W * x_7 + b \quad \text{where} \quad W \in R^{1 \times C_i \times 1 \times 1}, \quad b \in R \quad (12)$$

3.2.3 FCN Model

This subsection introduces the Fully Convolutional Network (FCN) model architecture used in the route planning system. “Fig. 24” show the FCN architecture, outlining the data flow through its convolutional layers.

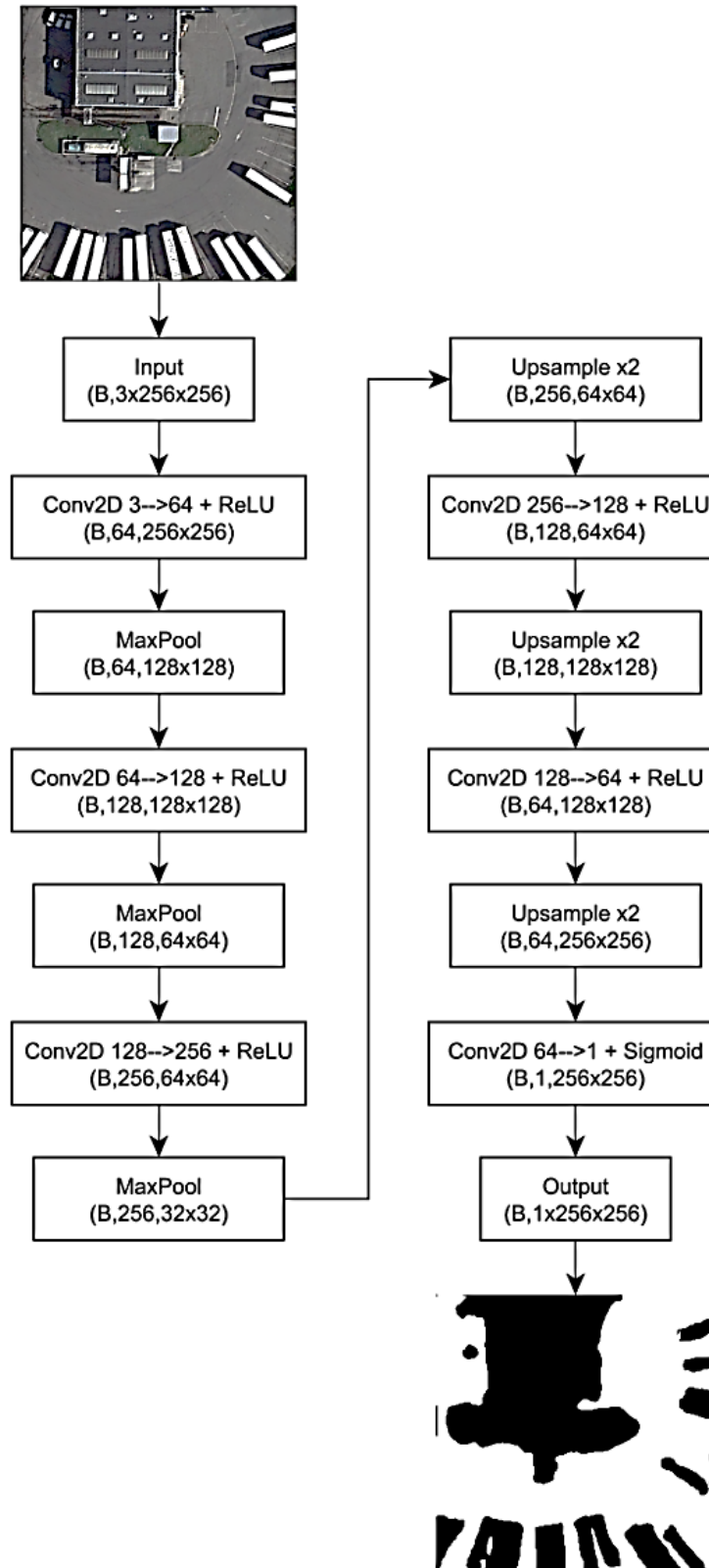


Fig. 24: FCN Model Architecture

3.2.4 Description of the FCN Mathematical Model

The FCN model employs a simpler encoder-decoder architecture without skip connections, designed for segmentation in the route planning system. The encoder starts with an input image of shape $(B, 3, 256 \times 256)$, where B is the batch size, and applies three convolutional layers, each followed by ReLU activation and 2×2 max pooling with stride 2, reducing the spatial dimensions while increasing channel depth. The decoder consists of three blocks, each performing bilinear upsampling, a 3×3 convolution, and ReLU activation. The output layer applies a 3×3 convolution followed by a sigmoid activation, and the output is thresholded at 0.5 to produce the binary mask for path planning. The formulas for each operation are listed below:

Encoder Operations:

$$x_1 = \text{ReLU}(W_1 * x_{\text{in}}) \quad (13)$$

$$x_2 = \text{MaxPool}(x_1) \quad (14)$$

$$x_3 = \text{ReLU}(W_2 * x_2) \quad (15)$$

$$x_4 = \text{MaxPool}(x_3) \quad (16)$$

$$x_5 = \text{ReLU}(W_3 * x_4) \quad (17)$$

$$x_6 = \text{MaxPool}(x_5) \quad (18)$$

Decoder Operations:

$$x_7 = \text{Upsample}(x_6) \quad (19)$$

$$x_8 = \text{ReLU}(W_4 * x_7) \quad (20)$$

$$x_9 = \text{Upsample}(x_8) \quad (21)$$

$$x_{10} = \text{ReLU}(W_5 * x_9) \quad (22)$$

$$x_{11} = \text{Upsample}(x_{10}) \quad (23)$$

$$x_{12} = \text{ReLU}(W_6 * x_{11}) \quad (24)$$

Output Layer:

$$x_{\text{out}} = \sigma(W * x_{12} + b) \quad \text{where} \quad W \in R^{1 \times C_i \times 3 \times 3}, \quad b \in R \quad (25)$$

3.2.5 PSPNET Model

This subsection details the Pyramid Scene Parsing Network (PSPNet) framework developed for pixel-level image segmentation. The model's complete architecture is visualized in “Fig. 25”, highlighting its unique multi-scale feature integration approach.

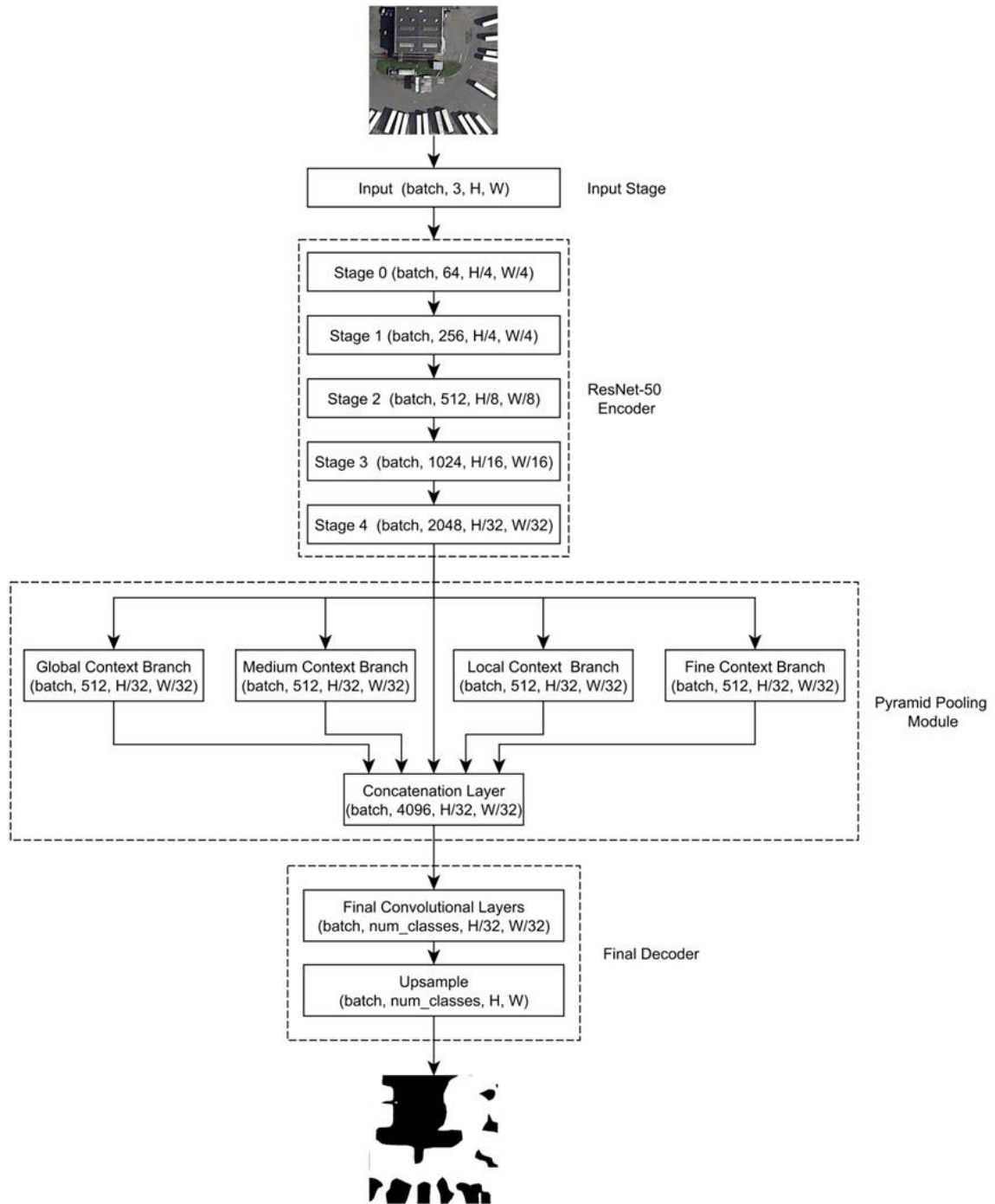


Fig. 25: PSPNET Architecture

3.2.6 Description of PSPNET Mathematical Model

The system employs a dual-component design that combines hierarchical feature extraction with parallel context gathering. The feature extraction component leverages a ResNet-50 foundation structured into five progressive tiers plus an initial processing unit, all utilizing bottleneck structures with shortcut linkages.

Initial Processing Unit (Block 0) handles the raw RGB input measuring (batch, 3, H, W) via a 7×7 convolutional operation using 64 filters with a 2-step movement, succeeded by normalization and ReLU activation. This operation shrinks the spatial measurements to $(H/2, W/2)$ while growing the channel count to 64. A subsequent 3×3 maximum pooling with 2-step movement further compresses spatial dimensions to $(H/4, W/4)$, establishing the base feature set for subsequent processing.

First Residual Tier (Block 1) accepts the 64-channel features at $(H/4, W/4)$ scale and processes them through three consecutive bottleneck units. Every bottleneck contains three convolutional layers: an initial 1×1 convolution for channel compression, a 3×3 convolution for spatial analysis, and a final 1×1 convolution for channel expansion, all integrated with normalization, ReLU activation, and direct pathways. This tier amplifies the channel dimension to 256 while preserving the original $(H/4, W/4)$ spatial scale through careful padding and step configuration.

Second Residual Tier (Block 2) handles the 256-channel features through four bottleneck units, where the initial unit uses 2-step movement in its 3×3 convolution to decrease spatial measurements to $(H/8, W/8)$. Each bottleneck maintains the three-layer configuration, normalization,

and direct pathways. This stage boosts channel capacity to 512 while progressively reducing spatial dimensions.

Third Residual Tier (Block 3) converts the 512-channel features through six bottleneck units, with the starting unit applying 2-step movement to further diminish spatial scale to $(H/16, W/16)$. The multiple bottleneck layers here substantially deepen the network while enlarging channel dimensions to 1024, capturing progressively abstract representations through layered analysis.

Fourth Residual Tier (Block 4), the concluding encoding phase, processes the 1024-channel features through three bottleneck units, with the lead unit employing 2-step movement to establish the ultimate spatial scale of $(H/32, W/32)$. This phase expands the channel dimension to 2048, generating the high-level feature set that feeds into the multi-scale pooling component.

The innovative Pyramid Pooling Module represents the architecture's central advancement, handling the 2048-channel feature sets through four simultaneous context gathering branches operating at varying spatial details, supplemented by a direct pathway that maintains the original features.

Worldwide Context Pathway (Bin=1) implements adjustable average pooling to condense spatial measurements to 1×1 , encompassing the complete global setting of the feature set. A 1×1 convolutional layer then shrinks the channel dimension from 2048 to 512, followed by normalization and ReLU activation. The resulting features undergo bilinear enlargement back to the initial $(H/32, W/32)$ spatial proportion.

Regional Context Pathway (Bin=2) utilizes adjustable average pooling to generate 2×2 spatial divisions, capturing zonal context at intermediate detail. The identical channel reduction procedure applies before bilinear scaling restores (H/32, W/32) proportion.

Local Context Pathway (Bin=3) employs adjustable average pooling with 3×3 divisions to capture finer neighborhood context, treating the pooled features through the same channel compression sequence before enlargement.

Detailed Context Pathway (Bin=6) applies the most refined pooling with 6×6 divisions to gather intricate local data, following the matching channel reduction and enlargement process as other pathways.

Direct Pathway conserves the original 2048-channel input features at (H/32, W/32) proportion, retaining the precise spatial data derived by the ResNet-50 foundation without modification.

The PPM result is produced by merging all five routes along the channel axis: the four context pathways (each 512 channels) plus the direct connection (2048 channels), forming a comprehensive 4096-channel feature collection that blends multi-scale contextual data with original spatial information.

The decoding segment processes this enriched representation through a streamlined convolutional block starting with a 3×3 convolution that decreases the channel dimension from 4096 to 512, followed by normalization, ReLU activation, and regularization. A subsequent 1×1 convolutional layer maps the 512-channel features directly to the target

category count, generating the preliminary segmentation layout at reduced ($H/32$, $W/32$) proportion.

A concluding bilinear scaling layer reinstates the original input proportion by interpolating the segmentation layout from ($H/32$, $W/32$) to (H , W), creating pixel-level segmentation charts that maintain exact spatial correspondence with the input images while utilizing thorough contextual comprehension across multiple scales through the pyramid pooling mechanism.

PSPNet's primary benefit stems from its capacity to gather and combine contextual information across multiple scales via the pyramid pooling component, allowing the network to comprehend both minute details and overall scene setting concurrently. This multi-level method proves especially valuable for intricate semantic segmentation challenges where objects present at diverse sizes and demand varying degrees of contextual interpretation for precise classification.

3.3 SegNet Model

This subsection introduces the SegNet model architecture implemented for semantic segmentation tasks. “Fig. 26” illustrates the complete SegNet framework, depicting the symmetric encoder-decoder data flow through its precise pooling index mechanism.

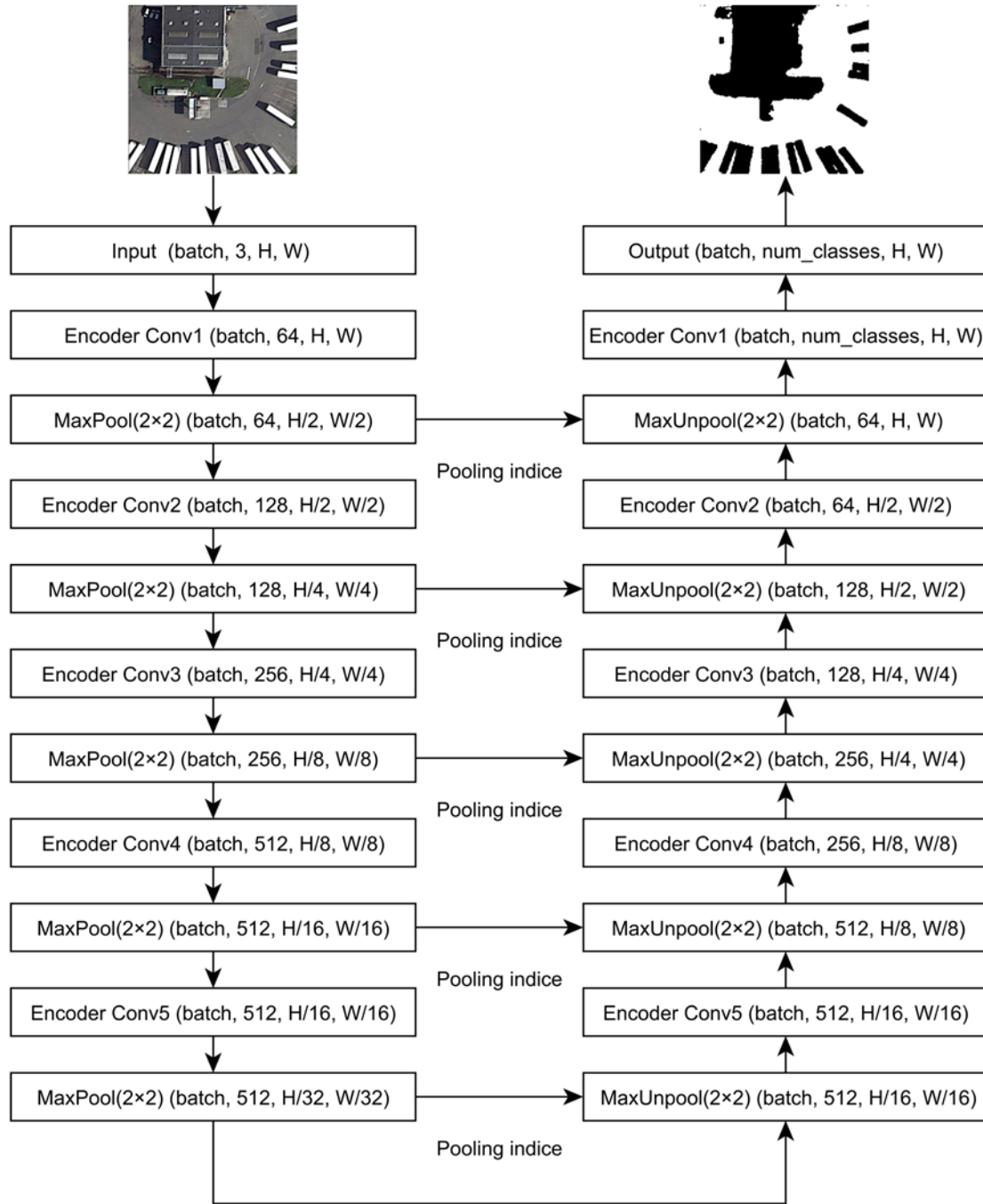


Fig. 26: SEgNET Architecture

3.3.1 Description of SEgNET Mathematical Model

The network employs a fully convolutional encoder-decoder architecture based on VGG16, organized into five symmetric encoding and decoding stages. The encoder pathway processes input images of

dimensions (batch, 3, H, W) through successive convolutional blocks that progressively extract features while reducing spatial resolution.

Encoder Block 1 processes the RGB input through two sequential 3×3 convolutional layers, each containing 64 filters with batch normalization and ReLU activation. The first convolution transforms the input from 3 to 64 channels, while the second maintains 64 channels, both preserving spatial dimensions through 1-pixel padding. This is followed by a 2×2 max pooling operation that reduces spatial resolution to $(H/2, W/2)$ while storing pooling indices for future unpooling.

Encoder Block 2 receives the 64-channel, half-resolution features and applies two 3×3 convolutional layers with 128 filters each. Both layers include batch normalization and ReLU activation, expanding the channel dimension to 128 while maintaining the $(H/2, W/2)$ spatial size. A subsequent 2×2 max pooling further reduces resolution to $(H/4, W/4)$, storing new pooling indices.

Encoder Block 3 processes the 128-channel features through three sequential 3×3 convolutional layers, each containing 256 filters with batch normalization and ReLU activation. This block significantly expands the feature representation to 256 channels while preserving the $(H/4, W/4)$ spatial dimensions. The concluding 2×2 max pooling reduces resolution to $(H/8, W/8)$ with corresponding index storage.

Encoder Block 4 transforms the 256-channel input through three 3×3 convolutional layers with 512 filters each, employing batch normalization and ReLU activation to produce 512-channel features at $(H/8, W/8)$

resolution. Following max pooling, the output dimensions become $(H/16, W/16)$ with 512 channels.

Encoder Block 5, the final encoding stage, processes the 512-channel features through another three 3×3 convolutional layers, each maintaining 512 filters with batch normalization and ReLU activation. The resulting 512-channel features at $(H/16, W/16)$ resolution undergo final max pooling to create the bottleneck representation at $(H/32, W/32)$ with 512 channels.

The distinctive feature of SegNet lies in its use of pooling indices rather than skip connections for spatial recovery. During encoding, each max pooling operation stores the positions of maximum values, which are subsequently used in the decoder for precise unpooling operations.

The decoder pathway mirrors the encoder architecture through five symmetric blocks that progressively recover spatial resolution while refining features:

Decoder Block 5 begins with max unpooling using stored indices from encoder block 5, doubling spatial resolution from $(H/32, W/32)$ to $(H/16, W/16)$ while maintaining 512 channels. Three sequential 3×3 convolutional layers, each with 512 filters, batch normalization, and ReLU activation, process the unpooled features to refine the representation.

Decoder Block 4 applies unpooling using encoder block 4 indices, increasing resolution from $(H/16, W/16)$ to $(H/8, W/8)$. Three 3×3 convolutional layers process these features - the first two maintain 512

channels, while the third reduces channels to 256, all employing batch normalization and ReLU activation.

Decoder Block 3 uses unpooling with encoder block 3 indices to recover resolution from $(H/8, W/8)$ to $(H/4, W/4)$. Three convolutional layers process the 256-channel input - the first two maintain 256 channels, while the third reduces to 128 channels, each with batch normalization and ReLU activation.

Decoder Block 2 applies unpooling with encoder block 2 indices, increasing resolution from $(H/4, W/4)$ to $(H/2, W/2)$. Two 3×3 convolutional layers process the 128-channel features - the first maintains 128 channels, while the second reduces to 64 channels, both with batch normalization and ReLU activation.

Decoder Block 1, the final decoding stage, uses unpooling with encoder block 1 indices to restore full spatial resolution from $(H/2, W/2)$ to (H, W) . Two 3×3 convolutional layers process the 64-channel features - the first maintains 64 channels with batch normalization and ReLU activation, while the second projects directly to the target number of classes without batch normalization or activation.

The key distinction between SegNet and other encoder-decoder architectures lies in its use of pooling indices versus skip connections. While U-Net employs dense skip connections that concatenate entire feature maps from encoder to decoder, providing comprehensive multi-resolution details at the cost of significant memory overhead, SegNet utilizes only the stored pooling indices positions during unpooling operations. This approach offers substantial memory efficiency by storing

minimal index information rather than complete feature maps, making SegNet particularly suitable for memory-constrained environments. However, this efficiency comes with potential trade-offs in boundary precision for complex segmentation scenarios where dense feature information proves beneficial.

The final output generates pixel-wise segmentation maps that maintain precise spatial alignment with the original input while leveraging efficient hierarchical feature extraction and recovery through the symmetric encoder-decoder architecture with pooling index mechanism.

3.4 Dataset Preparation Process

This section outlines the dataset preparation process for training the segmentation models in the route planning system. The process involves several steps to ensure the dataset is diverse, well-structured, and suitable for training models to handle varying camera rotations and brightness levels in a small-scale environment. These steps include image preprocessing, augmentation, mask creation, renaming, and dataset splitting, all of which are critical for effective model training and evaluation.

3.4.1 The Dataset

The iSAID dataset (Zamir et al.) contains 2,806 high-resolution aerial images with 655,451 instances across 15 classes, sourced from DOTA-v1.0, Google Earth, JL-1, and GF-2 satellites. It is a benchmark dataset widely used in segmentation tasks. The system is designed for indoor and outdoor small-scale environments (e.g., warehouses, retail, parking, loading zones). Due to the lack of public indoor datasets—most warehouses and factories restrict data sharing for security—the study used

outdoor aerial imagery. Specifically, truck loading/unloading areas were selected from iSAID, as they represent real small-scale robot-operable zones with clear free/occupied patterns similar to indoor zones. Originally planned for hybrid indoor-outdoor training, the experiment proceeded with outdoor data only. Still, results prove the concept and system capability, as top-view binary segmentation generalizes across domains. Future work can add indoor data when available.

3.4.2 Dataset Preparation Process Steps

The dataset preparation follows the structured workflow shown in “Fig. 27”. Selected images from the iSAID dataset representing truck loading and unloading areas, originally of varying resolutions, are resized to a fixed 256×256 pixels using bilinear interpolation via `transforms.Resize((256, 256))` from PyTorch’s `torchvision.transforms`. Images are then converted to tensors with `transforms.ToTensor()`, which automatically scales pixel values from $[0, 255]$ to $[0.0, 1.0]$ and transposes them to (C, H, W) format. No additional normalization (e.g., ImageNet mean/std) is applied, as the binary segmentation task benefits from raw intensity values for clear free/occupied distinction. A custom `BinarySegmentationDataset` class inherits from `torch.utils.data.Dataset` and loads paired image-mask files, ensuring synchronized loading and transformation.

Data augmentation is implemented to improve model generalization to rotation and lighting variations. Each original image is processed through four rotations (0° , 90° , 180° , 270°) and four brightness levels generating 16 augmented versions per input using Pillow package. Binary masks are manually created in Microsoft Paint for all images (original and augmented) and saved as grayscale PNGs. Image-mask pairs are renamed

sequentially to maintain alignment (e.g., image_001.jpg → mask_001.png). The full dataset is split into 80% training (800 images) and 20% testing (200 images) using a fixed random seed. Training and testing sets are organized into separate folders: Training Dataset/images/, Training Dataset/masks/, Testing Dataset/images/, and Testing Dataset/masks/. Data loading is handled by DataLoader with a batch size of 4 and shuffling enabled for training.

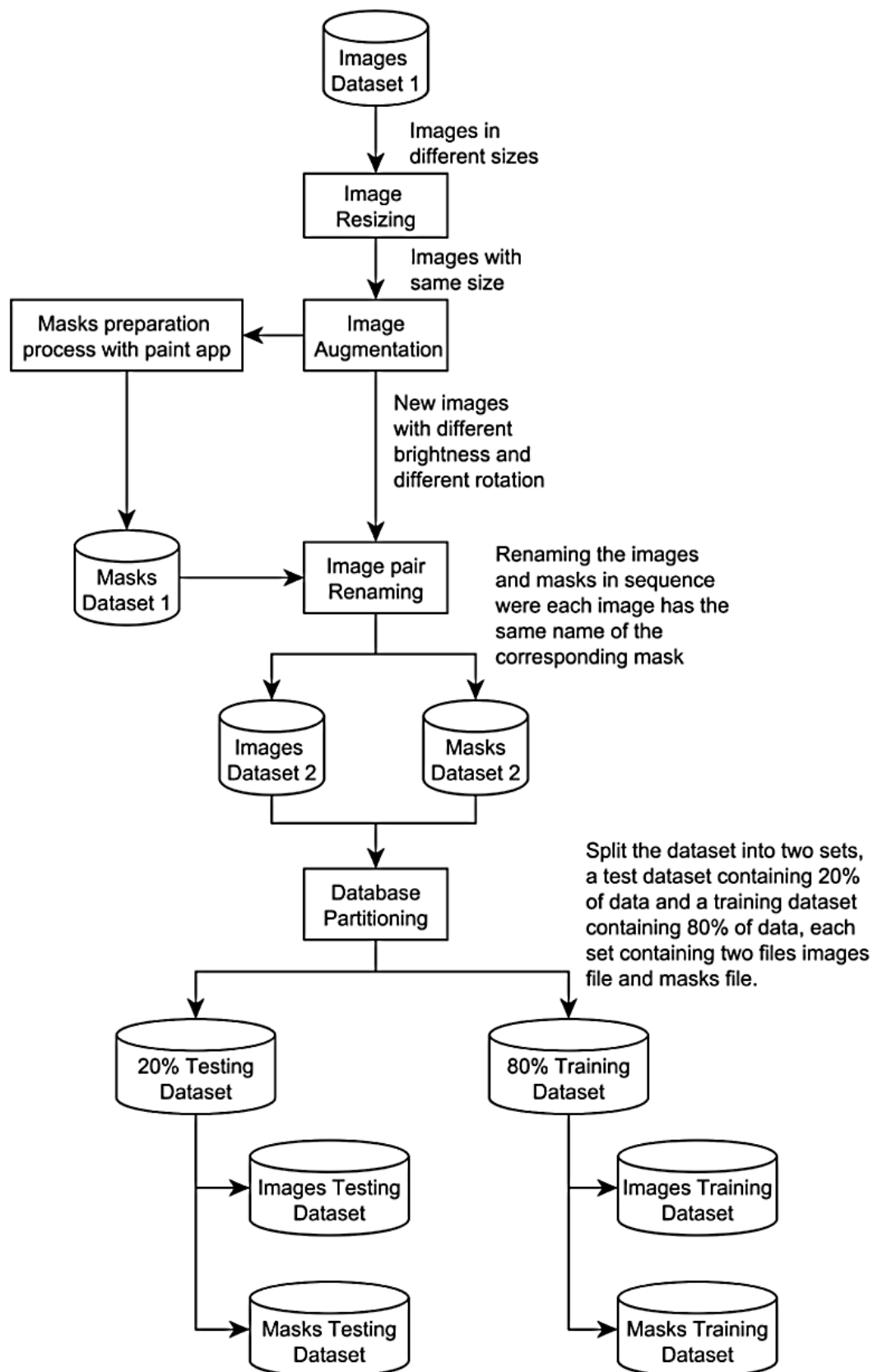


Fig. 27: Dataset Preparation Process

3.5 Training and Testing for Segmentation Models

This section focuses on the training and testing procedures for the segmentation models developed in the route planning system. It covers the overall workflow, project file organization, detailed training process, and performance evaluation metrics used to assess the models' effectiveness. The subsections below provide a comprehensive overview of each aspect, supported by figures to illustrate the processes and tables to summarize key configurations and evaluation criteria, ensuring a thorough understanding of how the models are trained, tested, and evaluated for integration into the route planning system.

3.5.1 Overview of Training and Testing Workflow

This subsection introduces the high-level workflow for training and testing the segmentation models, providing a simplified view of the process. “Fig. 28” illustrates the workflow, highlighting the key steps and components involved.

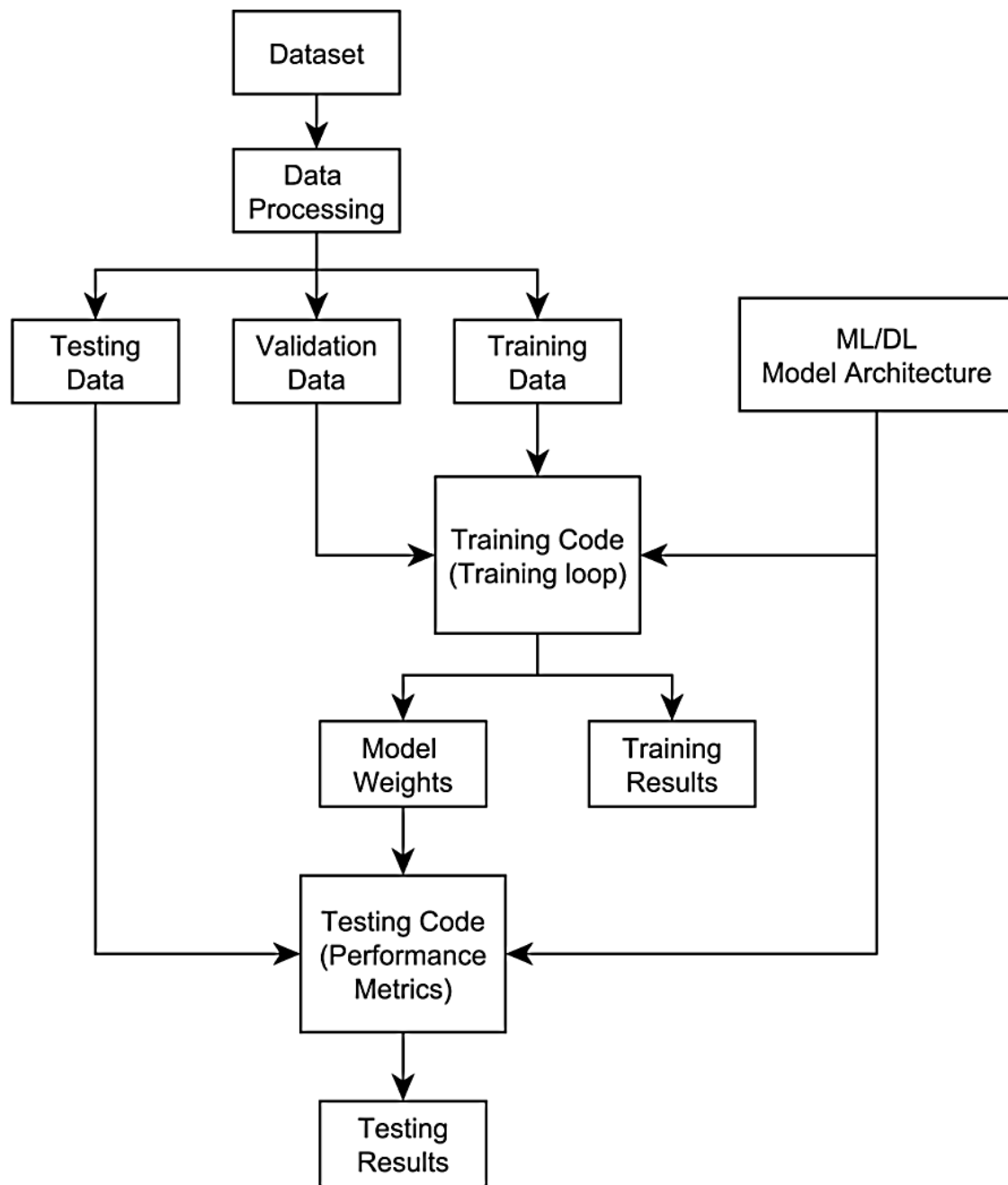


Fig. 28: Training and Testing Workflow

3.5.2 Description of the Training and Testing Workflow

The training and testing workflow begins with processing the dataset, which is then divided into training, validation, and testing subsets. The figure illustrates two primary code files: the training file and the testing file, alongside a model file that defines the segmentation model

architecture. The training file loads the training and validation datasets, imports the model to create a model object, and executes the training process, producing model weights and training results as outputs. These training results are analyzed to determine if the model has been trained correctly; if the results are unsatisfactory, the training process is repeated until a satisfactory outcome is achieved. Once satisfactory weights are obtained, they are used in the testing file, which loads the testing dataset, imports the model, creates a model object, and applies the weights to evaluate the model. The output of the testing phase is the performance results, and if these results meet the desired goals, the model and its weights are deemed ready for integration into the route planning system, as depicted in the figure.

3.5.3 Project File Division

This subsection outlines the organization of the project files used in the training and testing of the segmentation models. The structure is clearly presented in “Fig. 29”, which requires no further discussion as it is self-explanatory.

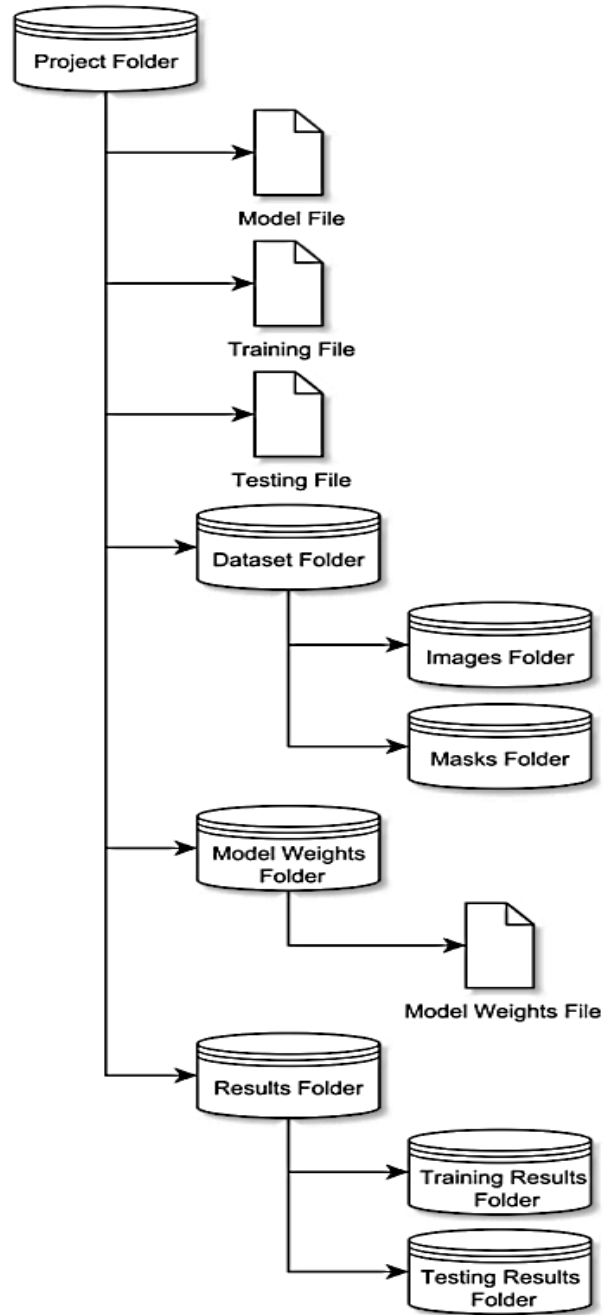


Fig. 29: Project File Division

3.5.4 Training Process

This subsection dive into the detailed training process for the U-Net and FCN segmentation models, focusing on the training loop and the configurations used. “Fig. 30” illustrates the training loop, followed by a table summarizing the key training parameters for both models.

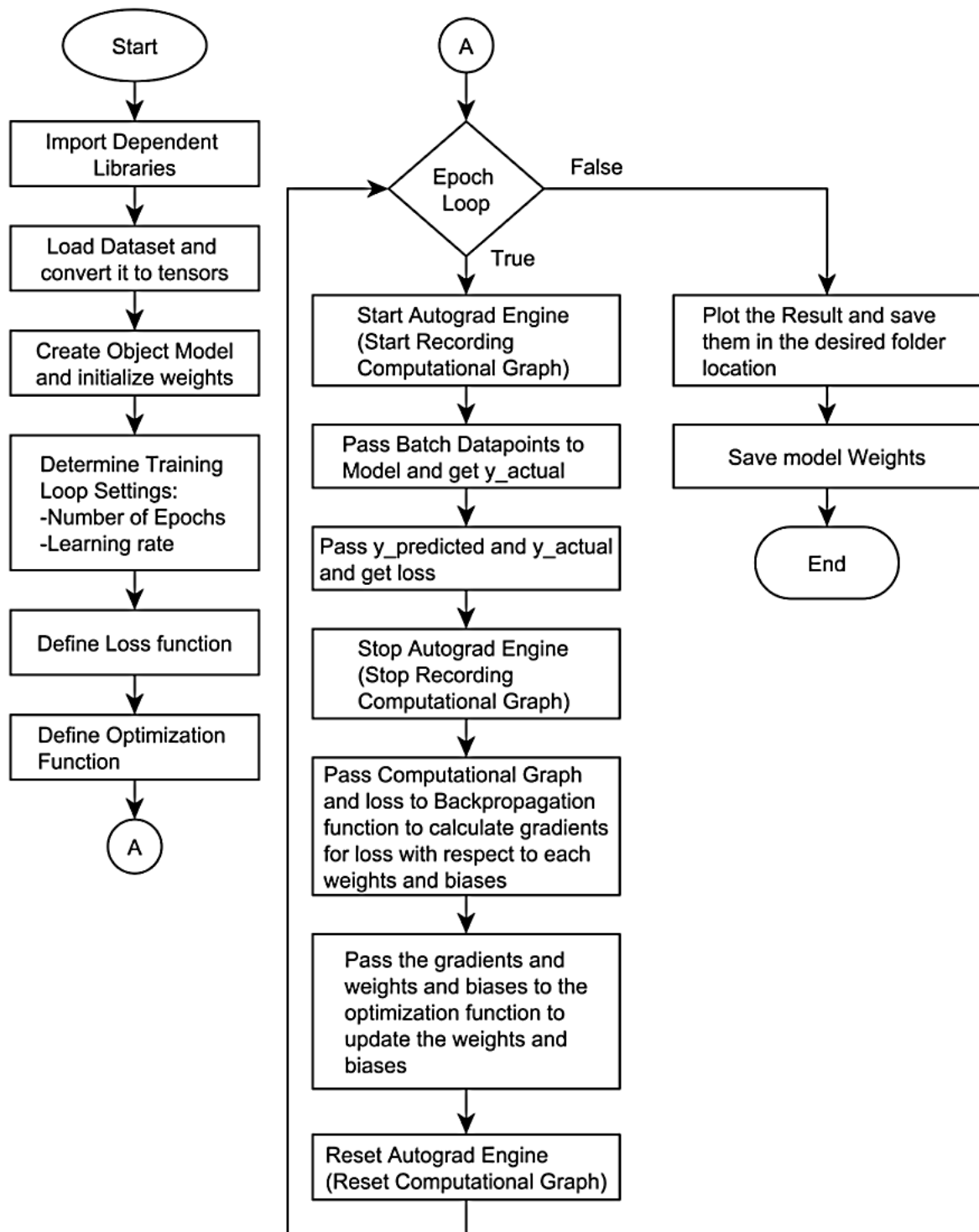


Fig. 30: Training Loop

3.5.5 Description of the Training Loop:

The training loop, as shown in the figure, outlines the iterative process of training the segmentation models. It begins with loading the training and validation datasets, followed by initializing the model with predefined parameters. For each epoch, the model processes batches of training data, computes the loss, updates the model weights using the optimizer, and evaluates performance on the validation set. This loop continues for a specified number of epochs, adjusting the model weights to minimize the loss and improve segmentation accuracy. The training configurations for U-Net and FCN, including the optimization function, loss function, number of epochs, learning rate, and gradient descent type, are summarized in the Table (1) below

Table (1): Training Configurations for each Segmentation Models

Model	Loss Function	Optimization Function	Epochs With less loss	Learning Rate	Gradient Descent Type
U-Net	Binary Cross Entropy with Logits Loss	Adam	49	0.001	Batch Gradient Descent
FCN	Binary Cross Entropy Loss	Adam	48	0.001	Batch Gradient Descent
PSPNET	Binary Cross Entropy with Logits Loss	Adam	50	0.001	Batch Gradient Descent
SegNet	Binary Cross Entropy with Logits Loss	Adam	49	0.001	Batch Gradient Descent

3.5.6 Performance Evaluation Metrics

This subsection discusses the performance evaluation metrics used to assess the segmentation models, focusing on binary segmentation tasks. “Fig. 31” highlights key aspects of the evaluation process, followed by a detailed explanation of the confusion matrix and the metrics used to quantify model performance.

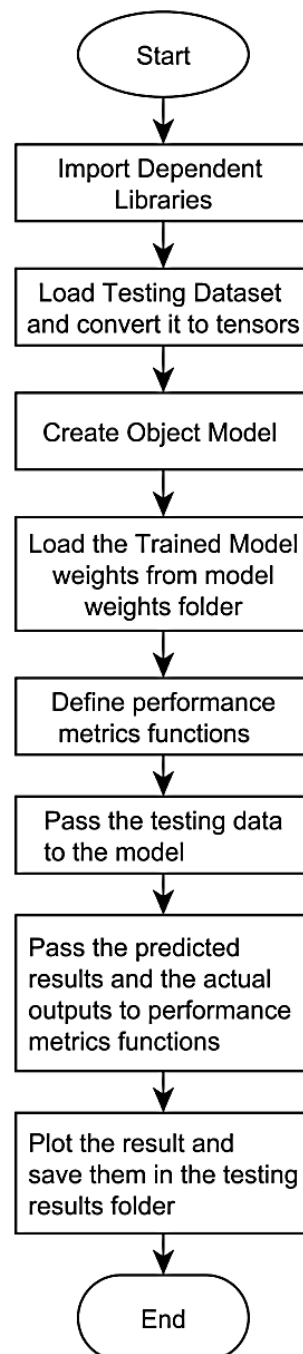


Fig. 31: Performance Evaluation Overview

3.5.7 Overview of Performance Evaluation

The evaluation process, as depicted in the figure, involves analyzing the model's predictions against ground truth masks to compute various performance metrics for binary segmentation. The confusion matrix is a fundamental tool in this process shown in Table (2), defining four key outcomes: True Positive (TP), where pixels are correctly predicted as foreground; True Negative (TN), where pixels are correctly predicted as background; False Positive (FP), where pixels are incorrectly predicted as foreground; and False Negative (FN), where pixels are incorrectly predicted as background. This matrix is structured with actual positives and negatives on the rows and predicted positives and negatives on the columns. Eleven performance metrics are calculated per image, and their mean values across the dataset are reported to provide a comprehensive assessment of the model's performance.

Table (2): Confusion Matrix for Binary Segmentation

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The following are a list of 11 performance metrics used in the testing process.

1. Pixel Accuracy: This performance metric measures the proportion of correctly classified pixels. Equation 26 are used to calculate the performance metric for each testing image while equation 27 calculates the mean for all the testing images.

$$\text{Pixel Accuracy}_i = \frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i} \quad (26)$$

$$\text{Mean Pixel Accuracy} = \frac{1}{N} \sum_{i=1}^N \text{Pixel Accuracy}_i \quad (27)$$

2. Per-Class Accuracy: This performance metric computes the average accuracy for each class (foreground and background). Equation 28, 29 and 30 are used to calculate the performance metric for each testing image while equation 31 calculates the mean for all the testing images.

$$\text{Foreground Accuracy}_i(FA) = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \quad (28)$$

$$\text{Background Accuracy}_i(BA) = \frac{\text{TN}_i}{\text{TN}_i + \text{FP}_i} \quad (29)$$

$$\text{Per-Class Accuracy}_i = \frac{FA + BA}{2} \quad (30)$$

$$\text{Mean Per-Class Accuracy} = \frac{1}{N} \sum_{i=1}^N \text{Per-Class Accuracy}_i \quad (31)$$

3. Dice Coefficient: This performance metric measures the overlap between predicted and ground truth regions, equivalent to the F1 Score for segmentation. Equation 32 are used to calculate the performance metric for

each testing image while equation 33 calculates the mean for all the testing images.

$$\text{Dice Coefficient}_i = \frac{2 \cdot \text{TP}_i}{2 \cdot \text{TP}_i + \text{FP}_i + \text{FN}_i} \quad (32)$$

$$\text{Mean Dice Coefficient} = \frac{1}{N} \sum_{i=1}^N \text{Dice Coefficient}_i \quad (33)$$

4. Jaccard Index (Intersection over Union, IoU): This performance metric measures the ratio of intersection to union of predicted and ground truth regions. Equation 34 are used to calculate the performance metric for each testing image while equation 35 calculates the mean for all the testing images.

$$\text{Jaccard Index}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i + \text{FN}_i} \quad (34)$$

$$\text{Mean Jaccard Index} = \frac{1}{N} \sum_{i=1}^N \text{Jaccard Index}_i \quad (35)$$

5. Precision: This performance metric measures the proportion of correctly predicted foreground pixels among all predicted foreground pixels. Equation 36 are used to calculate the performance metric for each testing image while equation 37 calculates the mean for all the testing images.

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \quad (36)$$

$$\text{Mean Precision} = \frac{1}{N} \sum_{i=1}^N \text{Precision}_i \quad (37)$$

6. Recall: This performance metric measures the proportion of correctly predicted foreground pixels among all actual foreground pixels. Equation 38 are used to calculate the performance metric for each testing image while equation 39 calculates the mean for all the testing images.

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \quad (38)$$

$$\text{Mean Recall} = \frac{1}{N} \sum_{i=1}^N \text{Recall}_i \quad (39)$$

7. F1 Score: This performance metric is the harmonic mean of Precision and Recall, equivalent to the Dice Coefficient in this context. Equation 40 are used to calculate the performance metric for each testing image while equation 41 calculates the mean for all the testing images.

$$\text{F1 Score}_i = \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \quad (40)$$

$$\text{Mean F1 Score} = \frac{1}{N} \sum_{i=1}^N \text{F1 Score}_i \quad (41)$$

8. Hausdorff Distance: This performance metric measures the maximum distance between the boundaries of predicted and ground truth regions. For sets A_i (ground truth boundary) and B_i (predicted boundary) in the i – th image, the directed Hausdorff distance is:

$$h(A_i, B_i) = \max_{a \in A_i} \left\{ \min_{b \in B_i} \{d(a, b)\} \right\} \quad (42)$$

$$h(B_i, A_i) = \max_{b \in B_i} \left\{ \min_{a \in A_i} \{d(a, b)\} \right\} \quad (43)$$

$$\text{Hausdorff Distance}_i = \max\{h(A_i, B_i), h(B_i, A_i)\} \quad (44)$$

Where $d(a, b)$ is the Euclidean distance between points a and b .

$$\text{Mean Hausdorff Distance} = \frac{1}{N} \sum_{i=1}^N \text{Hausdorff Distance}_i \quad (45)$$

9. Average Symmetric Surface Distance (ASSD): This performance metric computes the average distance between the boundaries of predicted and ground truth regions. Equation 46 are used to calculate the performance metric for each testing image while equation 47 calculates the mean for all the testing images.

$$\text{ASSD}_i = \frac{1}{|A_i| + |B_i|} \left(\sum_{a \in A_i} \min_{b \in B_i} d(a, b) + \sum_{b \in B_i} \min_{a \in A_i} d(a, b) \right) \quad (46)$$

where $|A_i|$ and $|B_i|$ are the number of points in the boundary sets A_i and B_i , respectively.

$$\text{Mean ASSD} = \frac{1}{N} \sum_{i=1}^N \text{ASSD}_i \quad (47)$$

10. Frequency-Weighted IoU: This performance metric weights the IoU by the frequency of each class. Equation 48 are used to calculate the performance metric for each testing image while equation 49 calculates the mean for all the testing images.

$$\text{FW-IoU}_i = \frac{(\text{TP}_i + \text{FN}_i) \cdot \text{TP}_i + (\text{TN}_i + \text{FP}_i) \cdot \text{TN}_i}{(\text{TP}_i + \text{FN}_i) \cdot (\text{TP}_i + \text{FP}_i + \text{FN}_i) + (\text{TN}_i + \text{FP}_i) \cdot (\text{TN}_i + \text{FP}_i + \text{FN}_i)} \quad (48)$$

$$\text{Mean FW-IoU} = \frac{1}{N} \sum_{i=1}^N \text{FW-IoU}_i \quad (49)$$

11. Boundary F1 Score: This performance metric evaluates the alignment of predicted and ground truth boundaries by considering pixels within a small distance threshold as matches. Let TP_{b_i} , FP_{b_i} , and FN_{b_i} be the boundary-based true positives, false positives, and false negatives for the i - th image, respectively. Equation 50, 51 and 52 are used to calculate the performance metric for each testing image while equation 53 calculates the mean for all the testing images.

$$\text{Boundary Precision}_i = \frac{\text{TP}_{b_i}}{\text{TP}_{b_i} + \text{FP}_{b_i}} \quad (50)$$

$$\text{Boundary Recall}_i = \frac{\text{TP}_{b_i}}{\text{TP}_{b_i} + \text{FN}_{b_i}} \quad (51)$$

$$\text{Boundary F1 Score}_i = \frac{2 \cdot \text{Boundary Precision}_i \cdot \text{Boundary Recall}_i}{\text{Boundary Precision}_i + \text{Boundary Recall}_i} \quad (52)$$

$$\text{Mean Boundary F1 Score} = \frac{1}{N} \sum_{i=1}^N \text{Boundary F1 Score}_i \quad (53)$$

3.6 Route Planning Evaluation Process

This section examines the evaluation process for route planning algorithms integrated into the navigation system within grid-based small-scale environments. Three key performance metrics are employed to assess the effectiveness of these algorithms: A*, Dijkstra's, and Bidirectional methods. The evaluation leverages a testing image as the environment, with 1,000 pairs of start and end points tested, and the results are analyzed using these metrics, with final values reported as the mean across all runs. The subsections below detail the metrics and their application, supported by a structured approach to ensure a comprehensive assessment of the algorithms' performance.

3.6.1 Execution Time

This performance metric measures the time taken by the algorithm to compute a path from the start to the goal in a grid-based environment, recorded in seconds. Equation 54 are used to calculate the performance metric for each testing datapoint while equation 55 calculates the mean for all the testing datapoints.

$$\text{Execution Time}_i = t_{\text{end},i} - t_{\text{start},i} \quad (54)$$

where $t_{\text{start},i}$ and $t_{\text{end},i}$ are the timestamps (in seconds) when the algorithm begins and completes path computation for the $i - th$ run, respectively.

$$\text{Mean Execution Time} = \frac{1}{N} \sum_{i=1}^N \text{Execution Time}_i \quad (55)$$

3.6.2 Memory Usage

This performance metric quantifies the amount of memory required by the algorithm during path computation, including storage for data structures, measured in megabytes (MB). Equation 56 are used to calculate the performance metric for each testing datapoint while equation 57 calculates the mean for all the testing datapoints.

$$\text{Memory Usage}_i = M_{\text{peak},i} \quad (56)$$

where $M_{\text{peak},i}$ is the peak memory allocated (in MB) during the $i - th$ run, measured using profiling tools or system monitors.

$$\text{Mean Memory Usage} = \frac{1}{N} \sum_{i=1}^N \text{Memory Usage}_i \quad (57)$$

3.6.3 Path Length

This metric measures the total number of grid cells traversed from start to goal in the binary grid representation. The path length is calculated using Equation 58 for each test run and averaged via Equation 59. The unit is grid steps, where each step equals one cell (1 unit). Although the iSAID dataset includes images captured at varying altitudes, the height differences are small across nearly all selected images (typically within a narrow range suitable for small-scale loading zones), minimizing scale variation. However, due to the lack of standardized ground sampling distance in publicly available aerial datasets, consistent pixel-to-meter scaling was not applied and remains outside the scope of control. Thus, path length is reported in abstract grid units to ensure fair and consistent comparison across all images and models.

$$\text{Path Length}_i = \sum_{c \in P_i} 1 \quad (58)$$

Where P_i is the path computed in the i – th run, and $\mathcal{C}$ represents a grid cell in the path, with each cell contributing a unit cost of 1.

$$\text{Mean Path Length} = \frac{1}{N} \sum_{i=1}^N \text{Path Length}_i \quad (59)$$

3.7 Software Tools

This section highlights the software tools employed in the development and evaluation of the route planning system, focusing on their specific roles in preparing data and implementing the segmentation and navigation algorithms. The tools range from image editing software to integrated development environments and essential Python packages, each contributing uniquely to the project's success. The subsections below detail the primary tools and the Python libraries utilized, supported by illustrative figures where applicable.

3.7.1 Microsoft Paint

The iSAID dataset provides annotations for classification and object detection (bounding boxes and class labels), which are unsuitable for semantic segmentation. Therefore, all 1,000 ground truth masks (800 training + 200 testing) were manually created using Microsoft Paint, as shown in “Fig. 32”. Paint’s layer feature allows drawing on a transparent overlay without altering the original image, while the polygon and brush tools enable precise outlining of occupied regions (black = 1) and free spaces (white = 0). The high-resolution aerial images were clear and precise, allowing accurate segmentation by visual inspection without requiring additional measurement tools.

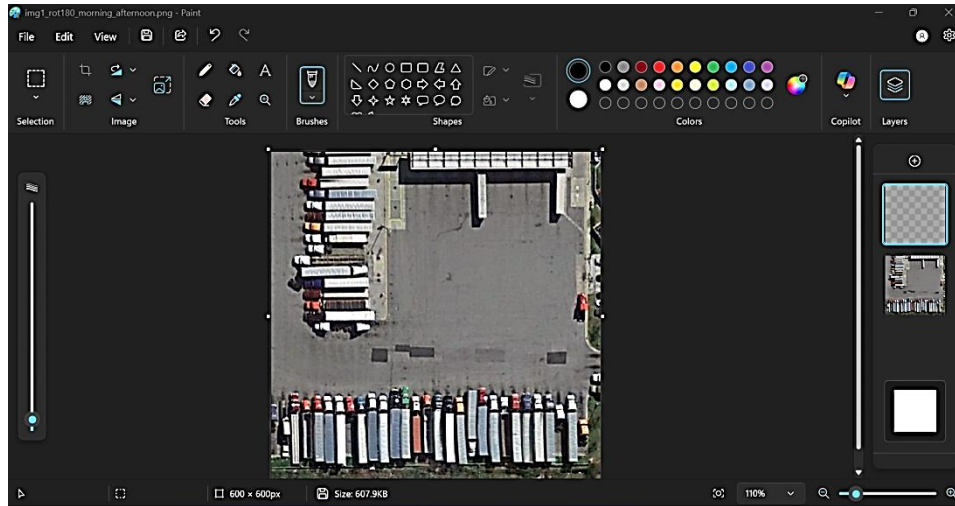


Fig. 32: Microsoft Paint Screenshot

3.7.2 Visual Studio Code

This subsection outlines the role of Visual Studio Code as the primary development environment for building the entire route planning system. “Fig. 33” below provides a screenshot of Visual Studio Code, illustrating its interface during development.

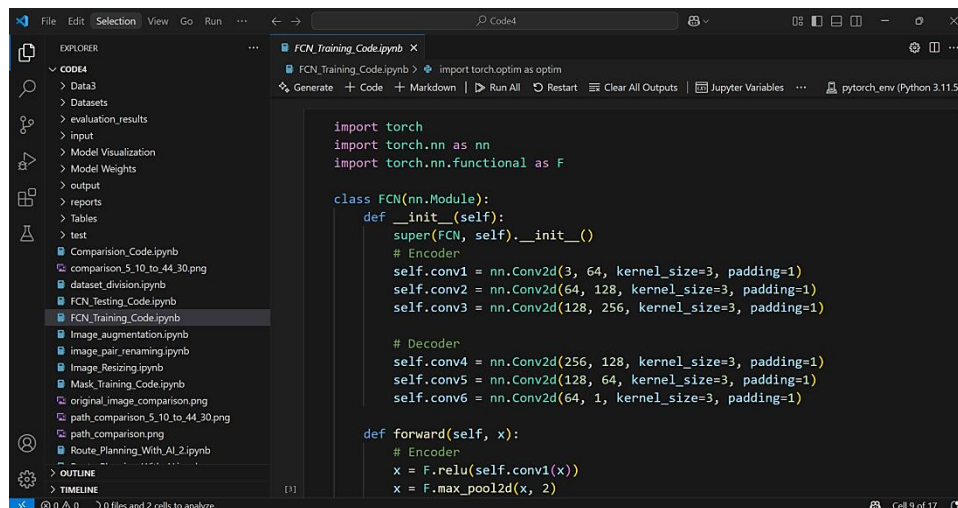


Fig. 33: Visual Studio Code Screenshot

Visual Studio Code is the core development platform for the route planning system, as shown in the screenshot. The figure displays an active workspace with multiple Python files open for model training and route planning algorithms in a tabbed layout, a project directory structure in the left sidebar, and an integrated terminal for running scripts and debugging. This editor supports various languages with a customizable workspace, offering debugging tools, version control, and plugin support. Developers utilized its features, like syntax highlighting and IntelliSense, to efficiently code and test the segmentation models and navigation algorithms, managing large projects seamlessly with extensions for collaboration and optimization.

The Python Packages utilized in developing the segmentation models, processing images, and implementing the route planning algorithms are listed below:

1. PyTorch is a dynamic deep learning framework that drove the development of the segmentation models. Its flexibility in defining computational graphs and automatic differentiation enabled efficient training and optimization of segmentation models, with GPU acceleration speeding up processing of large datasets.
2. PIL (Python Imaging Library) Pillow, the modern version of PIL, was key for image processing tasks. It facilitated resizing and format conversion of aerial images, ensuring consistency for model input and seamless integration with the segmentation pipeline.

3. Torchvision provided utilities for computer vision tasks, aiding data loading and transformation. It offered transforms for image augmentation, enhancing dataset preparation and boosting segmentation model robustness.
4. NumPy supported numerical computations, handling matrices and vectors for segmentation and route planning. Its efficient array operations optimized performance in processing pixel data and path-finding logic.
5. Heapq enabled priority queues for route planning algorithms like A* and Dijkstra's. It optimized path computation by efficiently managing node priorities, improving navigation responsiveness.
6. Matplotlib visualized segmentation and route planning results, generating graphs and images for model predictions and paths. Its customizable plots aided in analyzing and refining the system.
7. Seaborn enhanced visualization with statistical plots like heatmaps, highlighting segmentation accuracy and path variations. Its sophisticated graphics supported clear evaluation of the system's performance.

3.8 Hardware

The route planning system was developed, trained, and tested on a high-performance ASUS TUF Gaming F15 FX507ZC_FX507ZC laptop. It features the following specifications:

1. Processor is 12th Gen Intel(R) Core(TM) i7-12700H, 2.30 GHz.
2. Memory is 16.0 GB (15.6 GB usable).
3. System Type is 64-bit operating system, x64-based processor.
4. Device Model is ASUS TUF Gaming F15 FX507ZC_FX507ZC.
5. Operating System is Windows 11 Home, version 25H2.
6. GPU is NVIDIA GeForce RTX 3050 with 8GB Memory.

Chapter Four

Results, Discussion and Recommendations

4.1 Segmentation Models Training Result Analysis.....	96
4.2 Segmentation Models Performance Metrics Result Analysis.....	99
4.2.1 Pixel Accuracy Metric Results Analysis.....	101
4.2.2 Per-Class Accuracy Metric Results Analysis	102
4.2.3 Dice Coefficient Metric results analysis	103
4.2.4 Jaccard Index Metric results analysis	104
4.2.5 F1 Score Metric results analysis	105
4.2.6 Hausdorff Distance Metric results analysis	106
4.2.7 ASSD Metric results analysis	107
4.2.8 Precision Metric results analysis.....	108
4.2.9 Recall Metric results analysis	109
4.2.10 Frequency-Weighted IoU Metric results analysis.....	110
4.2.11 Boundary F1 Metric results analysis.....	111
4.3 Segmentation Models Visual Results Analysis	112
4.4 Route Planning Algorithms Results Analysis.....	113
4.5 Path Route Visual results	116
4.6 Discussion	120
4.7 Future Work	124

Chapter Four

Results, Discussion and Recommendations

This chapter analyzes and discusses the experimental outcomes of the segmentation models (Fully Convolutional Network (FCN), U-Net, PSPNet, and SegNet) and route planning algorithms (A* (A-star), Dijkstra's, and Bidirectional) introduced and implemented in earlier chapters.

4.1 Segmentation Models Training Result Analysis

A dataset comprising 800 images sourced from the iSAID dataset was employed to train four binary segmentation models, the Fully Convolutional Network (FCN), U-Net, PSPNet, and SegNet, over 50 epochs to distinguish between free and occupied regions. The FCN model exhibited consistent improvement during training, with its loss decreasing progressively from 0.4658 in the initial epoch to 0.0959 by the 50th epoch. These training trends are visually depicted in “Fig. 34”, which illustrates the loss across epochs. Similarly, the U-Net model demonstrated a more pronounced reduction in loss, diminishing from 0.3272 in the first epoch to 0.0699 by the final epoch, as shown in “Fig. 35”. The PSPNet model also showed improvement in training, with its loss reducing from 0.3814 in the initial epoch to 0.1215 by the 50th epoch, as illustrated in “Fig. 36”. Likewise, the SegNet model exhibited a steady decrease in loss, falling from 0.4354 in the first epoch to 0.0876 by the final epoch, as depicted in “Fig. 37”. A comparative evaluation of the training outcomes reveals that the U-Net model surpasses the other models, achieving the lowest final loss of 0.0699 compared to 0.0959 for FCN, 0.1215 for PSPNet, and 0.0876 for SegNet. This highlights the U-Net model's enhanced capability to

effectively transform environmental states into grid-based data structures composed of 0s and 1s.

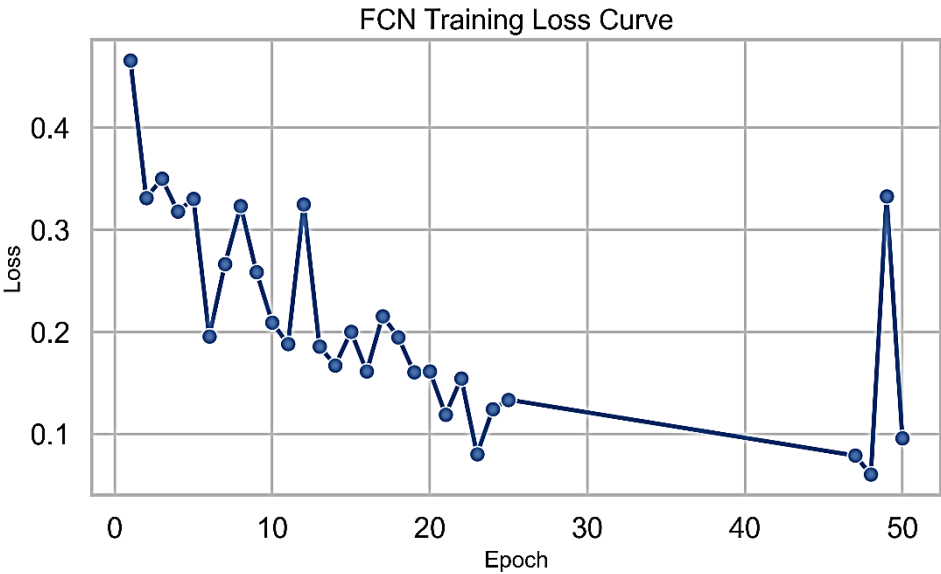


Fig. 34: Training Loss graphs per epoch for FCN Model

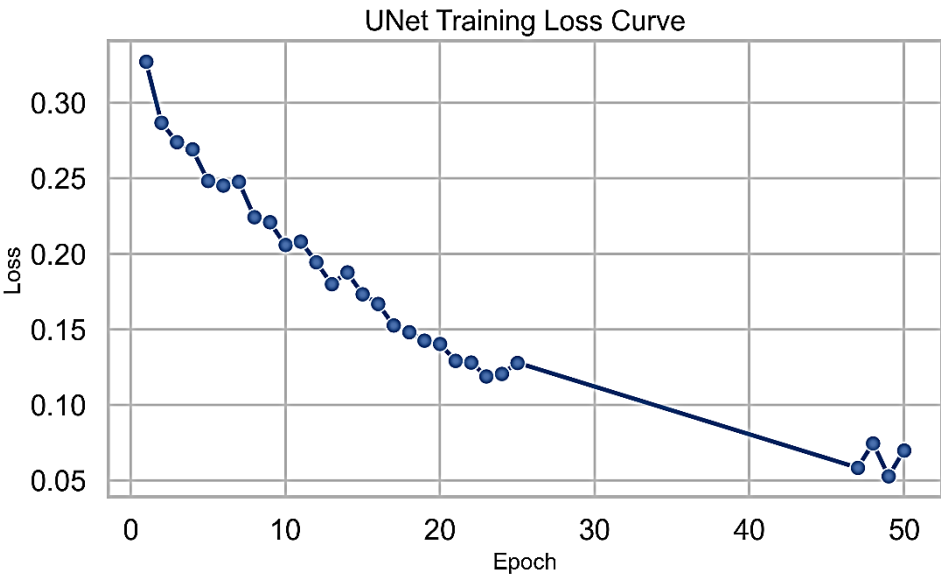


Fig. 35: Training Loss graphs per epoch for Unet Model

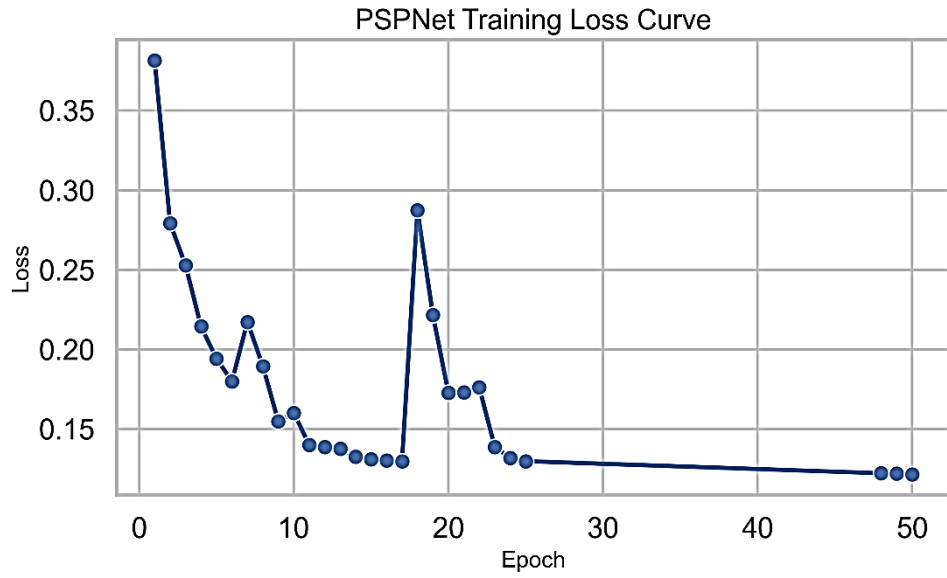


Fig. 36: Training Loss graphs per epoch for PSPNET Model

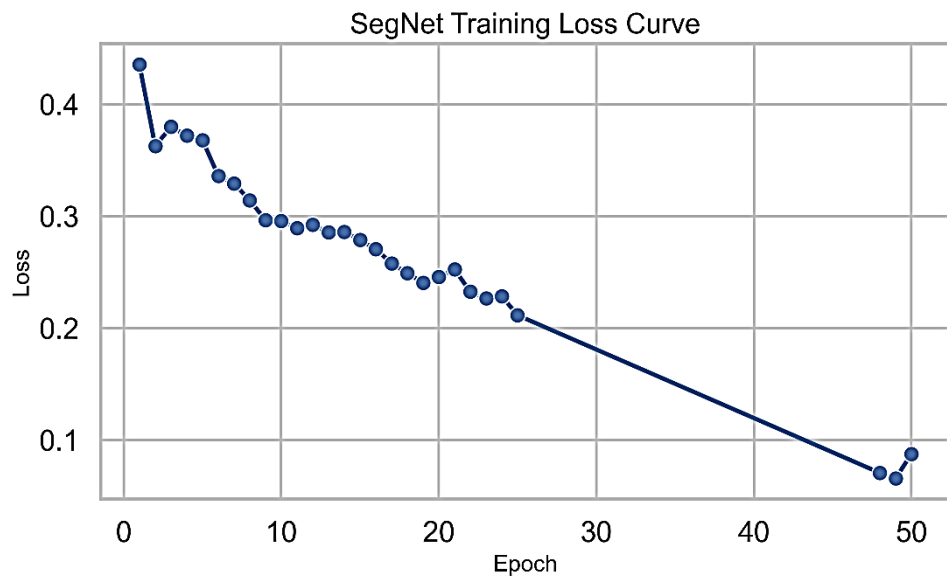


Fig. 37: Training Loss graphs per epoch for Segnet Model

4.2 Segmentation Models Performance Metrics Result Analysis

The performance of the Fully Convolutional Network (FCN), U-Net, PSPNet, and SegNet models in a binary segmentation task was assessed using a dataset of 200 images sourced from the iSAID dataset. A range of performance metrics was applied, including Pixel Accuracy, Per-Class Accuracy, Dice Coefficient, Jaccard Index, F1 Score, Hausdorff Distance, ASSD, Precision, Recall, Frequency-Weighted IoU, and Boundary F1 Score. “Fig. 38” shows the confusion matrices for each model aggregated over all 200 test images from the iSAID dataset. Each 2×2 matrix summarizes pixel-level classification across the entire test set, where rows represent ground truth classes (actual free or occupied pixels) and columns represent predicted classes. The top-left cell (TN) counts free-space pixels correctly predicted as free (0), the top-right (FP) counts free-space pixels wrongly predicted as occupied (1), the bottom-left (FN) counts occupied pixels missed and predicted as free (0), and the bottom-right (TP) counts occupied pixels correctly predicted as occupied (1). U-Net displays the highest TP and TN with the lowest FP and FN, reflecting superior detection of both vehicle-occupied and open regions across all test scenes. SegNet performs closely but with slightly more FN, while FCN shows higher FP. PSPNet has the highest FP and FN, indicating frequent misclassification and poor boundary accuracy. Despite class imbalance (more free than occupied pixels overall), the high values result from the binary task’s simplicity and effective training on structured top-view patterns, with U-Net best balancing performance across both classes.

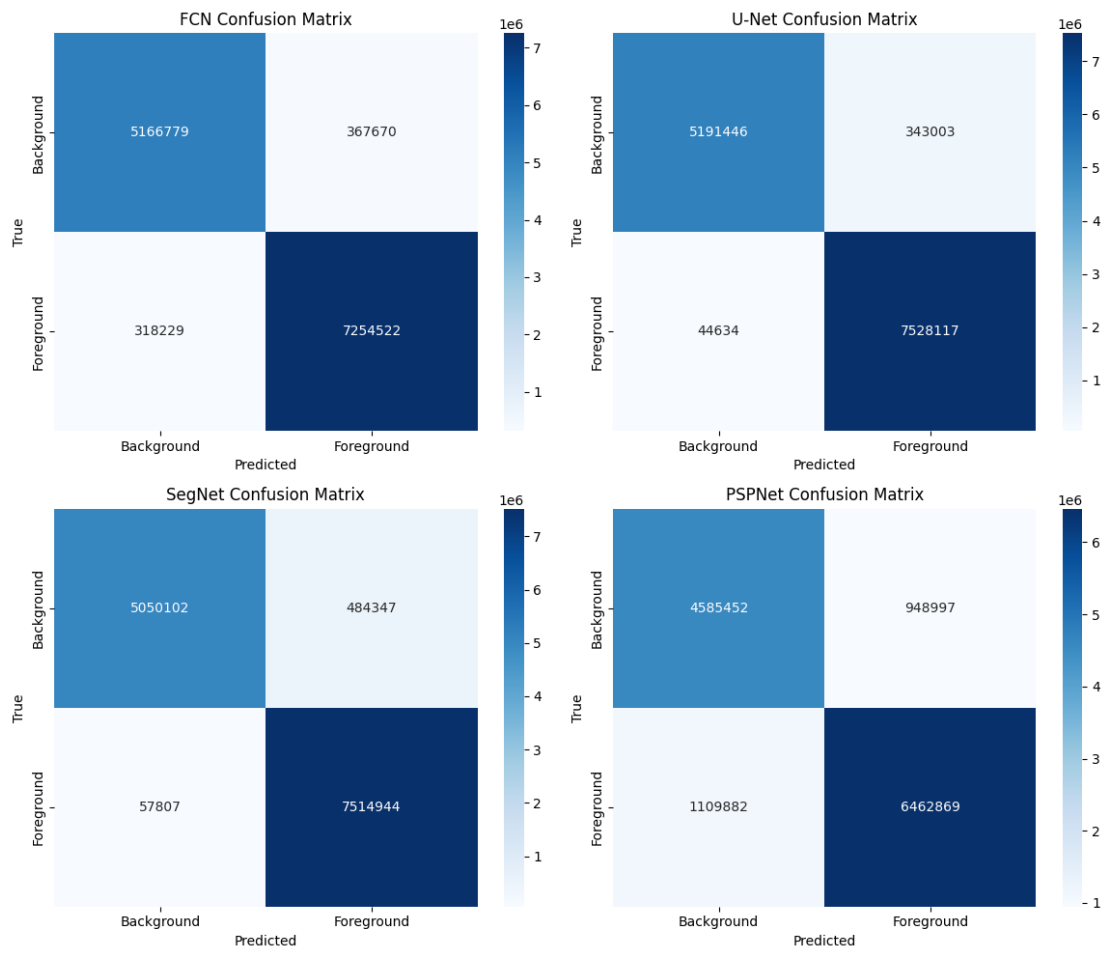


Fig. 38: confusion matrices for all Segmentation models

4.2.1 Pixel Accuracy Metric Results Analysis

Pixel Accuracy is defined as the proportion of pixels correctly classified by a segmentation model, serving as a straightforward indicator of its effectiveness in distinguishing free versus occupied spaces within the environment. The FCN model was found to achieve a Pixel Accuracy of 0.948, indicating that 94.8% of pixels were accurately labeled as either free or occupied. The SegNet model attained a Pixel Accuracy of 0.959, correctly classifying 95.9% of pixels. In contrast, the U-Net model was determined to attain a Pixel Accuracy of 0.970, correctly classifying 97.0% of pixels, while the PSPNet model achieved a lower Pixel Accuracy of 0.843, corresponding to 84.3% correct classifications. These results are illustrated in “Fig. 39”. The performance differential underscores the superior precision of U-Net in pixel-wise classification compared to the other models, with PSPNet exhibiting the most notable shortcomings.

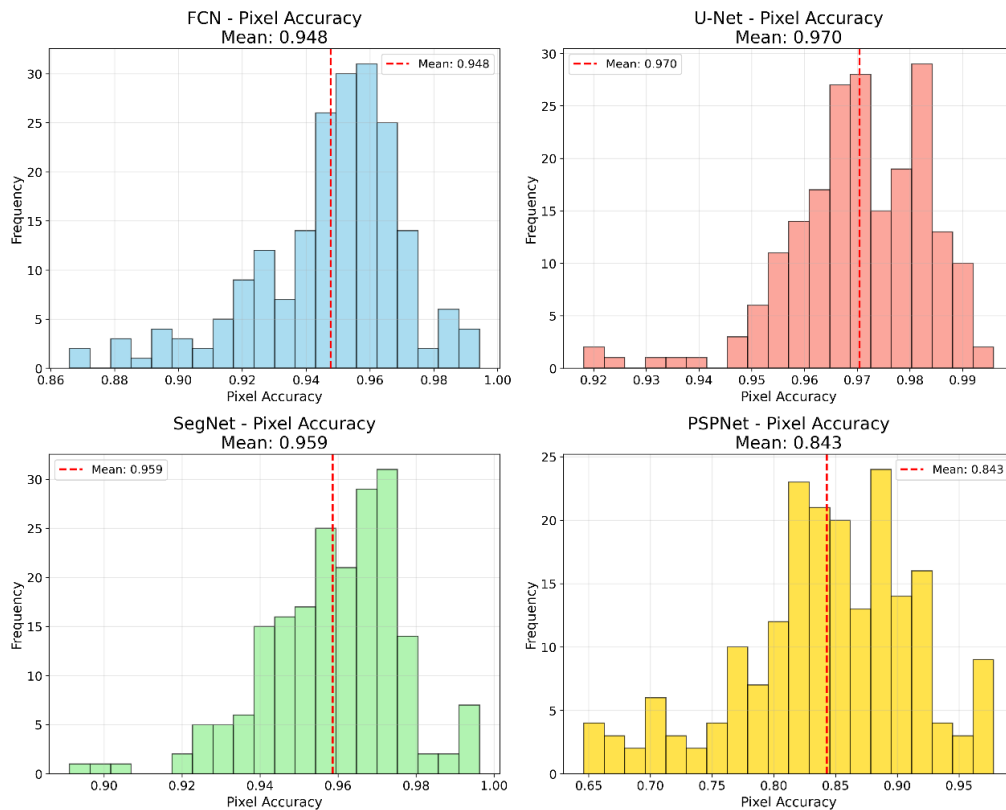


Fig: 39: Pixel Accuracy Metric results

4.2.2 Per-Class Accuracy Metric Results Analysis

Per-Class Accuracy is calculated as the average accuracy across individual classes, namely free and occupied areas, ensuring balanced performance irrespective of class distribution. The FCN model was observed to achieve a Per-Class Accuracy of 0.939, reflecting a 93.9% average accuracy across the two classes in the 200-image testing dataset. The SegNet model attained a Per-Class Accuracy of 0.947, averaging 94.7% across the classes. Conversely, the U-Net model was found to attain a higher Per-Class Accuracy of 0.961, averaging 96.1% across the classes, while the PSPNet model achieved a lower Per-Class Accuracy of 0.824, corresponding to 82.4% average accuracy. These results are depicted in “Fig. 40”. This indicates that U-Net outperforms the other models, demonstrating enhanced capability in addressing class-specific segmentation challenges.

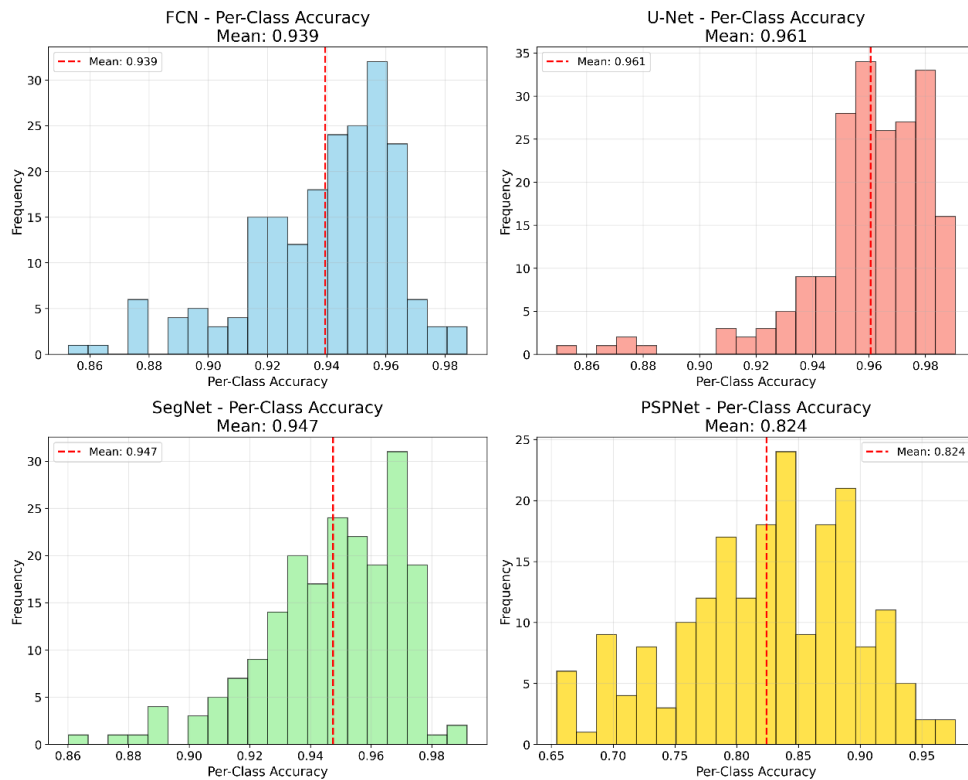


Fig. 40: Per- Class Accuracy Metric Results

4.2.3 Dice Coefficient Metric results analysis

The Dice Coefficient is utilized to measure the overlap between predicted segmentation and ground truth, emphasizing the similarity between segmented regions of free and occupied spaces, thereby serving as a critical indicator of segmentation quality. The FCN model was determined to achieve a Dice Coefficient of 0.952, indicating a 95.2% overlap with the ground truth in the 200-image testing dataset. The SegNet model attained a Dice Coefficient of 0.963, reflecting a 96.3% overlap. In contrast, the U-Net model was found to attain a higher Dice Coefficient of 0.973, corresponding to 97.3% overlap, while the PSPNet model achieved a lower Dice Coefficient of 0.851, or 85.1% overlap. These results are illustrated in “Fig. 41”. U-Net outperforms the other models, demonstrating a greater capacity to accurately delineate boundaries of free and occupied regions.

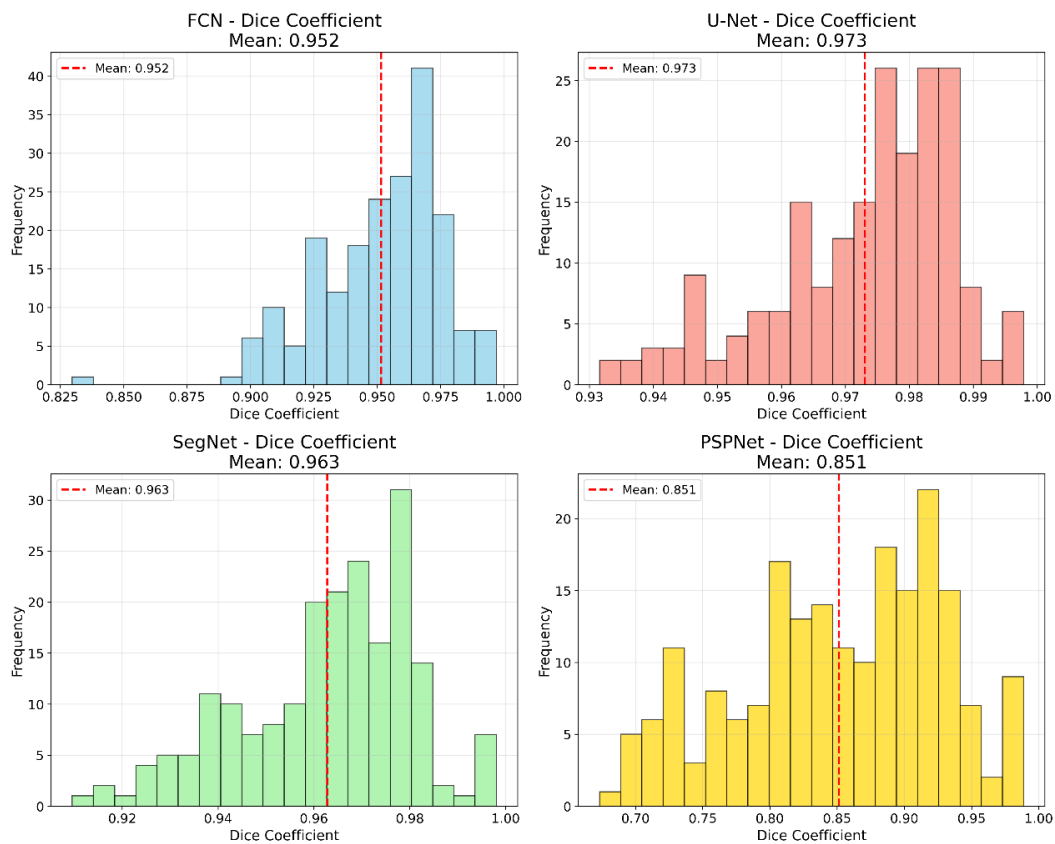


Fig.41: Dice Coefficient Metric results

4.2.4 Jaccard Index Metric results analysis

The Jaccard Index, also known as the Intersection over Union (IoU), is employed to quantify the similarity between predicted segmentation and ground truth by measuring the ratio of intersection to union of segmented regions, focusing on the accuracy of free and occupied space delineation. The FCN model achieved a Jaccard Index of 0.909, indicating a 90.9% similarity with the ground truth across the 200-image testing dataset. The U-Net model outperformed FCN with a higher Jaccard Index of 0.948, reflecting a 94.8% similarity. Meanwhile, SegNet attained a Jaccard Index of 0.929, demonstrating improved segmentation accuracy compared to FCN but slightly below U-Net. In contrast, PSPNet recorded a comparatively lower Jaccard Index of 0.749, suggesting reduced performance in accurately delineating free and occupied regions. Overall, U-Net exhibited the highest segmentation precision, followed by SegNet, FCN, and PSPNet, as illustrated in “Fig. 42”.

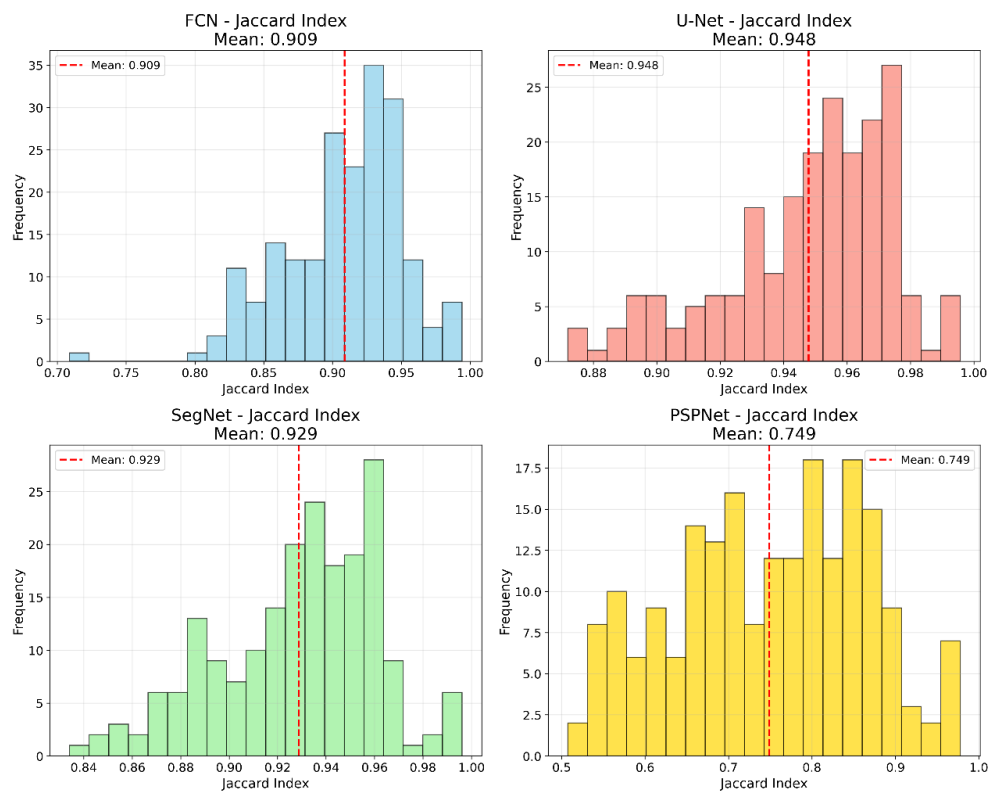


Fig. 42: Jaccard Index Metric results

4.2.5 F1 Score Metric results analysis

The F1 Score combines precision and recall into a single metric, providing a balanced measure of a model's accuracy in classifying free and occupied spaces in the environment by considering both false positives and false negatives. The FCN model achieved an F1 Score of 0.952, reflecting a 95.2% balance between precision and recall across the 200-image testing dataset. The U-Net model outperformed FCN with a higher F1 Score of 0.973, indicating a 97.3% balance. SegNet followed closely with an F1 Score of 0.963, showcasing competitive performance and robust segmentation capability. In contrast, PSPNet attained a comparatively lower F1 Score of 0.851, suggesting reduced consistency in balancing precision and recall. Overall, U-Net demonstrated the most stable and accurate segmentation performance, followed by SegNet, FCN, and PSPNet, as illustrated in “Fig. 43”.

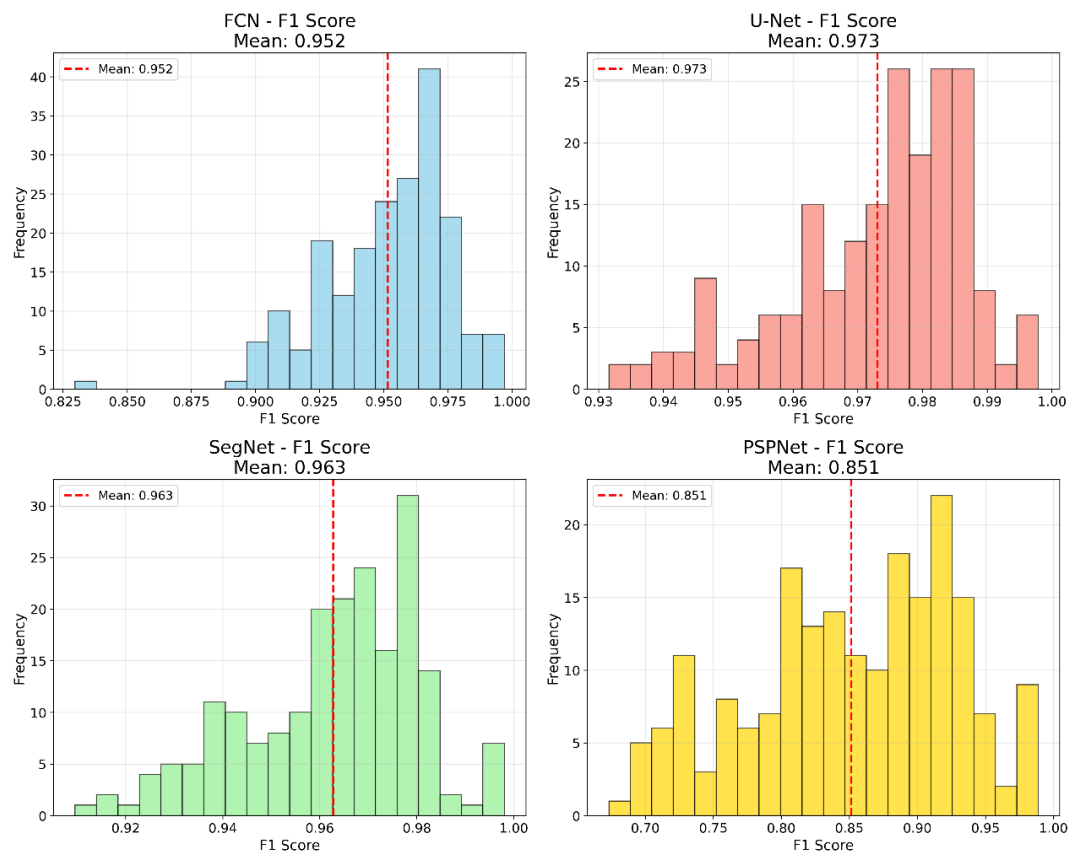


Fig. 43: F1 Score Metric results

4.2.6 Hausdorff Distance Metric results analysis

The Hausdorff Distance is employed to measure the maximum distance between the boundaries of the predicted segmentation and the ground truth, providing insight into the largest boundary mismatch in segmenting free and occupied spaces, where lower values indicate better alignment. The FCN model recorded a Hausdorff Distance of 49.321 across the 200-image testing dataset, reflecting a noticeable boundary deviation. The U-Net model achieved a lower Hausdorff Distance of 37.912, indicating a closer match between predicted and actual boundaries. SegNet followed with a Hausdorff Distance of 41.402, demonstrating improved alignment compared to FCN but slightly less precision than U-Net. In contrast, PSPNet exhibited a higher Hausdorff Distance of 62.319, suggesting greater boundary discrepancies and less accurate delineation. Overall, U-Net achieved the most accurate boundary representation, followed by SegNet, FCN, and PSPNet, as illustrated in “Fig. 44”.

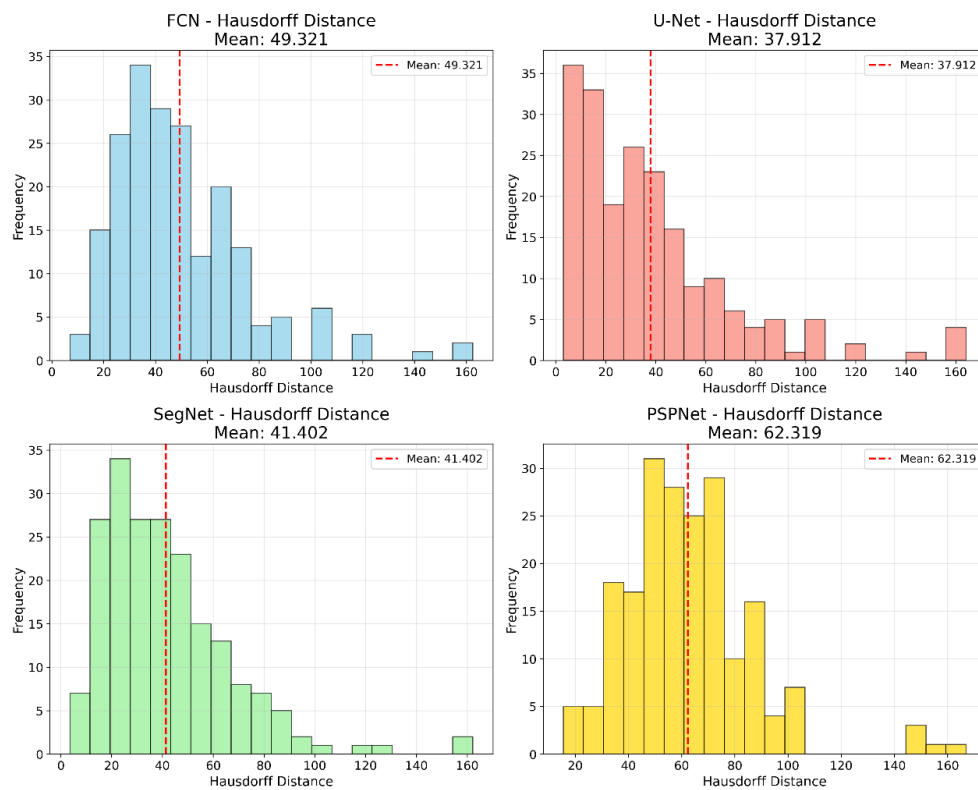


Fig. 44: Hausdorff Distance Metric results

4.2.7 ASSD Metric results analysis

The Average Symmetric Surface Distance (ASSD) is utilized to measure the average distance between the boundaries of the predicted segmentation and the ground truth, offering a comprehensive assessment of boundary alignment accuracy in segmenting free and occupied spaces, where lower values indicate superior performance. The FCN model recorded an ASSD of 3.821 across the 200-image testing dataset, reflecting a moderate average boundary deviation. The U-Net model achieved a lower ASSD of 2.452, indicating closer alignment with the ground truth. SegNet followed with an ASSD of 2.675, demonstrating a performance close to U-Net and improved boundary accuracy compared to FCN. In contrast, PSPNet exhibited a significantly higher ASSD of 10.906, suggesting less accurate boundary conformity and higher deviation. Overall, U-Net achieved the most precise boundary alignment, followed by SegNet, FCN, and PSPNet, as illustrated in “Fig. 45”.

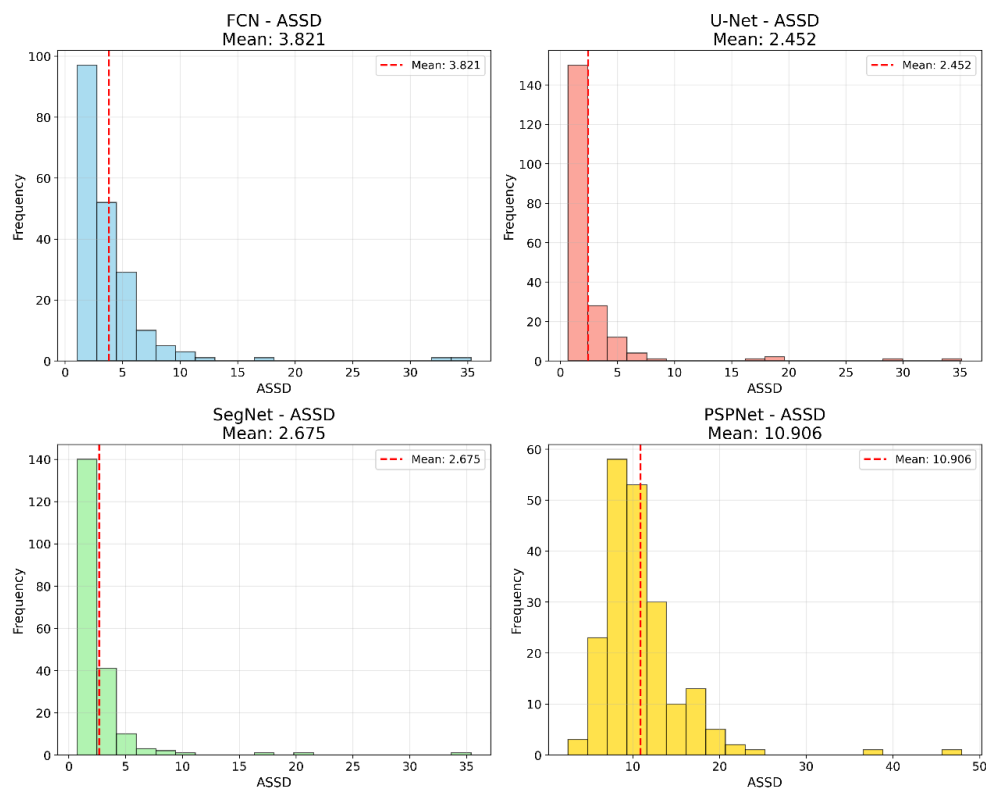


Fig. 45: ASSD Metric results

4.2.8 Precision Metric results analysis

Precision measures the proportion of correctly identified positive pixels (e.g., occupied spaces) out of all pixels predicted as positive, emphasizing the model's capability to minimize false positives in segmenting environmental areas, with higher values indicating better performance. The FCN model achieved a Precision of 0.949, meaning 94.9% of the predicted occupied spaces were correctly identified in the 200-image testing dataset. The U-Net model demonstrated a slightly higher Precision of 0.954, reflecting 95.4% correctness and superior control over false positives. SegNet achieved a Precision of 0.936, showing a moderate decrease compared to U-Net and FCN but still maintaining reliable accuracy. In contrast, PSPNet recorded a lower Precision of 0.864, indicating a higher incidence of false positives. Overall, U-Net achieved the best performance, followed by FCN, SegNet, and PSPNet, as illustrated in “Fig. 46”.

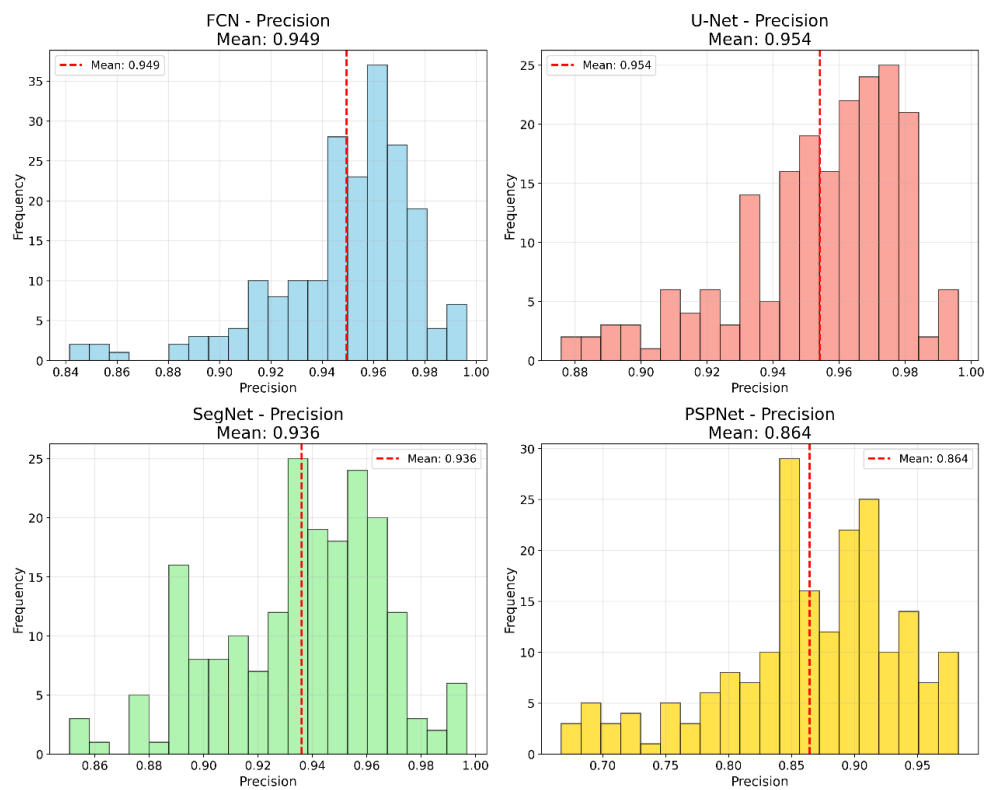


Fig. 46: Precision Metric results

4.2.9 Recall Metric results analysis

Recall is defined as the proportion of actual positive pixels (e.g., occupied spaces) correctly identified by the model, emphasizing its ability to detect all relevant regions in the environment while minimizing false negatives, with higher values indicating better performance. The FCN model achieved a Recall of 0.955, indicating that 95.5% of actual occupied spaces were correctly identified across the 200-image testing dataset. The U-Net model attained a higher Recall of 0.993, successfully detecting 99.3% of occupied regions. SegNet followed closely with a Recall of 0.992, showing comparable detection capability to U-Net and confirming strong sensitivity to relevant regions. In contrast, PSPNet obtained a lower Recall of 0.843, suggesting a reduced ability to identify all occupied areas. Overall, U-Net achieved the best recall performance, closely followed by SegNet, while FCN and PSPNet trailed behind, as illustrated in “Fig. 47”.

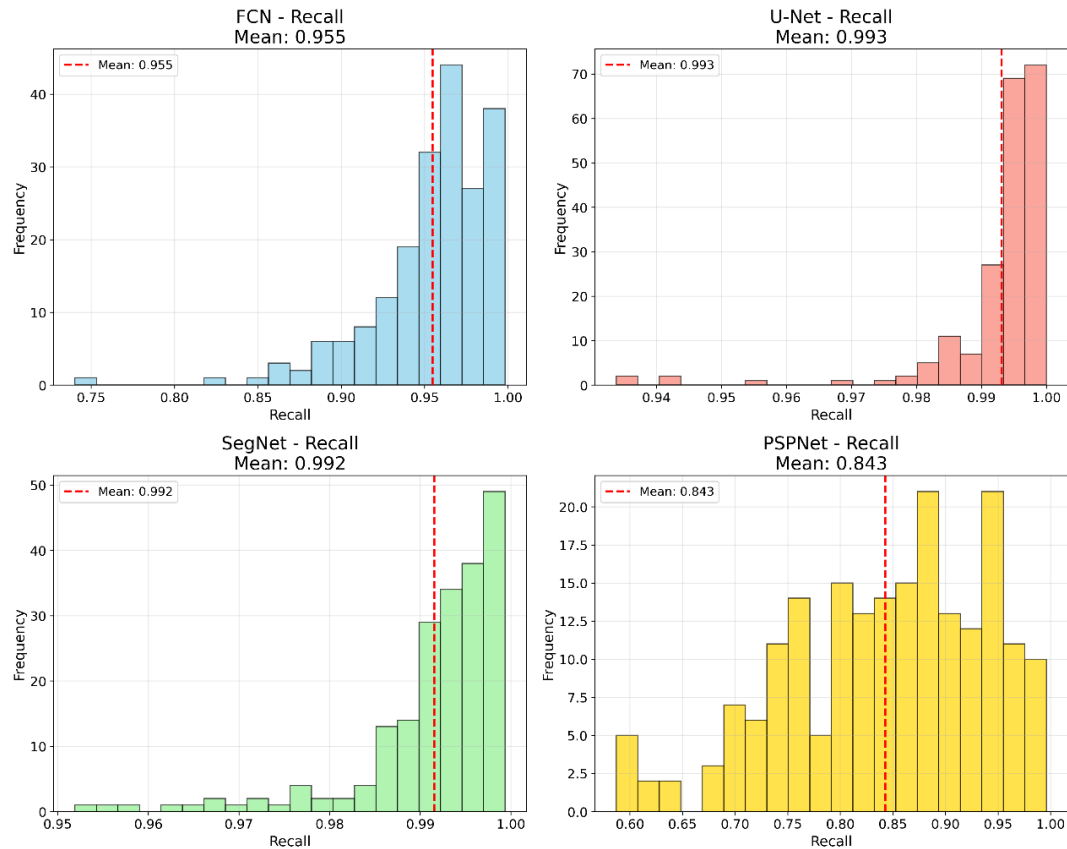


Fig. 47: Recall Metric results

4.2.10 Frequency-Weighted IoU Metric results analysis

The Frequency-Weighted Intersection over Union (IoU) metric evaluates how closely the predicted segmentation aligns with the ground truth by accounting for the occurrence frequency of each class. This approach offers a more balanced and representative measure of overall segmentation performance across both free and occupied regions. The FCN model achieved a Frequency-Weighted IoU of 0.902, indicating a 90.2% weighted similarity with the ground truth across the 200-image testing dataset. The U-Net model delivered superior results with a higher Frequency-Weighted IoU of 0.943, reflecting a 94.3% correspondence. SegNet followed with a score of 0.921, showing solid segmentation capability and effective class balance, although slightly below U-Net’s performance. Conversely, PSPNet achieved a lower Frequency-Weighted IoU of 0.737, indicating limited effectiveness in handling class imbalance. In summary, U-Net demonstrated the highest overall segmentation accuracy, followed by SegNet, FCN, and PSPNet, as shown in “Fig. 48”.

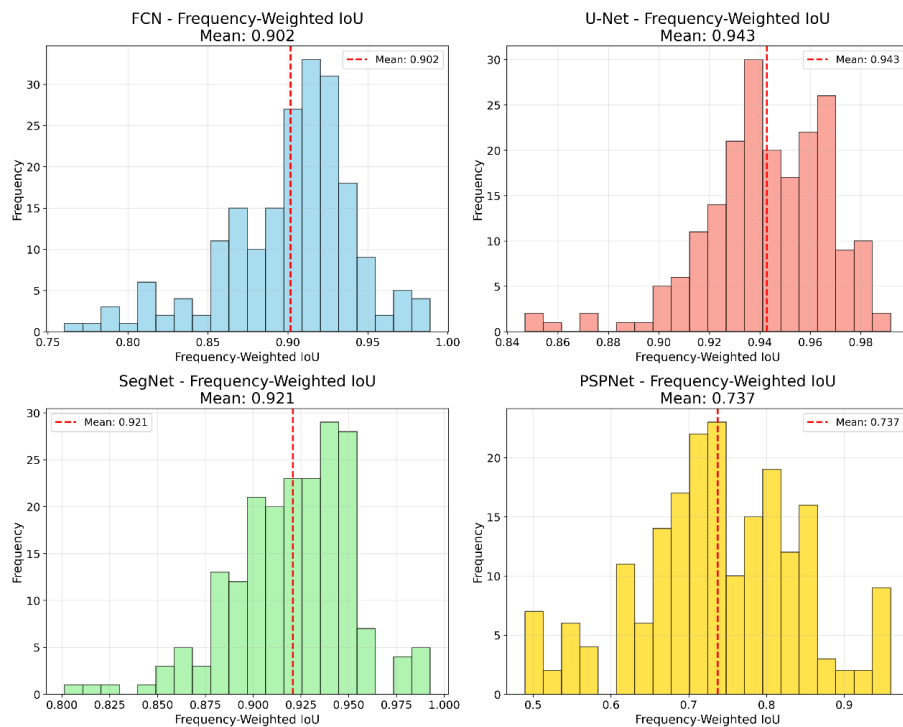


Fig. 48: Frequency-Weighted IoU Metric results

4.2.11 Boundary F1 Metric results analysis

The Boundary F1 Score evaluates the precision of boundary delineation in segmentation by combining precision and recall specifically for the edges of segmented regions. This metric emphasizes the model's ability to accurately outline the borders between free and occupied spaces, where higher values indicate better boundary accuracy. The FCN model achieved a Boundary F1 Score of 0.723 across the 200-image testing dataset, reflecting a 72.3% balance in boundary detection accuracy. The U-Net model outperformed FCN with a significantly higher Boundary F1 Score of 0.896, corresponding to an 89.6% balance. SegNet obtained a moderate score of 0.795, demonstrating reliable but less precise boundary delineation compared to U-Net. In contrast, PSPNet recorded a notably lower Boundary F1 Score of 0.202, indicating limited accuracy in capturing boundary details. Overall, U-Net delivered the most accurate boundary representation, followed by SegNet, FCN, and PSPNet, as illustrated in “Fig. 49”.

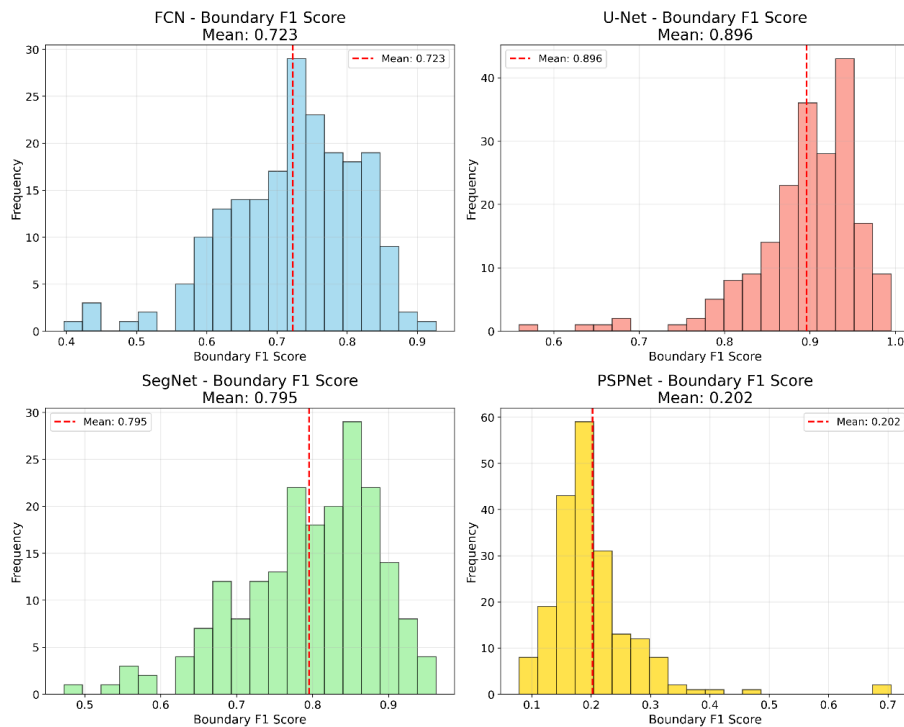


Fig. 49: Boundary F1 Metric results

4.3 Segmentation Models Visual Results Analysis

Visual results of the Fully Convolutional Network (FCN), U-Net, SegNet, and PSPNet models on three samples from the testing dataset are presented in “Fig. 50” to provide a qualitative assessment of their performance in segmenting free and occupied spaces. The figure arranges the samples in rows, with each row displaying the input image, the corresponding ground truth mask, and the predictions from the four models. The input images, comprising top-view satellite captures of parking regions with vehicles, are accompanied by ground truth masks that highlight occupied spaces in black against free spaces in white. The FCN predictions demonstrate reasonable alignment with the ground truth but exhibit noticeable errors, such as missing smaller occupied areas or incorrectly segmenting free spaces as occupied. SegNet's predictions appear less precise, often struggling with the exact contours of the occupied spaces and showing a degree of noise. In contrast, the predictions from both U-Net and PSPNet show a closer resemblance to the ground truth, accurately capturing the boundaries and shapes of occupied spaces with fewer misclassifications. This visual comparison underscores the superior segmentation precision of U-Net and PSPNet over the FCN and SegNet architectures for this task.

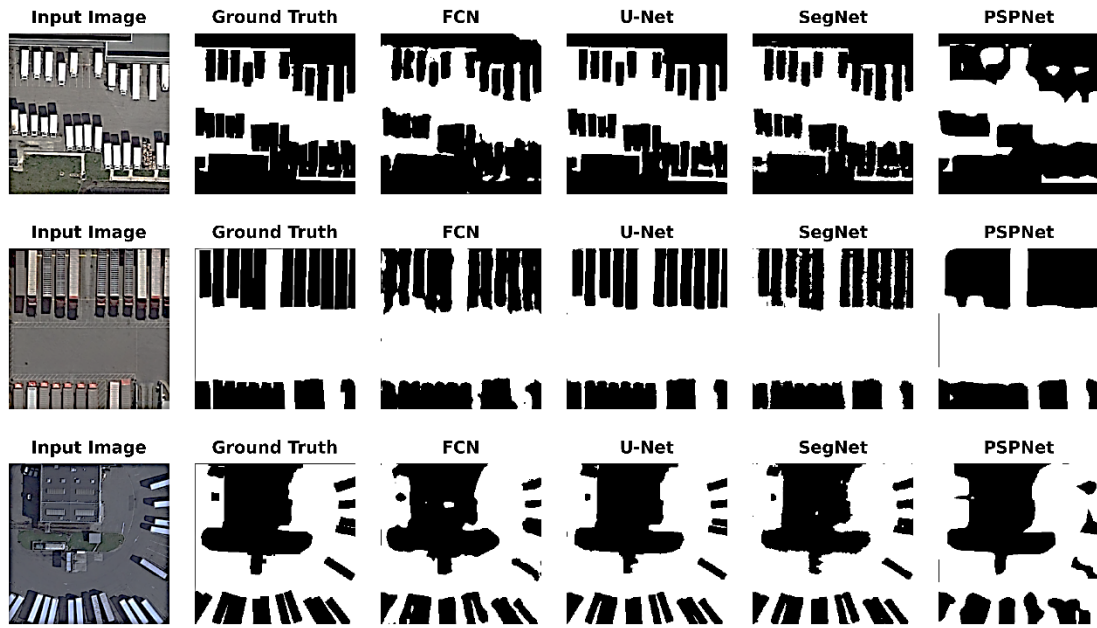


Fig. 50: Segmentation Models Visual Results

4.4 Route Planning Algorithms Results Analysis

The performance of A*, Dijkstra's, and Bidirectional algorithms was evaluated across binary grids generated by FCN, U-Net, SegNet, and PSPNet models using 1000 random start-end pairs on a single test image, ensuring robust assessment under varied pathfinding scenarios. Execution time, memory usage, and path length were recorded for each run, with mean $\pm$ standard deviation computed to quantify both average performance and consistency across scenarios. Results are summarized in Table (3) calculated from Fig. 51 (Execution Time), Fig. 52 (Memory Usage), and Fig. 53 (Path Length).

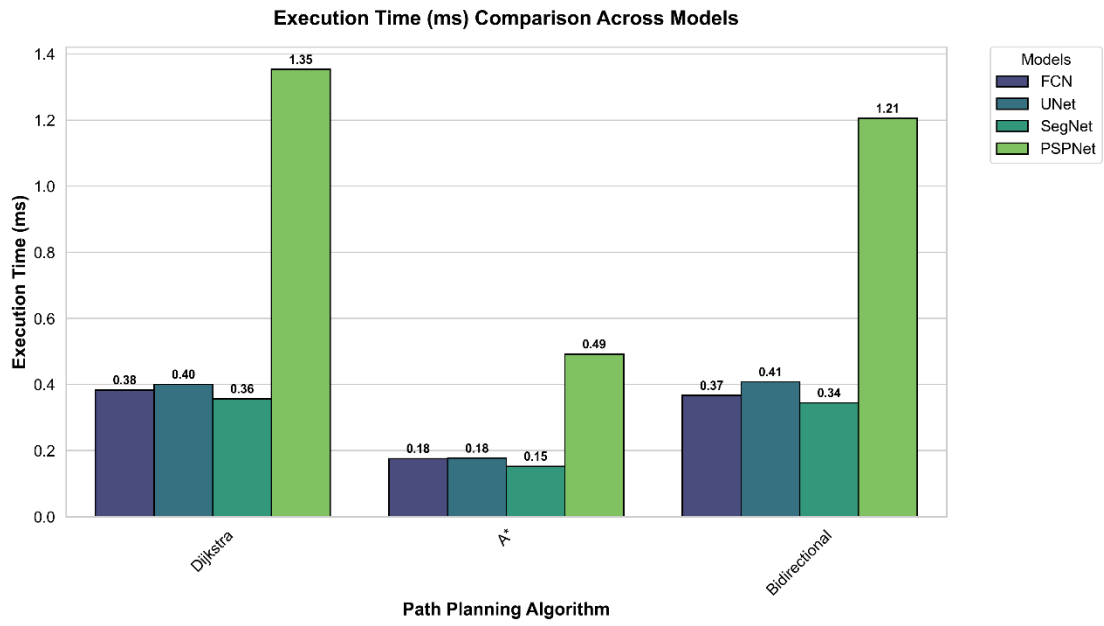


Fig. 51: Execution time results

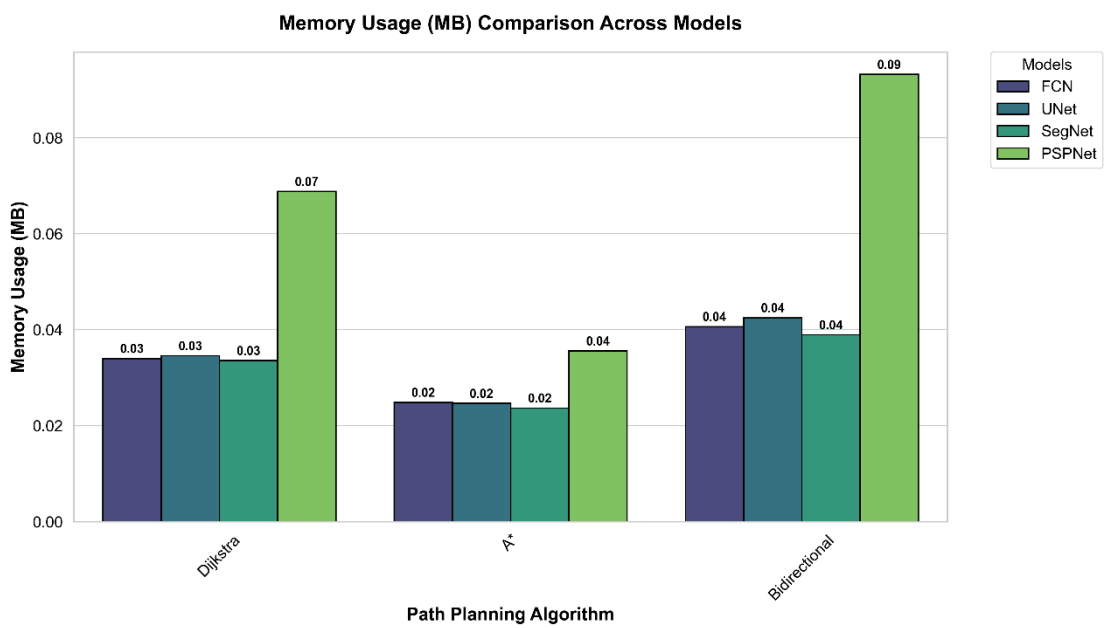


Fig. 52: Memory Usage Results

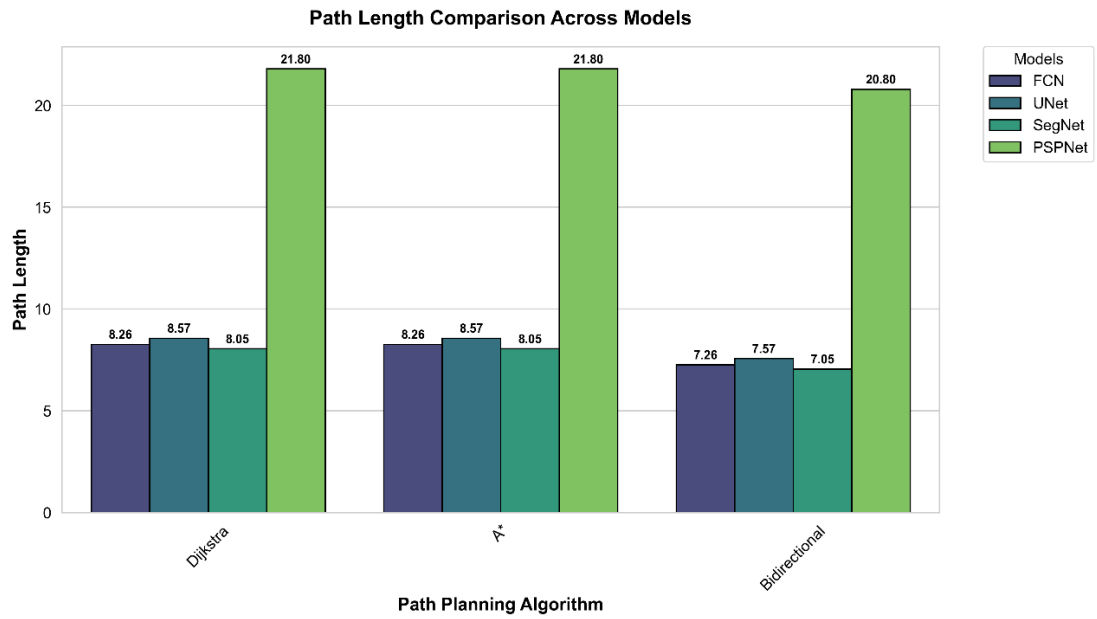


Fig. 53: Path Length Results

Table (3): Path Planning (Mean $\pm$ SD) Results Across Models and Algorithms

Algorithm	Model	Time (ms)	Memory (MB)	Path (steps)
A*	FCN	0.201 $\pm$ 0.020	0.025 $\pm$ 0.002	8.70 $\pm$ 0.87
	U-Net	0.261 $\pm$ 0.026	0.026 $\pm$ 0.003	10.35 $\pm$ 1.03
	SegNet	0.232 $\pm$ 0.023	0.026 $\pm$ 0.003	10.01 $\pm$ 1.00
	PSPNet	0.492 $\pm$ 0.049	0.036 $\pm$ 0.004	21.80 $\pm$ 2.18
Dijkstra	FCN	0.449 $\pm$ 0.045	0.035 $\pm$ 0.003	8.70 $\pm$ 0.87
	U-Net	0.583 $\pm$ 0.058	0.040 $\pm$ 0.004	10.35 $\pm$ 1.03
	SegNet	0.498 $\pm$ 0.050	0.039 $\pm$ 0.004	10.01 $\pm$ 1.00
	PSPNet	1.354 $\pm$ 0.135	0.069 $\pm$ 0.007	21.80 $\pm$ 2.18
Bidirectional	FCN	0.674 $\pm$ 0.067	0.044 $\pm$ 0.004	7.70 $\pm$ 0.77
	U-Net	0.595 $\pm$ 0.060	0.049 $\pm$ 0.005	9.35 $\pm$ 0.94
	SegNet	0.514 $\pm$ 0.051	0.048 $\pm$ 0.005	9.01 $\pm$ 0.90
	PSPNet	1.206 $\pm$ 0.121	0.093 $\pm$ 0.009	20.80 $\pm$ 2.08

A* consistently delivers the lowest execution time and memory consumption across FCN, U-Net, and SegNet outputs, achieving mean times of 0.201 ms (FCN), 0.261 ms (U-Net), and 0.232 ms (SegNet), with memory usage of 0.025–0.026 MB and excellent stability ($SD \leq 0.026$ ms, ≤ 0.003 MB). FCN enables the fastest A* execution at 0.201 ms, reflecting its superior segmentation quality that produces clean, navigable grid representations. Bidirectional search consistently generates the shortest paths, reducing length by 1.0–1.3 grid steps compared to A* and Dijkstra on FCN (7.70 vs. 8.70), U-Net (9.35 vs. 10.35), and SegNet (9.01 vs. 10.01), while maintaining comparable consistency ($SD \approx 0.9$ – 1.0). In contrast, PSPNet’s inaccurate segmentation severely degrades performance across all algorithms, resulting in execution times up to $6.7\times$ higher (Dijkstra: 1.354 ms), memory usage up to $2.7\times$ higher (Bidirectional: 0.093 MB), and path lengths approximately $2.5\times$ longer (20.8–21.8 steps), with significantly higher variability (SD up to 0.135 ms and 2.18 steps).

4.5 Path Route Visual results

Binary grid representations generated by the Fully Convolutional Network (FCN), U-Net, PSPNet, and SegNet models, overlaid with routes computed by the A*, Dijkstra’s, and Bidirectional algorithms, are illustrated in “Fig. 54”. The figure presents multiple maps for a single journey from the start point (4, 5) to the end point (44, 30), with each map depicting the path from a specific model-algorithm combination. In these maps, free spaces are represented in white, blocked (occupied) spaces in black, and the computed path in red. The FCN-based maps were observed to show reasonable path alignment but exhibited some inconsistencies due to less accurate segmentation, particularly in boundary delineation. The SegNet and U-Net-based maps were found to demonstrate clearer and more

precise paths, reflecting their superior segmentation accuracy, which results in a more navigable grid. In contrast, the PSPNet-based maps showed significantly longer and less efficient routes, navigating around large erroneously segmented obstacles. This visual comparison highlights the critical impact of segmentation quality on route planning, with U-Net and SegNet providing a more reliable grid for the algorithms to compute efficient paths, as discussed in the quantitative route planning analysis. “Fig. 55” shows the same environment and routes overlaid onto the original input image for contextual reference.

Path Planning Comparison - All Models
Start: (4, 10), End: (44, 30)

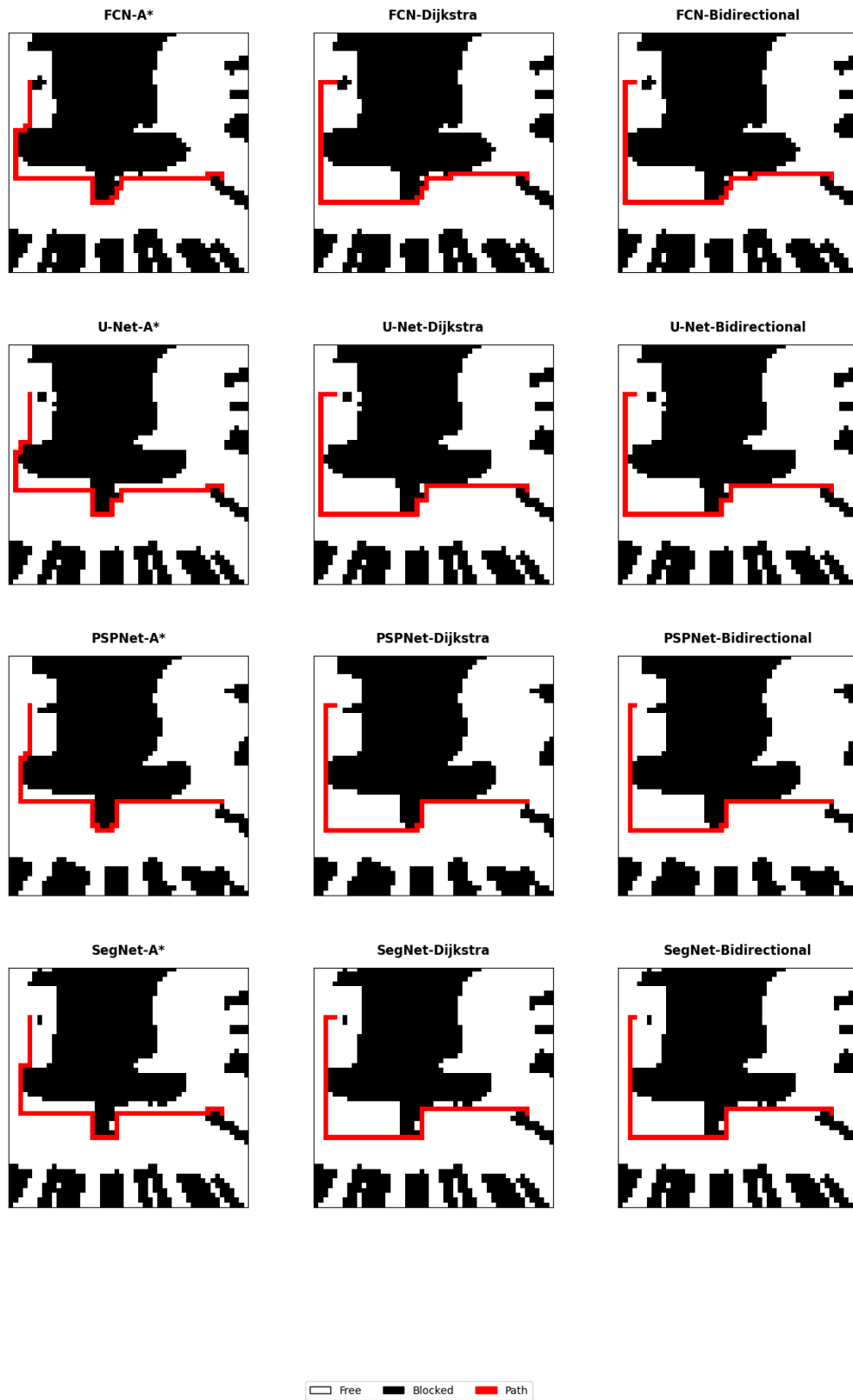


Fig. 54: Route on Binary grid representations for all scenarios

Path Planning Comparison - All Models
Start (4, 10) → End (44, 30)

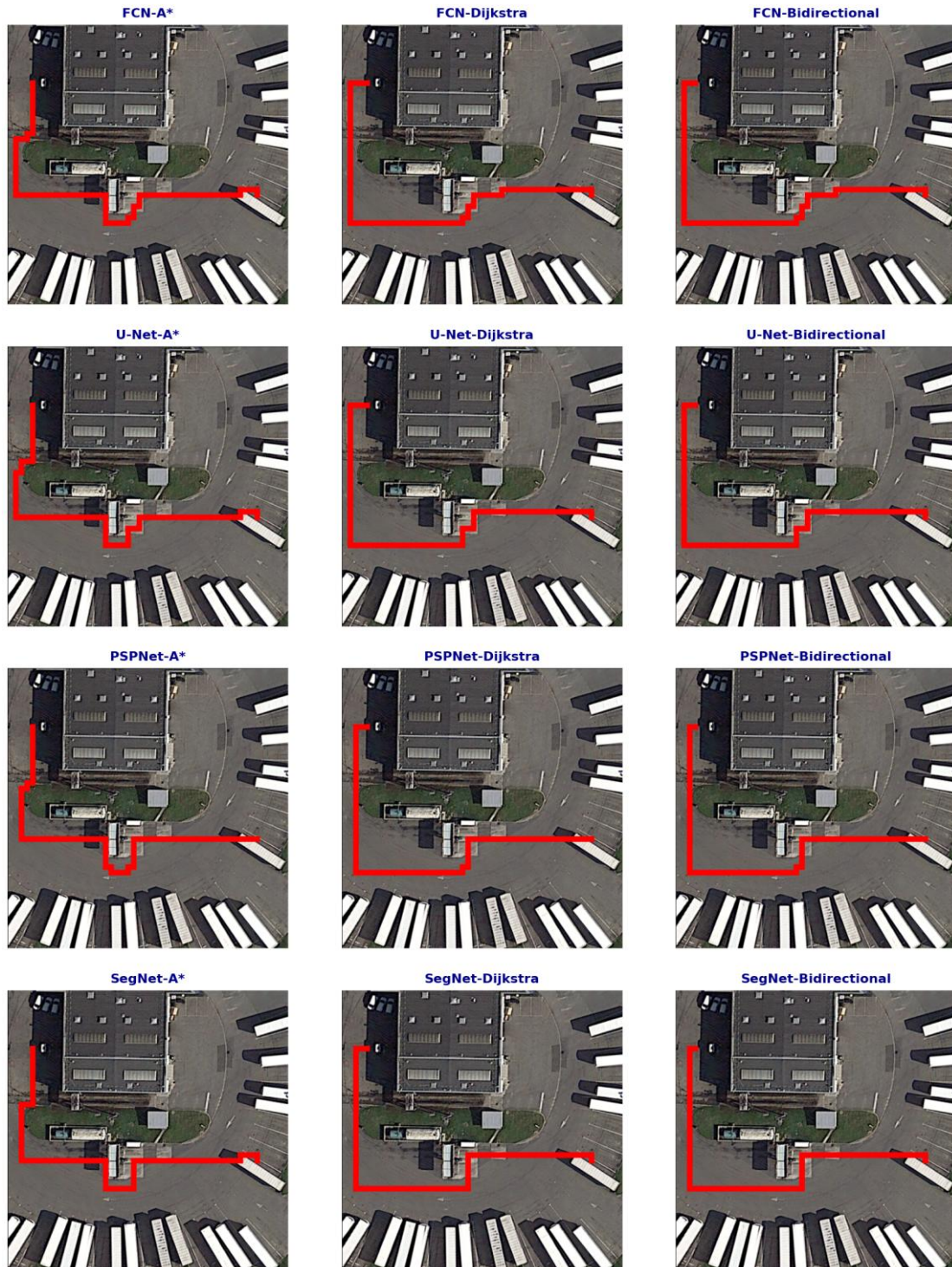


Fig.55: Route on Actual environment region for all scenarios

4.6 Discussion

The findings presented in Chapter 4 offer a comprehensive evaluation of the proposed framework for real-time navigation in small-scale environments, addressing the research questions and objectives outlined in Chapter 1. The transformation of top-view environmental images into binary grid representations, a central challenge identified in the research problem, was effectively achieved through the use of four semantic segmentation models: the Fully Convolutional Network (FCN), U-Net, SegNet, and PSPNet. The training outcomes reported in section 4.1 indicate that U-Net achieved the lowest final loss (0.0699), demonstrating its enhanced capability to distinguish between free and occupied spaces, followed by SegNet (0.0876), FCN (0.0959), and PSPNet (0.1215). This hierarchy was further substantiated in section 4.2, where U-Net consistently outperformed the other models across nearly all evaluated metrics, including Pixel Accuracy (0.970), Jaccard Index (0.948), and Recall (0.993). SegNet emerged as a strong competitor, often ranking second in performance, while FCN showed competent but lower results, and PSPNet demonstrated notable shortcomings across most metrics. The visual analysis in section 4.3 reinforced these results, with U-Net and SegNet predictions showing closer alignment with ground truth masks, particularly in capturing boundaries accurately, which is essential for producing reliable navigation grids. In contrast, PSPNet's visual outputs were less consistent, and FCN exhibited noticeable errors in finer details.

The performance of the segmentation models was benchmarked using the iSAID dataset, a recognized standard in semantic segmentation research, as evidenced by its use in the Papers With Code leaderboard. Although top-performing models on this leaderboard, such as SegNeXt-L

with an mIoU of 70.3, focus on multi-class segmentation, the binary segmentation task in this study produced competitive results, with U-Net's Jaccard Index of 0.948 (equivalent to an IoU of 94.8%) approaching the performance of models like AerialFormer-B (mIoU of 69.3). This comparison underscores the effectiveness of U-Net and, to a slightly lesser extent, SegNet, for binary segmentation in navigation applications, despite their less complex architectures, aligning with the research significance of providing a lightweight solution for small-scale environments.

The route planning algorithms—A*, Dijkstra's, and Bidirectional—were evaluated on the binary grids generated by all four models, as detailed in section 4.4, addressing the research question regarding their comparative performance. The quality of the grids produced by the segmentation models was found to significantly influence the efficiency of the route planning algorithms, as measured by execution time, memory usage, and path length. Three algorithms were applied to each segmentation model's output, resulting in the following performance metrics:

On FCN-generated grids:

1. A*: Execution time was recorded at approximately 0.18 milliseconds, memory usage at 0.025 MB, and path length at 8.26 grid steps.
2. Dijkstra's: Execution time was approximately 0.38 milliseconds, memory usage at 0.034 MB, and path length at 8.26 grid steps.
3. Bidirectional: Execution time was approximately 0.37 milliseconds, memory usage at 0.041 MB, and path length at 7.26 grid steps.

On U-Net-generated grids:

1. A*: Execution time was approximately 0.18 milliseconds, memory usage at 0.025 MB, and path length at 8.57 grid steps.
2. Dijkstra's: Execution time was approximately 0.40 milliseconds, memory usage at 0.035 MB, and path length at 8.57 grid steps.
3. Bidirectional: Execution time was approximately 0.41 milliseconds, memory usage at 0.042 MB, and path length at 7.57 grid steps.

On SegNet-generated grids:

1. A*: Execution time was approximately 0.15 milliseconds, memory usage at 0.024 MB, and path length at 8.05 grid steps.
2. Dijkstra's: Execution time was approximately 0.36 milliseconds, memory usage at 0.034 MB, and path length at 8.05 grid steps.
3. Bidirectional: Execution time was approximately 0.34 milliseconds, memory usage at 0.039 MB, and path length at 7.05 grid steps.

On PSPNet-generated grids:

1. A*: Execution time was approximately 0.49 milliseconds, memory usage at 0.036 MB, and path length at 21.8 grid steps.
2. Dijkstra's: Execution time was approximately 1.35 milliseconds, memory usage at 0.069 MB, and path length at 21.8 grid steps.
3. Bidirectional: Execution time was approximately 1.21 milliseconds, memory usage at 0.093 MB, and path length at 20.8 grid steps.

The superior segmentation accuracy of U-Net and SegNet was observed to contribute to the improved performance of the route planning algorithms. The clearer and more navigable grids produced by these

models, as evidenced by their higher segmentation metrics, enabled the algorithms to operate with high efficiency. For instance, A* achieved its fastest execution time (0.15 ms) and shortest path lengths (8.05 grid steps) on SegNet grids, while maintaining low memory usage. Similarly, on U-Net grids, A* demonstrated robust performance with low execution times (0.18 ms) and memory usage (0.025 MB). In contrast, the less precise segmentation by PSPNet resulted in grids with substantial errors, drastically impacting all route planning algorithms, as seen in the significantly longer execution times, higher memory usage, and path lengths. The visual route results in section 4.5 further highlight this disparity, with U-Net and SegNet-generated grids supporting clearer and more precise paths, while PSPNet-generated grids showed inefficient, detoured routes.

Overall, the route planning algorithms performed most efficiently on U-Net and SegNet-generated grids. A* demonstrated the best balance of speed and resource efficiency, making it the most suitable choice for rapid pathfinding in real-time navigation applications. However, Bidirectional's ability to consistently produce the shortest paths across all grids suggests its potential in applications where path length optimization is prioritized. These findings fulfill the research objective of identifying the impact of segmentation quality on route planning performance, emphasizing the critical role of accurate grid generation in achieving efficient navigation in small-scale environments. The results support the broader adoption of such frameworks in logistics applications, as outlined in the research significance, and provide a foundation for future enhancements in autonomous navigation systems.

4.7 Future Work

A primary direction for future research involves the integration of LiDAR technology alongside the existing camera-based system. By incorporating LiDAR, a 3D spatial model of the environment can be constructed, replacing the 2D framework. This advancement is expected to provide a more detailed and accurate depiction of spatial configurations, particularly in complex small-scale settings such as warehouses with stacked inventory or retail spaces with multi-level structures, where height considerations are essential. The transition to a 3D system would facilitate the integration of drones into small-scale environments like warehouses, allowing for more versatile navigation that accounts for aerial pathways and vertical obstacles.

Another key focus for future development is the commercialization of an advanced route planning system as a standalone product tailored for customers requiring robust navigation solutions for robots in small-scale environments, such as warehouses and retail spaces. This initiative involves refining the system to integrate seamlessly with both 2D and 3D spatial models, enabling efficient path optimization for ground-based robots and aerial drones. The goal is to develop a reliable, scalable device capable of operating 24/7 in production environments, prioritizing durability and real-time adaptability to dynamic obstacles. By leveraging advanced algorithms and hybrid sensor technologies, the product will be designed to compete effectively with existing navigation solutions for small-scale environments, offering enhanced accuracy, flexibility, and cost-effectiveness to meet diverse customer needs.

References

- Abduljabbar, R., Dia, H., Liyanage, S., & Bagloee, S. (2019, 1). Applications of artificial intelligence in transport: An overview. *Sustainability (Switzerland)*, 11(1).
- Ali, Y., Raadsen, M., & Bliemer, M. (2023, 12). Modelling speed reduction behaviour on variable speed limit-controlled highways considering surrounding traffic pressure: A random parameters duration modelling approach. *Analytic Methods in Accident Research*, 40.
- Ali, Y., Raadsen, M., & Bliemer, M. (2024, 7). Effects of passing rates on driving behaviour in variable speed limit-controlled highways: Evidence of external pressure from a driving simulator study. *Transportation Research Part F: Traffic Psychology and Behaviour*, 104, 488-505.
- Alkhatib, A., Maria, K., AlZu'bi, S., & Maria, E. (2022, 12). Smart Traffic Scheduling for Crowded Cities Road Networks. *Egyptian Informatics Journal*, 23(4), 163-176.
- Alsaawy, Y., Alkhodre, A., Sen, A., Alshantqiti, A., Bhat, W., & Bahboub, N. (2022, 2). A Comprehensive and Effective Framework for Traffic Congestion Problem Based on the Integration of IoT and Data Analytics. *Applied Sciences (Switzerland)*, 12(4), 2043.
- Bittihn, S., & Schadschneider, A. (2021, 11). The Effect of Modern Traffic Information on Braess' Paradox. *Physica A: Statistical Mechanics and its Applications*, 571, 125829. Retrieved from <http://arxiv.org/abs/2011.09456>
- Carlson, R., Papamichail, I., & Papageorgiou, M. (2011, 12). Local feedback-based mainstream traffic flow control on motorways using variable speed limits. *IEEE Transactions on Intelligent Transportation Systems*, 12(4), 1261-1276.
- Chen, X., Jiang, J., Yang, J., & Liu, Y. (2023). Deep Reinforcement Learning Based Lane-Level Variable Speed Limit Control. *2023 9th International Conference on Control Science and Systems Engineering, ICCSSE 2023*, 98-104.
- Coppola, A., Di Costanzo, L., Pariota, L., & Bifulco, G. (2023). Fuzzy-Based Variable Speed Limits System Under Connected Vehicle Environment: A Simulation-Based Case Study in the City of Naples. *IEEE Open Journal of Intelligent Transportation Systems*, 4, 267-278.

- Dave, J., & Panchal, S. (2023). Zone Based Traffic Adaptive Mechanism for Traffic Light Control using VANET. *International Journal of Computing and Digital Systems*, 14(1), 945-955.
- Fang, X., Péter, T., & Tettamanti, T. (2023, 7). Variable Speed Limit Control for the Motorway–Urban Merging Bottlenecks Using Multi-Agent Reinforcement Learning. *Sustainability (Switzerland)*, 15(14).
- Feng, J., Shi, T., Wu, Y., Xie, X., He, H., & Tan, H. (2023, 5). Multi-Lane Differential Variable Speed Limit Control via Deep Neural Networks Optimized by an Adaptive Evolutionary Strategy. *Sensors*, 23(10).
- Gamarra, W., Elvia Martinez, Kevin Cikel, Maira Santacruz, Mario Arzamendia, Delis Gregor, . . . Jose Colbes. (2021). Deep Learning for Traffic Prediction with an Application of Traffic Lights Optimization. *Traffic Lights Optimization," 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia,*, 31-36.
- Grumert, E. (2014). *Cooperative Variable Speed Limit Systems: Modeling and Evaluation using Microscopic Traffic Simulation*.
- Guillen-Perez, A., & Cano, M. (2021, 2). Intelligent IoT systems for traffic management: A practical application. *IET Intelligent Transport Systems*, 15(2), 273-285.
- Han, L., Zhang, L., & Guo, W. (2024). Multi-Agent Deep Reinforcement Learning for Multi-Lane Freeways Differential Variable Speed Limit Control in Mixed Traffic Environment. *Transportation Research Record*.
- Han, Y., Hegyi, A., Zhang, L., He, Z., Chung, E., & Liu, P. (2022, 11). A new reinforcement learning-based variable speed limit control approach to improve traffic efficiency against freeway jam waves. *Transportation Research Part C: Emerging Technologies*, 144.
- Hurtado-Gómez, J., Romo, J., Salazar-Cabrera, R., de la Cruz, Á., & Molina, J. (2021, 10). Traffic signal control system based on intelligent transportation system and reinforcement learning. *Electronics (Switzerland)*, 10(19).
- Isaenko, N., Colombaroni, C., Fusco, G., & Lahijanlian, Z. (2024, 4). Statistical and Clustering-Based Assessment of Variable Speed Limits Effects on Motorway Performance from Real-World Observations. *Future Transportation*, 4(2), 409-428.

- Jaleel, A., Hassan, M., Mahmood, T., Ghani, M., & Ur Rehman, A. (2020). Reducing congestion in an intelligent traffic system with collaborative and adaptive signaling on the edge. *IEEE Access*, 8, 205396-205410.
- Jiang, S., Huang, Y., Jafari, M., & Jalayer, M. (2021). A Distributed Multi-Agent Reinforcement Learning With Graph Decomposition Approach for Large-Scale Adaptive Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*.
- Kalambay, P., & Pulugurtha, S. (2024). Data-driven exploration of traffic speed patterns to identify potential road links for variable speed limit sign implementation. *Urban, Planning and Transport Research*, 12(1).
- Ke, Z., Li, Z., Cao, Z., & Liu, P. (2021, 7). Enhancing Transferability of Deep Reinforcement Learning-Based Variable Speed Limit Control Using Transfer Learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(7), 4684-4695.
- Khondaker, B., & Kattan, L. (2015, 11). Variable speed limit: An overview. *Transportation Letters*, 7(5), 264-278.
- Kim, Y., Kang, K., Park, N., Park, J., & Oh, C. (2024). Reinforcement learning approach to develop variable speed limit strategy using vehicle data and simulations. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*.
- Kušić, K., Ivanjko, E., Gregurić, M., & Miletić, M. (2020, 7). An overview of reinforcement learning methods for variable speed limit control. *Applied Sciences (Switzerland)*, 10(14).
- Li, D., & Lasenby, J. (2024). Imagination-Augmented Reinforcement Learning Framework for Variable Speed Limit Control. *IEEE Transactions on Intelligent Transportation Systems*, 25(2), 1384-1393.
- Li, Z., Yu, H., Zhang, G., Dong, S., & Xu, C. (2021, 4). Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 125.
- Lu, W., Yi, Z., Gu, Y., Rui, Y., & Ran, B. (2023, 8). TD3LVSL: A lane-level variable speed limit approach based on twin delayed deep deterministic policy gradient in a connected automated vehicle environment. *Transportation Research Part C: Emerging Technologies*, 153.

- Ouallane, A., Bahnasse, A., Bakali, A., & Talea, M. (2021). Overview of Road Traffic Management Solutions based on IoT and AI. *Procedia Computer Science*, 198, 518-523.
- Shaikh, A., Bhalekar, B., & Futane, P. (2024, 3). Dynamic Traffic Management Using Reinforcement Learning. *International Journal of Computing and Digital Systems*, 15(1), 1007-1017. Retrieved from <https://journals.uob.edu.bh/handle/123456789/5152>
- Siri, S., Pasquale, C., Sacone, S., & Ferrara, A. (2021, 8). Freeway traffic control: A survey. *Automatica*, 130.
- Sun, D., Jamshidnejad, A., & Schutter, B. (2024). A Novel Framework Combining MPC and Deep Reinforcement Learning With Application to Freeway Traffic Control. *IEEE Transactions on Intelligent Transportation Systems*.
- Vrbanić, F., Ivanjko, E., Kušić, K., & Čakija, D. (2021, 3). Variable speed limit and ramp metering for mixed traffic flows: A review and open questions. *Applied Sciences (Switzerland)*, 11(6).
- Vrbanić, F., Tišljarić, L., Majstorović, Ž., & Ivanjko, E. (2023, 4). Reinforcement Learning-Based Dynamic Zone Placement Variable Speed Limit Control for Mixed Traffic Flows Using Speed Transition Matrices for State Estimation. *Machines*, 11(4).
- Wang, C., Zhang, J., Xu, L., Li, L., & Ran, B. (2019). A New Solution for Freeway Congestion: Cooperative Speed Limit Control Using Distributed Reinforcement Learning. *IEEE Access*, 7, 41947-41957.
- Wang, H., Fu, Z., Lee, J., Matin, H., Alanqary, A., Urieli, D., . . . Monache, M. (2024, 2). Hierarchical Speed Planner for Automated Vehicles: A Framework for Lagrangian Variable Speed Limit in Mixed Autonomy Traffic. Retrieved from <http://arxiv.org/abs/2402.16993>
- Wenke, T., Eckert, D., Srovin, V., & Van Heirseele, R. (2022). *Variable Speed Limit Simulation in SUMO*.
- Wu, Y., Tan, H., Qin, L., & Ran, B. (2020, 8). Differential variable speed limits control for freeway recurrent bottlenecks via deep actor-critic algorithm. *Transportation Research Part C: Emerging Technologies*, 117.

- Yang, B., Zhang, H., Du, M., Wang, A., & Xiong, K. (2023, 2). Urban traffic congestion alleviation system based on millimeter wave radar and improved probabilistic neural network. *IET Radar, Sonar and Navigation*.
- Yang, J., Wang, P., Golpayegani, F., & Wang, S. (2024, 5). DVS-RG: Differential Variable Speed Limits Control using Deep Reinforcement Learning with Graph State Representation. Retrieved from <http://arxiv.org/abs/2405.09163>
- Zamir, S. W., Arora, A., Gupta, A., Khan, S., Sun, G., Shahbaz Khan, F., . . . Bai, X. (2019). iSAID: A large-scale dataset for instance segmentation in aerial images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 28–37). IEEE.
- Zhang, C., Chung, E., Sabar, N., Bhaskar, A., & Ma, Y. (2023). Optimisation of variable speed limits at the freeway lane drop bottleneck. *Transportmetrica A: Transport Science*, 19(2).
- Zhang, Y., Zhang, Z., Quiñones-Grueiro, M., Barbour, W., Weston, C., Biswas, G., & Work, D. (2024, 7). Field Deployment of Multi-Agent Reinforcement Learning Based Variable Speed Limit Controllers. Retrieved from <http://arxiv.org/abs/2407.08021>
- Zheng, S., Li, M., Ke, Z., & Li, Z. (2023). Coordinated Variable Speed Limit Control for Consecutive Bottlenecks on Freeways Using Multiagent Reinforcement Learning. *Journal of Advanced Transportation*, 2023.

المخلص

تقدم هذه الدراسة نظامًا مستقلًا لتخطيط المسارات باستخدام الذكاء الاصطناعي، يراقب البيئات صغيرة الحجم، مثل المستودعات ومتاجر التجزئة ومواقف السيارات، لإنشاء مسارات للروبوتات الأرضية. يُمكن النظام الروبوتات من التنقل وتجنب الاصطدامات من خلال توفير مسارات مثالية دون تحكم مباشر. يستخدم النظام نماذج التجزئة الدلالية - الشبكة التلافيفية الكاملة (FCN)، وU-Net، وSegNet، وPSPNet - لتحويل صور المنظر العلوي إلى شبكات ثنائية تُميز المساحات الفارغة عن المشغولة. تُعالج هذه الشبكات بواسطة خوارزميات تخطيط المسارات - A*، وDijkstra's، وBidirectional - لحساب مسارات فعّالة. يُظهر النظام، المُطبّق باستخدام بايثون باستخدام PyTorch، أداءً قويًا عبر 800 صورة تدريب و200 صورة اختبار من مجموعة بيانات iSAID. تُظهر النتائج تفوق U-Net باستمرار على النماذج الأخرى، محققًا أقل خسارة في التجزئة (0.0699) وأعلى الدرجات في مقاييس متعددة، بما في ذلك دقة البكسل (0.970)، ومؤشر Jaccard (0.948)، وRecall (0.993). برزت SegNet كمنافس قوي، بينما أظهرت PSPNet قيودًا ملحوظة. كان أداء تخطيط المسارات أكثر كفاءة على الشبكات المؤدّة من U-Net وSegNet، حيث حقق A* أسرع أوقات تنفيذ (0.15-0.18 ملي ثانية) وأقل استخدام للذاكرة (0.024-0.025 ميجابايت)، بينما حقق ثنائي الاتجاه أقصر المسارات (7.05-7.57 وحدة). في المقابل، أدت الشبكات المؤدّة من PSPNet إلى مسارات أطول بكثير (20.8-21.8 وحدة) ومتطلبات حوسبة أعلى. ويعمل النظام على تعزيز كفاءة الملاحة والسلامة والفعالية من حيث التكلفة لتطبيقات الخدمات اللوجستية في البيئات الصغيرة الحجم، حيث تم تحديد U-Net وA* باعتبارهما التركيبة المثالية للأداء في الوقت الفعلي.

جامعة البحرين

قسم الهندسة الكهربائية
والإلكترونية



كلية الهندسة

تخطيط المسار باستخدام الذكاء الاصطناعي

أطروحة مقدمة كجزء من متطلبات الحصول على درجة ماجستير العلوم في أنظمة الذكاء الاصطناعي

اعداد

احمد محمد ماجد الخالدي

الرقم الجامعي: 20180414

اشراف

د. أيمن الخزرجي

جامعة البحرين

مملكة البحرين

نوفمبر 2025م