REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

LEARNING-BASED SCENE EXTRAPOLATION IN ROBOTICS

Selma GÜZEL

DOCTOR OF PHILOSOPHY THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Asst. Prof. Erkan USLU

May, 2026

REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

LEARNING-BASED SCENE EXTRAPOLATION IN ROBOTICS

A thesis submitted by Selma GÜZEL in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY** is approved by the committee on 21.05.2026 in Department of Computer Engineering, Program of Computer Engineering.

Asst. Prof. Erkan USLU
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Asst. Prof. Erkan USLU, Supervisor
Yildiz Technical University _____

Prof. Dr. M. Fatih AMASYALI, Member
Yildiz Technical University _____

Prof. Dr. Songül VARLI, Member
Yildiz Technical University _____

Prof. Dr. Hatice KÖSE, Member
Istanbul Technical University _____

Prof. Dr. Sırma YAVUZ, Member
Atlas University _____

I hereby declare that the study entitled “Learning-based Scene Extrapolation in Robotics” that I prepared under the supervision of Asst. Prof. Erkan USLU was conducted in full compliance with the relevant legal requirements regarding data collection and data use; that all information obtained from other sources has been fully and properly acknowledged in the main text and references; that no fabrication and/or falsification has been made with respect to the research data or findings; and that throughout the study I have adhered to the principles of scientific research and academic ethics.

In the course of this thesis study, Generative Artificial Intelligence (GenAI) tools—specifically ChatGPT (GPT-5, OpenAI), Claude (Anthropic), Gemini (Google) were utilized under the author’s supervision for purposes such as language editing, academic writing support, code development, data visualization, and literature assistance.

I declare that I am aware of the risk that GenAI tools may generate inaccurate or fabricated information (hallucinations), and that I have carefully reviewed and verified all outputs obtained from these tools in terms of accuracy, integrity, reliability, and compliance with ethical standards. The use of GenAI tools has not, at any stage, replaced the author’s intellectual contribution, scientific judgment, or original idea generation. Full responsibility for the accuracy, currency, and scientific validity of all information, data, analyses, claims, and references presented in this thesis rests entirely with me. I accept full administrative and legal responsibility for any consequences that may arise if this declaration is determined to be false.

Selma GÜZEL

Signature

To my dear parents, for their unwavering support and prayers

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor Asst. Prof. Erkan USLU, who generously supported me throughout the thesis process and played a significant role in the development of this work. I am also deeply thankful to my former thesis advisor Prof. Dr. Sırma YAVUZ, who guided me in determining the topic of this thesis and contributed to its development with her valuable insights. I would like to thank the thesis committee members Prof. Dr. Hatice KÖSE and Prof. Dr. M. Fatih AMASYALI for their evaluations and constructive feedback that contributed to the improvement of this study. Lastly, I express my heartfelt thanks to my dear family, who have always provided me with both moral and financial support during this journey.

Selma GÜZEL

TABLE OF CONTENTS

LIST OF SYMBOLS xii

LIST OF ABBREVIATIONS xiv

LIST OF FIGURES xvi

LIST OF TABLES xxi

ABSTRACT xxiii

ÖZET xxv

1 INTRODUCTION 1

1.1 Motivation 1
1.2 Overview 2
1.3 Problem Definition and Hypothesis 7
1.4 Research Motivation and Background 9
1.5 Research Objectives 10
1.6 Contributions 11
1.7 Scope and Limitations 12
1.8 Thesis Organisation 13

2 LITERATURE REVIEW 15

2.1 MCN-SLAM: Map Completion with Deep Generative Models for SLAM [19] 18
2.2 Occupancy Anticipation for Efficient Exploration and Navigation [22] 21
2.3 Uncertainty-Driven Planning for Exploration and Navigation [18] . 24
2.4 Cross-Modal Map Learning for Vision and Language Navigation [27] 26
2.5 Occupancy Map Prediction using Generative and Fully Convolutional Networks for Vehicle Navigation [14] 29
2.6 Uncertainty-Aware Occupancy Map Prediction Using Generative Networks [13] 31
2.7 Learned Map Prediction for Enhanced Mobile Robot Exploration [15] 34

2.8 Im2Pano3D: Extrapolating 360° Structure and Semantics Beyond the Field of View [28] . . . 37
2.9 Deep 3D Semantic Scene Extrapolation [29] . . . 40
2.10 Extreme Relative Pose Estimation for RGB-D Scans via Scene Completion [26] . . . 43
2.11 Emergence of Exploratory Look-Around Behaviors through Active Observation Completion [25] . . . 46
2.12 Map-Predictive Motion Planning in Unknown Environments [16] . . . 49
2.13 Unifying Map and Landmark Based Representations for Visual Navigation [30] . . . 51
2.14 SSCNav: Confidence-Aware Semantic Scene Completion for Visual Semantic Navigation [23] . . . 54
2.15 Seeing the Un-Scene: Learning Amodal Semantic Maps for Room Navigation [24] . . . 57
2.16 SenseExpo: Efficient Autonomous Exploration with Prediction Information from Lightweight Neural Networks [11] . . . 60
2.17 MapEx: Indoor Structure Exploration with Probabilistic Information Gain from Global Map Predictions [12] . . . 63
2.18 Occupancy Grid Artefact Removal and Error Correction using GANs [20] . . . 67
2.19 Map Prediction and Generative Entropy for Multi-Agent Map Prediction [17] . . . 70
2.20 Geometry-Based Completion via Structural Inference [21] . . . 73

3 MAP COMPLETION **81**
3.1 Mathematical Problem Formulation . . . 82
3.2 Evaluation Protocol and the MASKED Metric . . . 83
3.3 Rule-Based Geometric Baseline . . . 84
3.3.1 Algorithm 1: Nearest-Structure Completion . . . 85
3.3.2 Algorithm 2: Bounding-Box Fallback Completion . . . 86
3.3.3 Limitations of the Geometric Baseline . . . 87
3.4 Models for Upper-to-Lower Local Completion . . . 88
3.4.1 ImageGPT . . . 88
3.4.2 Conditional Variational Autoencoder (CVAE) . . . 90
3.5 Models for Global Map Completion . . . 92
3.5.1 CVAE (Global, 128×128) . . . 92
3.5.2 Conditional GAN (CGAN / Pix2Pix) . . . 94
3.5.3 RePaint-Style Diffusion (DDPM, Global) . . . 95
3.5.4 Vision Transformer Encoder–Decoder (ViT) . . . 96

3.5.5 Residual U-Net (Global, 3+3 Stages) . . . 97
3.5.6 GEOM: Rule-Based Baseline at Global Scale . . . 98
3.6 Models for Frontier-Centered Local Completion . . . 98
3.6.1 Residual U-Net Variants . . . 99
3.6.2 LaMa-Style Inpainting Model (LamaUNet) . . . 101
3.6.3 ImageGPT-Style Autoregressive Baseline ($\text{ImageGPT}_{\text{local}}$) . . . 102
3.6.4 Row-Wise LSTM Baseline (LSTM_{row}) . . . 103
3.6.5 Diffusion-Based RePaint Model ($\text{Diffusion}_{\text{local}}$) . . . 104
3.6.6 GAN Variants (Pix2Pix-Hinge and Seg-GAN) . . . 105
3.6.7 Conditional Variational Autoencoder (CVAE) . . . 106
3.6.8 Vision Transformer Encoder–Decoder ($\text{ViT}_{\text{frontier}}$) . . . 107
3.7 Chapter Summary . . . 108

4 MAP COMPLETION IN FRONTIER-BASED EXPLORATION 110
4.1 Overview . . . 110
4.2 System Architecture and Design Evolution . . . 111
4.2.1 Earlier Design: Separate Publisher–Subscriber Node . . . 111
4.2.2 Current Design: Extended C++ Node with Batch Service . . . 112
4.3 Full Pipeline Architecture . . . 112
4.4 Frontier Extraction and Representation . . . 113
4.4.1 BFS-Based Detection . . . 113
4.4.2 Frontier Point Representation: Centroid and Safest Point . . . 114
4.5 Frontier Clustering . . . 114
4.6 Local Map Completion Integration . . . 115
4.6.1 Operational Modes . . . 115
4.6.2 Batch Service Architecture and Patch Caching . . . 116
4.6.3 Gap Closing . . . 116
4.6.4 Completion Score Computation . . . 116
4.7 Frontier Filtering . . . 117
4.8 Frontier Scoring Mechanism . . . 118
4.9 Target Selection and Execution . . . 119
4.9.1 Goal Dispatch . . . 119
4.9.2 Progress Monitoring and Blacklisting . . . 119
4.9.3 Navigation Execution . . . 120
4.10 Configuration Parameters . . . 120
4.11 Algorithm . . . 121
4.12 Summary . . . 121

5 GT ORACLE NAVIGATION STUDY 123
5.1 Motivation . . . 123

5.2 Oracle Mode Design 124
5.2.1 Ground-Truth Map Loading and Alignment 124
5.2.2 Wall Closing in the Oracle Setting 124
5.2.3 Frontier Filtering in the Oracle Setting 125
5.3 Summary 125

6 DATASETS 126
6.1 Source Dataset: HouseExpo 127
6.2 Static Map Conversion for Generative Training 128
6.2.1 Motivation 128
6.2.2 Pipeline Overview 128
6.2.3 Occupancy Map Generation and Pixel Classification 129
6.2.4 Algorithm: Static Map Conversion 130
6.2.5 Applications and Benefits 130
6.2.6 Static Dataset Output 132
6.3 Gazebo-Based Simulation Pipeline 132
6.3.1 Motivation 132
6.3.2 Pipeline Overview 133
6.3.3 Custom Voronoi-Based Map Saving Node 134
6.3.4 Input-Output Map Pairs for Global Completion 135
6.3.5 Post-Processing: Frontier-Based Patch Extraction 136
6.3.6 Algorithm: Full Simulation-Based Dataset Generation . . . 137
6.3.7 Gazebo-Based Dataset Output 138
6.4 Dataset Splits and Quality Control 139
6.4.1 Static Dataset Split 139
6.4.2 Global Map Completion Dataset Split 139
6.4.3 Frontier Patch Dataset Split 140
6.5 Comprehensive Dataset Comparison 140
6.6 Summary 141

7 EXPERIMENTS 143
7.1 Rule-Based Baseline: Geometric Map Completion 144
7.1.1 Experimental Setup 144
7.1.2 Quantitative Results 145
7.1.3 Qualitative Results 145
7.2 Upper-to-Lower Directional Local Completion 150
7.2.1 Experimental Setup 150
7.2.2 Quantitative Results 150
7.2.3 Qualitative Results 151
7.3 Global Map Completion 151

7.3.1 Experimental Setup . . . 153
7.3.2 Quantitative Results . . . 153
7.3.3 Qualitative Results . . . 154
7.4 Frontier-Centered Patch-Based Local Map Completion . . . 155
7.4.1 Experimental Setup . . . 155
7.4.2 Quantitative Results . . . 156
7.4.3 Qualitative Results . . . 160
7.5 Final Model Selection for Online Exploration . . . 164
7.6 GT Oracle Navigation Study: Supplementary Single-Environment Analysis . . . 165
7.6.1 Motivation and Oracle Framework . . . 165
7.6.2 Scenarios . . . 166
7.6.3 Experimental Setup . . . 166
7.6.4 Quantitative Results . . . 167
7.6.5 Qualitative Results . . . 168
7.6.6 Why This Approach Was Not Adopted for the Exploration Pipeline . . . 168
7.7 System-Level Exploration Evaluation: Setup . . . 170
7.7.1 Experimental Setup . . . 170
7.7.2 Effect of Clustering on Exploration Efficiency . . . 174
7.8 System-Level Exploration Evaluation: Results . . . 175
7.8.1 Quantitative Results . . . 175
7.8.2 Qualitative Results . . . 184
7.9 Chapter Summary and Key Findings . . . 184
7.9.1 Model Selection Hierarchy . . . 184
7.9.2 System-Level Performance Summary . . . 186
7.9.3 Structural and Methodological Insights . . . 187

8 DISCUSSION 189
8.1 Completion Quality and Exploration Efficiency . . . 189
8.2 Convolutional Priors and the Geometry-Dominant Nature of Local Completion . . . 190
8.3 The Role of Spatial Scale . . . 191
8.4 Isolated Metrics vs. System-Level Benefit . . . 192
8.4.1 FULL vs. MASKED Evaluation . . . 192
8.4.2 Oracle Observation and Achieved Gain . . . 192
8.5 Limitations . . . 193
8.6 Relation to Prior Work . . . 195
8.7 Summary . . . 196

9 FUTURE WORK **198**
9.1 Model and Integration Improvements . . . 198
9.1.1 Larger and Adaptive Patch Sizes . . . 198
9.1.2 Multi-Scale Completion Architectures . . . 199
9.1.3 Tighter Completion–Navigation Coupling . . . 199
9.1.4 Robust Stall and Oscillation Detection . . . 201
9.1.5 Online Model Adaptation . . . 201
9.1.6 Loss Function Refinement . . . 201
9.1.7 Uncertainty-Aware Frontier Scoring . . . 201
9.1.8 Richer Frontier Scoring Terms . . . 202
9.2 Benchmarking Against Related Systems . . . 202
9.2.1 Direct Comparison with Information-Gain-Based Frontier Scoring . . . 202
9.3 Exploration Pipeline Extensions . . . 203
9.3.1 Late-Stage Exploration Strategy . . . 203
9.3.2 Geometric Filtering Heuristics . . . 204
9.3.3 Map Completion–Aware Global Path Planning . . . 204
9.3.4 Synchronisation-Aware Closed Live Map . . . 204
9.4 Generalisation and Deployment . . . 205
9.4.1 Physical Robot Validation . . . 205
9.4.2 Online Evaluation of Map Completion Quality . . . 205
9.4.3 Multi-Environment Generalisation . . . 206
9.4.4 Sensor Modality Extension . . . 206
9.4.5 Dynamic Environments . . . 206
9.4.6 PseudoSLAM–ROS Integration for Accelerated Data Collection . . . 206
9.4.7 Coverage-Based Dataset Construction . . . 207
9.5 Longer-Term Research Directions . . . 207
9.5.1 Semantic and Task-Aware Completion . . . 207
9.5.2 Multi-Robot and Collaborative Completion . . . 208
9.6 Summary . . . 208

10 CONCLUSION **209**
10.1 Summary of Contributions . . . 209
10.1.1 Contribution 1: Dataset and Preprocessing Pipeline . . . 209
10.1.2 Contribution 2: Comprehensive Model Evaluation . . . 209
10.1.3 Contribution 3: Frontier Scoring with Predicted Information Gain . . . 210
10.1.4 Contribution 4: System-Level Integration and Evaluation . . . 211

10.2 Key Findings 211
10.3 Closing Remarks 212

REFERENCES **214**

PUBLICATIONS FROM THE THESIS **219**

LIST OF SYMBOLS

w_k	Completion reward weight (`completion_known_scale`)
$\boldsymbol{\mu}$	CVAE latent mean vector
$\boldsymbol{\sigma}^2$	CVAE latent variance vector
z	CVAE latent variable
w_d	Distance weight in frontier cost (`potential_scale`)
d	Euclidean distance from robot to frontier (metres)
t	Exploration time step index
c_{wall}	Fixed wall-prediction penalty in frontier cost (= 100)
δ_c	Frontier clustering radius threshold (metres)
s	Frontier size (metres)
w_s	Frontier size weight in frontier cost (`gain_scale`)
β	KL divergence weight in CVAE training
$\mathcal{G}(\cdot)$	Map completion model
M_{miss}	Missing (unknown) cells of the occupancy map
θ	Model parameters
n_{cur}	Number of free pixels in the raw input patch
n_{pred}	Number of free pixels in the completed patch
M_{obs}	Observed cells of the occupancy map
M	Occupancy grid map
H, W	Occupancy map height and width (cells)
r	Occupancy map resolution (metres/cell)
P	Patch side length in cells (default: 32)
$\hat{M}$	Predicted (completed) occupancy map

r_f	Predicted free-space gain score
α	Reconstruction loss weight in CVAE training
$\mathbf{s}$	Safest point used as navigation goal
$\mathcal{F}$	Set of frontier cell world coordinates
$\mathbf{c}$	Snapped frontier centroid used for patch extraction
T	Total diffusion timesteps

LIST OF ABBREVIATIONS

BASE	Classical frontier-based exploration baseline (no map completion)
BDS	Large (Big) Dataset Setting
BFS	Breadth-First Search
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CVAE	Conditional Variational Autoencoder
DDIM	Denoising Diffusion Implicit Models
DDPM	Denoising Diffusion Probabilistic Models
DWA	Dynamic Window Approach
FFC	Fast Fourier Convolution
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
GEOM	Rule-based Geometric Completion Baseline
GT	Ground Truth
IoU	Intersection over Union
KLD	Kullback–Leibler Divergence
LaMa	Large Mask inpainting model
LSTM	Long Short-Term Memory
MC	Map Completion (exploration configuration)
MSE	Mean Squared Error
PGM	Portable GrayMap
PNG	Portable Network Graphics
ReLU	Rectified Linear Unit

RMSE	Root Mean Squared Error
ROS	Robot Operating System
SDF	Simulation Description Format
SDS	Small Dataset Setting
SLAM	Simultaneous Localisation and Mapping
SSIM	Structural Similarity Index Measure
STL	Standard Tessellation Language (3D mesh format)
TEB	Timed Elastic Band (local planner)
UQI	Universal Quality Index
VAE	Variational Autoencoder
ViT	Vision Transformer

LIST OF FIGURES

Figure 1.1 Offline global map completion through scene extrapolation. A partially observed occupancy map is extended by predicting unknown regions based on the observed spatial structure. 3

Figure 1.2 Research workflow of this thesis. The main flow (solid lines) proceeds from dataset construction through map completion methods, model training and selection, to system integration and exploration evaluation. The secondary path (dashed lines) represents the GT Oracle navigation study, conducted as an exploratory by-product. 3

Figure 1.3 System architecture of the completion-aware exploration pipeline. The map completion module is integrated between frontier detection and goal selection, enabling frontier-based local map completion during exploration. The SLAM module provides the occupancy map, while the navigation stack (move_base) generates motion commands. 7

Figure 1.4 Online illustration of frontier-based local map completion during exploration. A local patch around each candidate frontier is extracted and completed, and the predicted free-space gain is used to influence frontier selection. 8

Figure 1.5 Illustrative example of map completion guiding robot navigation toward a more informative region. The predicted obstacle in the unknown region redirects the robot from its default path toward a genuinely free area." 9

Figure 3.1 Step-by-step execution of the geometric door discovery and map completion pipeline (Algorithm 1). See text for pipeline details. 86

Figure 3.2 Overview of the geometric completion pipeline showing both algorithms. Starting from the input map, the pipeline proceeds through re-rotation, bounding box detection (Canny), door detection, and guide line extraction (Hough transform). Algorithm 1 fills regions using the nearest wall or guide line; Algorithm 2 falls back to the bounding box when no reference is found. 87
Figure 3.3 Architecture of the ImageGPT model adapted for upper-to-lower map completion. 89
Figure 3.4 Architecture of the decoder-only CVAE for upper-to-lower completion. 91
Figure 3.5 Global CVAE (128×128) with pyramid conditioning. 93
Figure 3.6 Global CGAN (Pix2Pix-hinge) for 128×128 map completion. . 94
Figure 3.7 Conditioned RePaint diffusion model for global map completion (128×128). 95
Figure 3.8 ViT encoder–decoder for global map completion (128×128). . 96
Figure 3.9 Residual U-Net (3+3 stages) for global 128×128 map completion. 97
Figure 3.10 Residual U-Net ($\text{UNet}_{\text{base}}$, base$= 48$) for frontier-centered local completion. 99
Figure 3.11 Attention U-Net (UNet_{att}) for frontier-centered local completion. 100
Figure 3.12 LaMa-style U-Net (LamaUNet, base$= 64$) for frontier-centered local completion. 102
Figure 3.13 ImageGPT-style causal transformer ($\text{ImageGPT}_{\text{local}}$) for frontier-centered completion. 102
Figure 3.14 LSTM cell with forget, input, and output gates. 103
Figure 3.15 Row-wise bidirectional LSTM (LSTM_{row}) for frontier-centered completion. 103
Figure 3.16 Conditioned RePaint diffusion model for frontier-centered completion (32×32). 104
Figure 3.17 Pix2Pix-Hinge vs. Seg-GAN for frontier-centered completion. . 106
Figure 3.18 CVAE for frontier-centered completion (32×32) with pyramid conditioning. 107
Figure 3.19 ViT encoder–decoder ($\text{ViT}_{\text{frontier}}$) for frontier-centered local completion (32×32). 107
Figure 6.1 Example using fixed values for PGM dimensions and origin coordinates in the static map conversion pipeline (Algorithm 4). 131
Figure 6.2 Example using STL-based origin coordinates and dynamically computed PGM dimensions in the static map conversion pipeline (Algorithm 4). 131

Figure 6.3 Voronoi graph structure and crossing points used for map saving triggers. 135
Figure 6.4 Example of an input-output map pair generated from sequential exploration data, where the input corresponds to time t and the ground-truth output corresponds to time $t + 3$. 135
Figure 6.5 Representative occupancy maps from Dataset 1, collected via Gazebo-based simulation across diverse HouseExpo floor plans. Upper row: partial maps at time t. Lower row: corresponding more complete maps at time $t + 3$. 136
Figure 6.6 Extracted 32×32 input (top) and output (bottom) patches centred at frontiers. Note that the bounding boxes around the patches are for visualization purposes and do not represent wall structures. The leftmost column corresponds to the map in Figure 6.4. 137
Figure 6.7 Frontier-centered patch extraction from sequential SLAM maps. A 32×32 patch centred on a selected frontier is extracted from both the input map at time t and the output map at time $t + 3$, forming a training pair. Yellow circles: frontier centroids; red circle: selected frontier; blue rectangle: 32×32 patch region. 138
Figure 7.1 Comparison on a partially observed environment: (a) detected door locations, (b) ground truth region of interest, (c) completion produced by Algorithm 1, and (d) completion produced by Algorithm 2. 146
Figure 7.2 Comparison on a scenario with limited empty space inside the bounding box: Algorithm 2 provides a more complete reconstruction than Algorithm 1. 147
Figure 7.3 Impact of threshold selection on completion performance. A higher threshold (125) detects the door more precisely and enables correct room completion, while a lower threshold (100) misses the door in this case and leaves the room undetected. . . 148
Figure 7.4 Limitation: non-axis-aligned walls cannot be handled by either algorithm. The guide-line extraction fails to follow the diagonal geometry, resulting in incorrect or absent completions. 149
Figure 7.5 Successful upper-to-lower completion results from ImageGPT across four orientation angles. 151
Figure 7.6 Successful upper-to-lower completion results from CVAE across four orientation angles. 152

Figure 7.7 Failure cases from ImageGPT (left) and CVAE (right). First columns: lack of guiding structure in input. Second columns: shape inconsistencies or unrealistic continuations. 152
Figure 7.8 Qualitative global completion examples. Columns left to right: input, CVAE, CGAN, RePaint, ViT, UNet, GEOM, target. . . . 154
Figure 7.9 Qualitative comparison of primary model families on the SDS test set. Columns left to right: input condition (cond), ViT, UNet, RePaint, Pix2Pix, GEOM, CVAE, blended result, and ground-truth target. 161
Figure 7.10 Qualitative results for GAN-based and sequential model variants. Columns: input, Seg-GAN, LSTM, LaMa, ImageGPT, blended, and target. 162
Figure 7.11 Qualitative comparison across U-Net architecture variants on the SDS test set. Columns left to right: input condition (cond), $\text{UNet}_{\text{base}}$, UNet_{new}, UNet_{att}, UNet_{4L}, blended result, and ground-truth target. All configurations produce structurally consistent maps, with marginal differences in obstacle recall. . . 163
Figure 7.12 The HouseExpo environment used in the GT oracle navigation study. The map contains narrow doorways, parallel corridors with spurious wall gaps, and exterior spurious frontiers from SLAM noise. 167
Figure 7.13 Parallel corridors problem. (a–b) Baseline zigzag attempts through wall gaps; (c) eventual resolution. (d) LMC+Filt partial fix; (e) LMC+Filt with alignment shift; (f) Full system (LMC+GWC+Filt) resolved. The pink rectangles show the patches around frontiers. If the completion is performed they are filled with the corresponding patch from GT map. Green spheres are frontiers and the orange sphere is the target frontier. 169
Figure 7.14 Exterior frontier problem. (a) SLAM noise creates spurious target beyond wall which causes the robot to attempt to reach an unreachable region; (b) Filtering eliminates the invalid frontiers. 169
Figure 7.15 Ground-truth occupancy maps of the eight evaluation environments. Black = wall; white = free space; grey = unknown. Environments ordered by difficulty tier (Easy, Medium, Hard). 171
Figure 7.16 Average elapsed time to reach each coverage milestone across eight environments (error bars: ± 1 SD; Wilcoxon signed-rank significance markers, $n = 24$). MC (blue) is consistently faster than BASE (red) at all milestones except Max. 176

Figure 7.17 MC time advantage (%) relative to BASE for each environment–milestone combination. Green = MC faster; red = BASE faster. Each cell: $(t_{\mathrm{BASE}} - t_{\mathrm{MC}})/t_{\mathrm{BASE}} \times 100$. The 95% column is the only one where all eight environments are green; `4bb74477` is the only predominantly red row. 177
Figure 7.18 Statistical significance map (paired t-test, $n = 3$ per cell). Dark green: $p < 0.001$ (MC); mid green: $p < 0.01$; light green: $p < 0.05$; grey: n.s.; pink/red: BL significant. 177
Figure 7.19 Elapsed time to reach each coverage milestone per environment. Blue solid: MC; red dashed: BASE. Shaded area marks where one method leads. Corner label "MC k/7" counts MC milestone wins. 178
Figure 7.20 Total distance travelled to reach each milestone per environment. Layout identical to Figure 7.19. 178
Figure 7.21 Scatter plot: average MC time advantage over BASE (y-axis) per environment, grouped by difficulty tier (x-axis). Colours: green = Easy, orange = Medium, red = Hard. Points within the same tier are separated by a small horizontal jitter for readability. `4bb74477` is the only point below the zero line. . . 183
Figure 7.22 Structural profile radar charts for all eight environments (normalised per axis). Each panel reports the MC milestone win count. The `4bb74477` panel (MC 1/7) shows an unremarkable polygon, confirming that the outlier behaviour is not captured by aggregate structural metrics. 184
Figure 7.23 Completion-guided frontier evaluation examples. Left column: patch selection; right column: prediction. Green: selected targets; orange: centroids; cyan: current goal. 185

LIST OF TABLES

Table 2.1 Comprehensive categorization of reviewed studies by functional roles 16
Table 2.2 Map prediction models in reviewed studies 16
Table 2.3 Exploration methods in reviewed studies 17
Table 2.4 Datasets used in reviewed studies: properties 18
Table 2.5 Datasets used in reviewed studies: paper–dataset mapping 19
Table 2.6 Evaluation protocols in reviewed studies 77
Table 2.7 Comparison of exploration systems for integration with patch-based map prediction 80
Table 4.1 Key configuration parameters and their roles 120
Table 6.1 Comparison of static, full-map, and patch-based datasets 141
Table 6.2 Dataset statistics for all pipelines 141
Table 7.1 IoU results for the rule-based geometric completion algorithms across 7,302 HouseExpo test maps 145
Table 7.2 Upper-to-lower completion results for ImageGPT and CVAE across four orientation angles ($0°, 30°, 60°, 90°$) and their average (best average values in bold) 150
Table 7.3 Frontier-centered local completion on SDS, MASKED evaluation, where † = excluded (throughput < 20 img/s) 157
Table 7.4 Frontier-centered local completion on SDS, FULL evaluation with throughput 158
Table 7.5 Frontier-centered local completion on BDS, MASKED evaluation 158
Table 7.6 Frontier-centered local completion on BDS, FULL evaluation . . 159
Table 7.7 Effect of patch size on UNet performance (MASKED evaluation) 159
Table 7.8 UNet variant comparison on SDS, MASKED evaluation (best values in bold) 164
Table 7.9 Navigation metrics comparison across five scenarios and four SLAM runs (best results per run in bold) 168
Table 7.10 Average performance improvement over baseline for each scenario 168
Table 7.11 Structural properties of the eight evaluation environments; difficulty tiers assigned by a four-rule heuristic 171
Table 7.12 Effect of frontier clustering on BASE exploration efficiency . . . 175

Table 7.13 Cross-environment average elapsed time (seconds) to reach each coverage milestone; Δ = MC$-$BASE (negative = MC faster) . . . 175
Table 7.14 3-run average elapsed time (seconds) for `4bb74477` (best value per milestone in bold) 181

ABSTRACT

Learning-based Scene Extrapolation in Robotics

Selma GÜZEL

Department of Computer Engineering
Doctor of Philosophy Thesis

Supervisor: Asst. Prof. Erkan USLU

In autonomous exploration, robots incrementally construct occupancy grid maps from sensor observations to represent partially observed environments. Due to limited field of view, occlusions, and sensor noise, such maps often include extensive unknown regions that negatively affect frontier selection and decision-making, and prevent robots from reasoning about the structure that lies beyond the current sensor boundary. This thesis studies **map prediction (scene extrapolation)** as a learning problem where missing regions of partial occupancy maps are inferred from observed structure and analyzes how these predictions influence exploration behavior and efficiency within a ROS-based navigation pipeline.

Four map completion formulations are studied: a training-free **rule-based geometric baseline**; **directional local completion** using a **CVAE** and an **ImageGPT-style** model; **global map completion** (128×128) comparing six model families; and **frontier-centered local completion** on 32×32 patches with irregular unknown regions. The best-performing model is deployed in the live exploration pipeline and evaluated at the system level.

A custom dataset is derived from HouseExpo environments via a Gazebo–ROS pipeline, with environment-level splits to prevent data leakage. Model families are compared using a masked-region evaluation protocol—metrics over originally unknown cells only—supported by IoU, $\mathrm{F1}$, SSIM, and throughput.

Results indicate that **frontier-centered completion is primarily governed by**

geometric structure best served by the residual U-Net family, achieving the highest accuracy over masked regions and real-time throughput. The selected U-Net augments the frontier cost function within an extended `explore_lite` framework with two prediction-driven terms: a free-space gain reward and a wall-prediction penalty. Completed patches are not injected into the navigation map, and wall-predicted frontiers are penalized rather than discarded, to limit the impact of erroneous predictions.

System-level evaluation across **eight HouseExpo simulation environments** shows that completion-aware frontier ranking outperforms the classical baseline in **45 of 56 milestone comparisons (80.4%)**, with an average speedup of **7.2%**. The **95% coverage milestone** serves as the primary evaluation boundary—beyond this point, remaining frontier cells correspond to geometrically constrained corner pockets better resolved by offline or human-assisted methods—where MC achieves a consistent and statistically significant **+10.4%** advantage across all eight environments. A supplementary single-environment oracle study recording a **+48%** navigation improvement should be interpreted as an exploratory by-product observation, as it is partly attributable to wall-gap path planning artifacts on a single map.

Overall, this thesis highlights that map completion must be assessed not only through visual similarity metrics but also by its downstream impact on robotic autonomy. The proposed framework offers a practical, modular, and ROS-compatible pathway to incorporate predictive scene extrapolation into frontier-based exploration of indoor environments.

Keywords: Scene Extrapolation, Frontier-based Exploration, Occupancy Grid Maps, U-Net, ROS.

ÖZET

Robotikte Öğrenme Tabanlı Sahne Ekstrapolasyonu

Selma GÜZEL

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Dr. Öğretim Üyesi Erkan USLU

Otonom keşif süreçlerinde robotlar, kısmen gözlemlenmiş ortamları temsil edebilmek için sensör verilerinden yararlanarak doluluk (occupancy) haritalarını kademeli olarak oluşturur. Sınırlı görüş alanı, örtülmeler (occlusion) ve sensör gürültüsü nedeniyle bu haritalar çoğu zaman geniş bilinmeyen bölgeler içerir; bu durum frontier seçimi ve karar verme süreçlerini olumsuz etkileyerek robotun mevcut sensör sınırının ötesindeki yapıyı çıkarım yapmasını zorlaştırır.

Bu tez, **harita tahmini (sahne ekstrapolasyonu)** problemini, kısmi doluluk haritalarındaki eksik bölgelerin gözlemlenen yapıya dayanarak çıkarıldığı bir öğrenme problemi olarak ele almakta ve bu tahminlerin ROS tabanlı bir navigasyon sistemi içerisinde keşif davranışı ve verimliliği üzerindeki etkisini analiz etmektedir.

Çalışmada dört harita tamamlama formülasyonu incelenmiştir: eğitim gerektirmeyen **kural tabanlı geometrik yöntem**; **CVAE** ve **ImageGPT benzeri** model kullanan **yönlü yerel tamamlama**; altı model ailesiyle karşılaştırılan **global harita tamamlama** (128×128); ve en kritik formülasyon olan, 32×32 frontier yamalarındaki düzensiz bilinmeyen bölgeleri tahmin eden **frontier-merkezli yerel tamamlama**. En iyi model canlı keşif pipeline'ına entegre edilerek sistem düzeyinde değerlendirilmiştir.

HouseExpo ortamlarından Gazebo–ROS tabanlı bir pipeline ile özel bir veri kümesi türetilmiş, veri sızıntısını önlemek için ortam bazlı ayrım uygulanmıştır. Model aileleri, yalnızca bilinmeyen hücreler üzerinden hesaplanan **maskeli bölge değerlendirme** protokolüyle karşılaştırılmış; IoU, F1, SSIM ve işlem hızı ölçütleri

kullanılmıştır.

Elde edilen sonuçlar, **frontier-merkezli tamamlamanın büyük ölçüde geometrik yapıya bağlı bir problem olduğunu ve artık (residual) U-Net mimarisinin bu yapı için en uygun çözümü sunduğunu** göstermektedir. Bu model, maskeli bölgelerde en yüksek doğruluğu ve gerçek zamanlı işlem hızını sağlamıştır. Seçilen U-Net, genişletilmiş `explore_lite` çerçevesinde frontier maliyet fonksiyonuna serbest alan kazanımı ödülü ve duvar tahmini cezası olmak üzere iki bileşen ekler. Hatalı tahminlerin etkisini sınırlamak için tamamlanan bölgeler navigasyon haritasına eklenmemiş, duvar tahminli frontier'lar ise silinmek yerine maliyetlendirilmiştir.

Sekiz HouseExpo simülasyon ortamı üzerinde yapılan sistem düzeyi değerlendirmeler, tamamlama destekli frontier sıralamasının temel yönteme göre **56 karşılaştırmanın 45'inde (%80,4)** daha iyi performans gösterdiğini ve ortalama **%7,2 hızlanma** sağladığını ortaya koymuştur. **%95 kapsama eşiği** ana değerlendirme sınırı olarak belirlenmiş; bu eşikten sonra kalan frontier'ların çoğunlukla geometrik olarak kısıtlı bölgelerden oluştuğu gözlemlenmiştir. Bu noktada yöntem, tüm ortamlarda tutarlı ve istatistiksel olarak anlamlı **+%10,4** avantaj sağlamıştır. Tek bir ortamda gerçekleştirilen oracle çalışmasında **+%48** navigasyon iyileştirmesi gözlemlenmiş olmakla birlikte, bu sonuç kısmen duvar boşluğu yol planlama eserlerinden kaynaklandığından açıklayıcı bir gözlem olarak değerlendirilmelidir.

Sonuç olarak bu tez, harita tamamlama yöntemlerinin yalnızca görsel benzerlik metrikleriyle değil, aynı zamanda robotik özerkliğe olan etkileri üzerinden değerlendirilmesi gerektiğini vurgulamaktadır. Önerilen çerçeve, iç mekânlarda frontier tabanlı keşif süreçlerine sahne ekstrapolasyonunun entegre edilmesi için modüler ve ROS uyumlu bir yaklaşım sunmaktadır.

Anahtar Kelimeler: Sahne Ekstrapolasyonu, Frontier Tabanlı Keşif, Doluluk Haritaları, U-Net, ROS.

1
INTRODUCTION

1.1 Motivation

The operational integrity of autonomous mobile systems is frequently compromised by the partial observability inherent in real-world environments. Factors such as the finite scanning range of sensors, physical occlusions, and stochastic SLAM noise typically result in occupancy grid representations characterized by substantial unobserved territories. Such informational voids not only hinder efficient target identification during exploration but also introduce instabilities in path planning, often leading to sub-optimal or physically infeasible trajectories near partially mapped boundaries. This thesis studies map completion as a predictive learning problem in which unobserved regions are inferred from available spatial evidence, allowing the system to estimate structures beyond its sensing boundary.

This computational challenge finds a compelling parallel in human spatial cognition, where individuals routinely infer unobserved environmental layouts by leveraging prior experience and structural context. Such biological intuition serves as the conceptual catalyst for scene extrapolation in robotics—a predictive paradigm aimed at generating plausible hypotheses regarding the spatial configuration beyond the current sensor horizon. By exploiting learned spatial priors, scene extrapolation extends reactive mapping and enables the system to form informed estimates about regions that have not yet been directly observed.

This is especially valuable in indoor environments where architectural regularities (corridors, rooms, junctions) provide strong statistical cues. While many prior works evaluate completion quality in isolation—and only a limited subset integrates predictions into active exploration pipelines—in robotics, however, the key issue is whether predicted maps improve downstream exploration behaviour rather than only their standalone quality.

This thesis therefore emphasises:

- building a realistic simulation-based data pipeline aligned with actual robot exploration trajectories,
- evaluating a broad range of generative model families under multiple completion formulations with appropriate evaluation protocols,
- and measuring downstream benefit through ROS-based exploration experiments, with a supplementary oracle observation to contextualise the potential of tighter navigation coupling.

1.2 Overview

Autonomous robots operating in unstructured or partially observable environments must build a reliable internal representation of their surroundings to make informed decisions. In practice, this representation is often an *occupancy grid map*—a 2D discretisation of space into *free*, *occupied* (obstacle), and *unknown* regions. Although occupancy grids provide a convenient interface between perception and planning, they are inherently limited by sensor range, occlusions, viewpoint constraints, and noise. As a result, robots must operate under significant spatial uncertainty, which complicates frontier selection and reduces the stability of navigation planning.

In this thesis, *scene extrapolation* is used in close relation to *map completion* to denote the prediction of unobserved environmental structure from partial occupancy information. While map completion is a broader term encompassing both inpainting and outpainting, the primary focus is on predicting unknown regions based on observed spatial information to extend the robot's belief beyond its immediate sensor range.

An illustrative example of offline global map completion is shown in Figure 1.1, where a partially observed occupancy map is extended by predicting the structure of unknown regions based on the observed layout.

The overall research workflow of this thesis is illustrated in Figure 1.2, showing the progression from dataset construction through map completion and system integration to the final exploration evaluation.

From Map Completion to System-Level Benefit

The need for scene extrapolation arises across many real-world indoor tasks involving partial observability (e.g., narrow corridors, cluttered rooms, multi-room

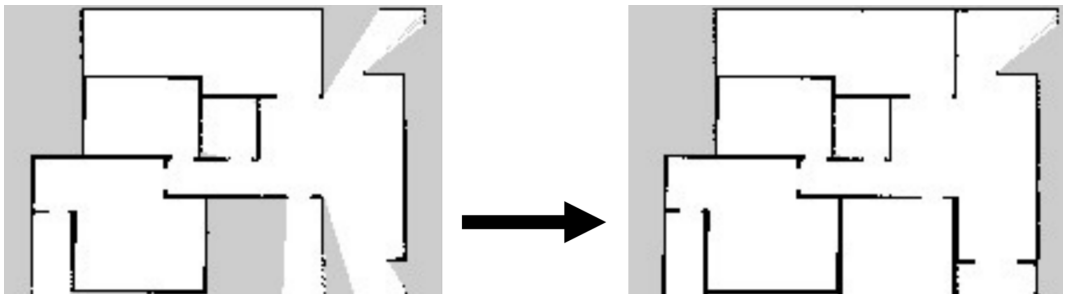


Figure 1.1 Offline global map completion through scene extrapolation. A partially observed occupancy map is extended by predicting unknown regions based on the observed spatial structure.

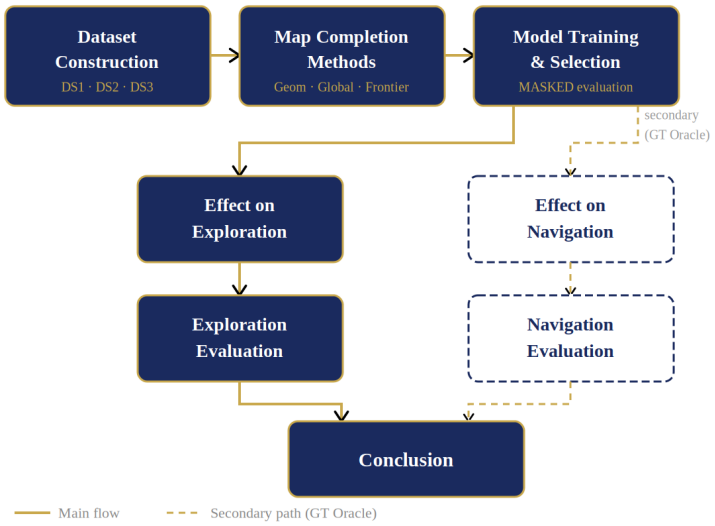


Figure 1.2 Research workflow of this thesis. The main flow (solid lines) proceeds from dataset construction through map completion methods, model training and selection, to system integration and exploration evaluation. The secondary path (dashed lines) represents the GT Oracle navigation study, conducted as an exploratory by-product.

layouts).Without extrapolation, the robot may repeatedly commit to frontiers that yield minimal information gain or that lie adjacent to walls with no navigable continuation. By estimating the occupancy structure beyond a frontier boundary, scene extrapolation can support the selection of more informative exploration targets and **avoid committing to wall-adjacent frontiers**, enabling more meaningful frontier ranking.

In the proposed system, the 95% coverage milestone serves as the primary evaluation boundary: beyond this point, the remaining unknown cells are geometrically isolated corner pockets that frontier-based exploration deliberately defers, and that are better addressed by offline global completion or human-assisted map verification.

Accordingly, this thesis focuses on **map prediction** (scene extrapolation) using learning-based models that complete unseen regions of an occupancy grid from partial map evidence. Rather than treating completion as a purely visual task, the thesis evaluates its **downstream impact** through integration into a ROS-based exploration pipeline.

A critical architectural decision distinguishes this work from prior approaches: map completion is used *exclusively at the frontier selection stage*. The path planner and the navigation stack operate throughout on the unmodified SLAM-generated occupancy map. This strict separation ensures that any observed performance difference between configurations is attributable solely to the frontier selection policy, enabling clean experimental attribution. A supplementary single-environment oracle study (Chapter 5) observes the effect of ideal local completion combined with wall closing and frontier filtering on navigation time; this is treated as an exploratory by-product observation rather than a system-level design target.

Model Families Considered in This Thesis

A core component of this research is a structured and comparative evaluation of multiple model families, selected to represent both classical baselines and modern generative paradigms:

- **VAE-family models:** Conditional Variational Autoencoders (CVAE) for conditional generation with an explicit latent space.
- **GAN-family models:** Conditional GANs in Pix2Pix-style form with hinge-loss variants, and a segmentation-oriented GAN variant (Seg-GAN).

- **Transformer-family models:** Vision Transformer (ViT) encoder–decoder architectures and an ImageGPT-style causal transformer baseline.
- **Diffusion-family models:** A RePaint-inspired diffusion formulation for unknown-region completion through iterative denoising.
- **Convolutional completion baselines:** Residual U-Net models and several U-Net variants (attention-gated, deeper multi-scale, class-weighted, patch-size adaptations).
- **Inpainting-inspired CNNs:** A LaMa-style architecture augmented with Fourier features to inject global frequency-domain context.
- **Sequence baselines:** A row-wise LSTM baseline to quantify how much structure can be captured with 1D sequential modelling.
- **Rule-based geometric baseline:** A training-free method based on door detection and structural line inference, providing a lower-bound reference.

This breadth is essential because occupancy map completion is not a single, fixed task: the appropriate model depends on how the partial observation is defined (directional masking vs. frontier-centered unknown regions) and how the prediction is used (offline map quality vs. online frontier scoring).

The Deployed System: Frontier-Centered Patch Completion

The primary experimental configuration deploys a **residual UNet** trained on frontier-centered 32×32 patches extracted from realistic SLAM maps. For each candidate frontier, a patch is extracted from the live SLAM costmap, completed by the UNet via a batch ROS service, and the resulting free-space gain r_f augments the frontier cost function. All other aspects of the exploration and navigation pipeline—frontier detection, path planning, velocity control—remain unchanged.

This design contrasts with the work that used a CVAE decoder conditioned on the upper half of a patch to predict the lower half (upper-to-lower formulation). While that formulation is studied as a controlled benchmark for directional conditional generation, adapting it to live frontiers would require rotating each patch to align with the robot's heading direction before inference—introducing additional computational overhead per frontier—and even then the known/unknown boundary would not necessarily follow a clean upper/lower split, since frontier shapes are inherently irregular. The frontier-centered full-patch setting avoids these

assumptions entirely and is the robotically relevant formulation deployed in this thesis.

Datasets and Completion Formulations

This thesis evaluates map prediction under three completion formulations:

- **Directional local completion (upper-to-lower):** The upper half of a 32×32 patch is provided and the lower half is generated. This formulation supports controlled analysis of conditional generation under a fixed directional prior and was the subject of the initial generative study [1]. It is not used in the deployed exploration system.
- **Frontier-centered local completion (full-patch):** A 32×32 patch centred on a frontier contains an irregular unknown region; the model predicts completion while preserving known cells. This is the primary formulation and the one deployed online.
- **Global map completion:** Full partial maps resized to 128×128 are completed to support offline model family comparison [2].

To ensure realistic training signals, the dataset construction deploys a TurtleBot3 robot in Gazebo across environments derived from the HouseExpo floor plan collection [3], collects sequential SLAM maps at Voronoi crossing points, and extracts frontier-centered patch pairs offline [4]. Dataset splits are structured to prevent environment-level leakage (patches from the same environment are assigned exclusively to train, validation, or test). A static mathematical conversion pipeline also provides a large-scale controlled dataset for the directional completion study [5].

Throughout this thesis, `monospace font` is used to denote implementation-specific identifiers such as class names, ROS package names, and configuration keys.

The interaction between the SLAM module, the exploration module, and the proposed map completion component is illustrated in Figure 1.3. The SLAM module continuously updates the occupancy map, which is used by the exploration module (`explore_lite++` — an extended and optimised version of `explore_lite` [6] with map completion integrated). Within this pipeline, the proposed map completer is integrated between frontier detection and goal selection,

enabling frontier-based local map completion. Specifically, local patches around candidate frontiers are processed to predict unknown regions, allowing the system to better estimate the potential information gain of each frontier. The navigation stack (`move_base`) then generates movement commands based on the selected goal.

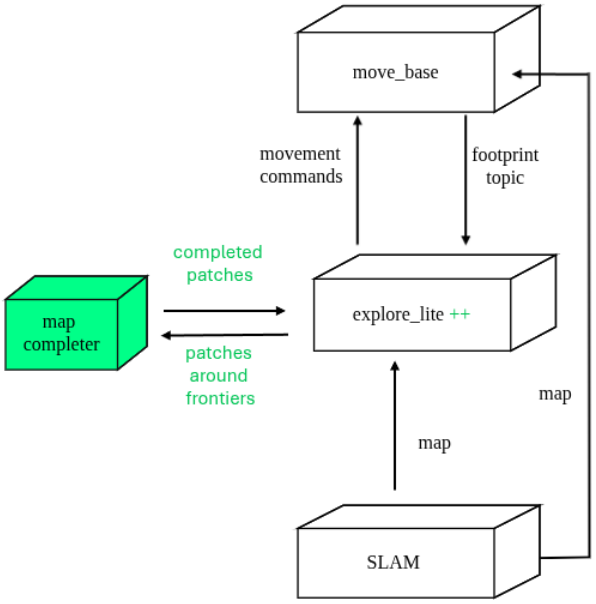


Figure 1.3 System architecture of the completion-aware exploration pipeline. The map completion module is integrated between frontier detection and goal selection, enabling frontier-based local map completion during exploration. The SLAM module provides the occupancy map, while the navigation stack (move_base) generates motion commands.

Figure 1.4 illustrates how frontier-based local map completion is applied online during exploration. For each candidate frontier, a local patch is extracted from the SLAM map and completed by the model, allowing the system to estimate potential information gain before selecting the next goal.

1.3 Problem Definition and Hypothesis

Let $M \in \{0, 1, 2\}^{H \times W}$ denote an occupancy grid map, with cells encoded as free (0), occupied (1), or unknown (2). The map is incrementally built from sensor data; let M_{obs} and M_{miss} denote the observed and unknown cell sets respectively. The central problem is predicting M_{miss} from M_{obs} via a learned mapping $\mathcal{F}(\cdot;\, \theta)$, where θ may correspond to a discriminative model (e.g. U-Net) or a generative one (e.g. CVAE, GAN, diffusion).

This thesis addresses the problem under three completion formulations —

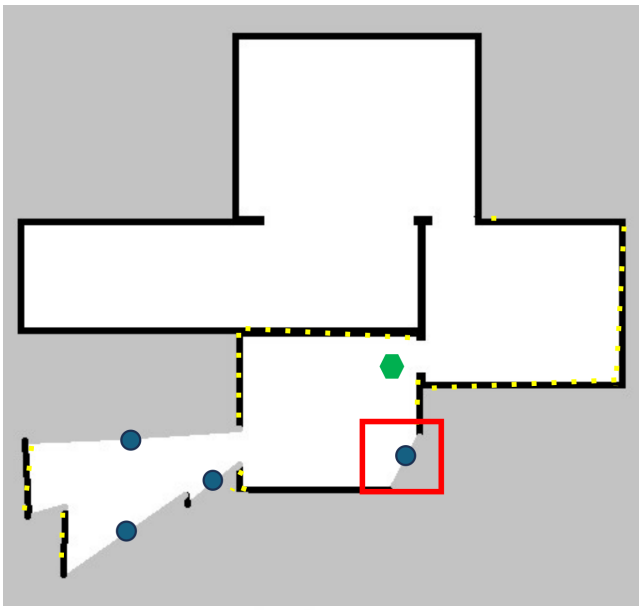


Figure 1.4 Online illustration of frontier-based local map completion during exploration. A local patch around each candidate frontier is extracted and completed, and the predicted free-space gain is used to influence frontier selection.

upper-to-lower, frontier-centered, and global — each differing in how M_{miss} is defined and how the prediction is used. The formal mathematical definitions, per-formulation notation, and model-specific training objectives are given in Chapter 3 (Section 3.1).

In the frontier-centered deployment setting, the completed patch $\hat{M}$ is not written back to the SLAM map. Instead, it informs the frontier cost function through a predicted free-space gain score and a predicted-on-wall flag. The augmented cost formulation and its parameters are detailed in Chapter 4 (Section 4.8).

Hypothesis.

The central hypothesis of this thesis is twofold:

$\mathcal{H}_1$ **(Predictive Completion)** A learning-based model can infer the unobserved occupancy region $\hat{M}_u$ from a partially observed local map M_o:

$$M_o \longrightarrow f_\theta \longrightarrow \hat{M}_u$$

$\mathcal{H}_2$ **(Exploration Efficiency)** Incorporating $\hat{M}_u$ into the frontier selection cost function reduces total exploration time compared to classical frontier-based

exploration:

$$C(f_i) \leftarrow C(f_i, \hat{M}_u^{(i)}) \Rightarrow \text{faster coverage}$$

Figure 1.5 illustrates the expected effect of map completion: by predicting obstacles in unknown regions, the robot is redirected toward genuinely free areas rather than frontiers concealing potential obstacles.

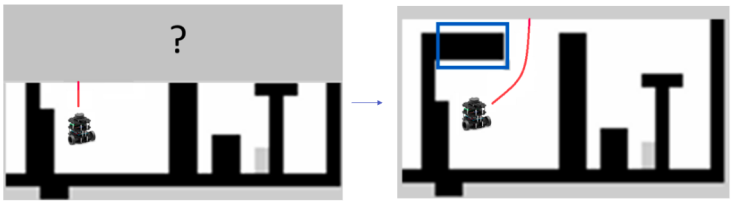


Figure 1.5 Illustrative example of map completion guiding robot navigation toward a more informative region. The predicted obstacle in the unknown region redirects the robot from its default path toward a genuinely free area."

1.4 Research Motivation and Background

This thesis evolved through a series of deliberate design decisions, each motivated by practical observations made during the preceding stage. Understanding this progression clarifies why multiple dataset pipelines and completion formulations are studied, rather than a single unified approach.

The initial plan was to construct a global sequential map dataset (Dataset 1) and evaluate global map completion directly on full SLAM maps. However, two limitations became apparent: first, resizing maps to a fixed resolution introduced structural information loss that was expected to affect completion quality; second, the computational cost of large-scale Gazebo-based data collection constrained the dataset size. These observations motivated a shift toward local completion, where smaller patches retain structural detail and data collection is more tractable.

The first local formulation explored was directional completion (upper-to-lower), inspired by the MNIST half-image prediction setting of ImageGPT [7]. This was initially conceived as a controlled feasibility study to assess whether generative models could learn occupancy structure from partial observations. When results were promising, the formulation was considered for exploration integration — rotating patches to align with the robot's heading direction. However, preliminary integration experiments revealed that even small additional operations per frontier (patch rotation, alignment) measurably affected pipeline latency. Furthermore, the

directional formulation required realistic SLAM-based training data; collecting this at scale via Gazebo was costly. This motivated Dataset 2, a mathematically derived pipeline that generates large-scale directional patch data without simulation.

Global map completion was subsequently studied using Dataset 1 — which had been extended to its current size by this stage — motivated by two primary goals: assessing whether CVAE generalises to a realistic SLAM-based dataset, and contributing a multi-model family comparison to the literature. Reviewer feedback recommended including UNet and RePaint; UNet proved strongest, directly motivating its selection for frontier-centered deployment.

The frontier-centered formulation emerged as the most robotically relevant setting. Rather than adapting the directional formulation to live exploration — which would require per-frontier patch rotation and still not produce the irregular unknown regions characteristic of real frontiers — a new dataset (Dataset 3) was constructed by applying the same BFS frontier detection used at deployment time to sequential SLAM maps from Dataset 1. This ensures that the training patch distribution closely reflects the patches encountered during live exploration.

Throughout this progression, each stage informed the next: dataset design, formulation choice, model selection, and system integration were developed iteratively rather than planned in full at the outset.

1.5 Research Objectives

This thesis aims to bridge the gap between predictive scene completion and practical robotic autonomy. The objectives are:

- **Develop a data pipeline aligned with robot exploration:** Generate occupancy map pairs and frontier-centered patches from realistic SLAM trajectories, ensuring reproducibility and preventing environment-level train/test leakage.
- **Design and compare multiple map completion formulations:** Evaluate directional completion, frontier-centered local completion, and global completion under task-appropriate protocols, with MASKED evaluation as the primary criterion for frontier-centered models.
- **Implement and benchmark diverse model families:** Compare CVAE, Pix2Pix-style GANs (including Seg-GAN), ViT, ImageGPT-style transformers, RePaint-style diffusion, LaMa-style Fourier-augmented

CNNs, U-Net variants, LSTM baselines, and a rule-based geometric baseline under unified protocols.

- **Integrate completion into frontier scoring:** Incorporate predicted free-space gain and obstacle prediction into the frontier cost function of the `explore_lite` ROS package without modifying the navigation stack. Frontiers whose neighbourhood is predicted to lie inside a wall receive a fixed penalty, while reachable frontiers are rewarded proportionally to their predicted free-space gain. This dual signal enables controlled attribution of performance gains.
- **Quantify downstream impact:** Evaluate changes in exploration efficiency using milestone-based elapsed time across eight HouseExpo environments, and conduct a supplementary single-environment oracle observation to contextualise the potential of navigation-map integration. To ensure results are robust against stochastic navigation behaviors and OS-level timing variances, all experiments are conducted with multiple repetitions per environment, allowing for a statistical evaluation that mitigates the impact of outlier trajectories.

1.6 Contributions

This thesis contributes to both map completion methodology and robotic system integration:

1. **A modular dataset and preprocessing pipeline:** A reproducible pipeline for collecting realistic occupancy maps and extracting training pairs and frontier-centered patches from exploration trajectories [4, 5]. Key methodological decisions include switching from all Voronoi points to crossing points only, manual near-duplicate pruning, and environment-level split assignment to prevent data leakage across train/validation/test sets.
2. **A comprehensive model suite for occupancy map completion:** Adaptation and evaluation of multiple model families (CVAE, Pix2Pix-style GANs including Seg-GAN, ViT, ImageGPT-style transformers, RePaint diffusion, LaMa-style Fourier CNNs, U-Net variants, LSTM baselines, and a rule-based geometric baseline) under three completion formulations and unified FULL/MASKED evaluation protocols. The key finding is that frontier-centered local completion is a *geometry-dominant* task in which convolutional U-Net architectures outperform attention-based and sequence-based alternatives regardless of training data size.

3. **Frontier scoring with predicted information gain:** A completion-aware frontier ranking mechanism integrated into the `explore_lite` ROS package. The mechanism extracts a 32×32 frontier-centered patch and completes it via a batch UNet ROS service. The resulting free-space gain score and a predicted-on-wall penalty are incorporated directly into the existing distance-size cost function, effectively de-prioritizing frontiers predicted to be impassable while rewarding those with high expected utility.

4. **System-level integration and evaluation:** A modular ROS pipeline allows map completion to influence frontier selection while keeping path planning and navigation unchanged, enabling controlled attribution of performance gains. Experimental evaluation across eight HouseExpo environments demonstrates that **MC outperforms the classical baseline in 80.4% of milestone comparisons (45/56)**, with an average **speedup of 7.2%** across all milestones. The most practically relevant gain occurs at the **95% coverage milestone**—the point at which remaining frontier cells correspond to geometrically constrained corner pockets that the system deliberately defers and that can be resolved offline or via human-assisted methods—where MC achieves a consistent **+10.4% speedup across all eight environments**. These improvements are observed to be **statistically significant** across repeated trials.

   A supplementary single-environment ground-truth oracle study, conducted with repeated runs, records a **+48% navigation improvement** under ideal local completion with wall closing and frontier filtering [8, 9]. This result is reported as an **auxiliary observation**; the improvement is partly attributable to wall-gap path planning artifacts arising from the gap-closing mechanism, and the study is not intended as a performance target for future integration work.

1.7 Scope and Limitations

Scope

- **Map representation:** 2D occupancy grids with three semantic states (free/occupied/unknown), compatible with standard ROS navigation.
- **Simulation environments:** Indoor, static environments (HouseExpo/Gazebo-derived) enabling controlled, reproducible experiments.
- **Completion tasks:** Directional local completion (benchmark), frontier-centered local completion (deployed), and global completion

(offline comparison).

- **System integration:** Completion-aware frontier selection via predicted free-space gain, where wall-aware penalisation is incorporated directly into the frontier cost formulation. Path planning and navigation operate exclusively on the unmodified SLAM map.

Limitations

- **Simulation-only validation:** Results are validated in Gazebo simulation; real-world noise, sensor limitations, and long-term SLAM drift remain as future work.
- **Static-environment assumption:** Dynamic obstacles and non-stationary layouts are not modelled.
- **2D-only prediction:** No semantic labelling or 3D reconstruction is addressed.
- **Fixed patch size:** The deployed UNet uses a fixed 32×32 prediction window, which is insufficient for environments whose relevant structural features exceed the patch scale. The `4bb74477` oracle analysis demonstrates this limitation and motivates adaptive patch sizing.
- **Compute constraints for heavy models:** Transformer and diffusion models are not suitable for real-time frontier scoring at current hardware speeds, motivating careful task-dependent model selection.

1.8 Thesis Organisation

The remainder of this document is organised as follows:

- **Chapter 2: Literature Review** — Existing work on scene extrapolation, map completion, generative modelling in robotics, and frontier-based exploration systems. Includes a rationale for the `explore_lite` backbone choice.
- **Chapter 3: Map Completion** — All model architectures for directional, frontier-centered, and global completion, including the rule-based geometric baseline and U-Net variants. Dataset references are minimal here; full construction and split details are given in Chapter 6.

- **Chapter 4: Map Completion in Frontier-Based Exploration** — The complete exploration pipeline: frontier clustering, batch completion service (UNet model), completion score computation, and target selection. The map completion module is exposed as a ROS service following a client–server paradigm, where frontier-centered patches are transmitted in batches and completed maps are received asynchronously.

- **Chapter 5: GT Oracle Navigation Study** — A supplementary single-environment study using ground-truth floor-plan patches as a perfect local oracle. The study characterises the navigation benefit achievable with ideal local completion combined with wall closing and frontier filtering. Results are reported as an exploratory observation rather than a performance target; the relationship between this result and the deployed UNet system is discussed in Chapter 8.

- **Chapter 6: Datasets** —Construction and curation of the three dataset families used in this thesis: the static mathematical conversion dataset for directional completion, the Gazebo-based global sequential map dataset, and the frontier-centered patch dataset (SDS and BDS variants) extracted from the global sequential map dataset.

- **Chapter 7: Experiments** — Quantitative and qualitative comparisons across all completion formulations under FULL and MASKED evaluation protocols; comprehensive system-level exploration evaluation across eight HouseExpo environments with milestone-based analysis.

- **Chapter 8: Discussion** — Interpretation of findings, including the quality–efficiency relationship, spatial scale effects, the gap between isolated metrics and system-level benefit, and limitations.

- **Chapter 9: Future Work** — Directions toward larger patch sizes, tighter navigation coupling, physical robot validation, and longer-term extensions.

- **Chapter 10: Conclusion** — Summary of contributions and key findings of this thesis.

2
LITERATURE REVIEW

Robotic scene understanding involves perceiving, interpreting, and interacting with environments through sensor data and intelligent algorithms. To model indoor spatial structures, the robotics community predominantly adopts the occupancy grid paradigm, which effectively partitions the environment into a discrete lattice. Within this framework, each spatial unit is assigned a state reflecting its navigational status: traversable free space, occupied obstacles, or unobserved regions. While conventional mapping algorithms update these states through direct sensor measurements, they remain fundamentally limited by the line-of-sight constraints of the platform.

The emergence of **learning-based scene extrapolation** has substantially advanced this paradigm [10]. Deep neural networks (CNNs, transformers, etc.) can now predict unobserved regions by learning spatial priors from data. This capability is particularly valuable for applications requiring efficient exploration under partial observability, such as search-and-rescue or warehouse robotics.

Unlike existing studies that focus strictly on either global or local completion, this thesis bridges this gap by evaluating both scales and integrating a frontier-centric local completion model directly into a live ROS navigation stack. This allows for a system-level assessment of how predictive spatial priors influence real-time exploration decisions.

The studies can be categorized as in Table 2.1.

To provide a structured overview of the current research landscape, existing methodologies are categorized herein based on two primary dimensions: the functional role of spatial prediction within the mapping process, and the underlying algorithmic strategy employed for autonomous exploration. A detailed taxonomy of map-prediction paradigms is presented in Table 2.2, while the diverse exploration methodologies are systematically compared in Table 2.3.

Table 2.1 Comprehensive categorization of reviewed studies by functional roles

Category	Representative Studies
Exploration-Oriented Systems	[11], [12], [13], [14], [15], [16], [17], [18]
SLAM Completion / Artefact Removal	[19], [20], [21], [13], [15]
Navigation Augmentation	[22], [16], [23], [24], [25], [26], [18]
Semantic Scene Understanding	[27], [28], [29], [21]
Multi-Agent Coordination	[17], [30]
Simulation-Only Evaluation	[25], [26], [20], [18]
Rule-Based / Geometric Completion	[9], [21]
Frontier Scoring and Decision-Making	[12], [11], [15]
Uncertainty-Aware Mapping	[12], [13], [16], [18]

Table 2.2 Map prediction models in reviewed studies

Study	Model Type	Input Format	Prediction Metrics	Occ. Grid
[19]	U-Net / pix2pix	Partial map + mask	F-measure	Yes
[22]	ResNet18 + U-Net decoder	RGB-D egocentric	IoU, Coverage	Yes
[18]	U-Net Ensemble + ResNet18	Proj. depth image	IoU, Coverage, Variance	Yes
[14]	U-Net / pix2pix	5m×5m partial map	IoU	Yes
[13]	Multi-Head U-Net	Partial map + mask	Pixel-wise NLL	Yes
[15]	ResNet-VAE	Binary partial map	BCE, SSIM	Yes
[28]	CNN + adversarial	Partial RGB-D pano	L1, GAN, scene attr. loss	No
[29]	CNN-SE + PixelCNN	Voxel scene	Cross-entropy	No
[26]	CNN + GAN	Cube-map RGB-D	Chamfer, F-score	No
[25]	Bidirectional LSTM	Partial viewgrid	MSE, SSIM	No
[16]	CNP (Neural Process)	Sparse point cloud	MSE, Entropy	Yes
[23]	CNN + confidence net	RGB-D semantic	IoU, SPL	No
[24]	LSTM (amodal)	RGB top-down	SPL, Coverage	No
[11]	U-Net + Transformer + FFC	3-channel occupancy	PSNR, SSIM, LPIPS, FID	Yes
[12]	LaMa Ensemble (n_p=3)	Partial occupancy map	Coverage, IoU, TU	Yes
[20]	CycleGAN / CUT	Noisy occupancy map	FID, pixel error	Yes
[17]	LaMa / Stable Diffusion	Global occupancy map	FID, SSIM	Yes
[21]	Rule-based geometry	Wall/door segments	IoU, Coverage	Yes
[9]	Heuristic completion	Door detection	IoU	Yes

Table 2.3 Exploration methods in reviewed studies

Study	Exploration Strategy	Frontier-Based	Modular	ROS-Compat.
[19]	Post-SLAM completion (no exploration control)	No	Yes	Partial
[22]	RL-based goal-directed navigation	No	No	No
[18]	Uncertainty-aware RRT path selection	No	Yes	No
[14]	Offline prediction (no exploration integration)	No	No	No
[13]	Global uncertainty scan	No	Partial	No
[15]	Flood-fill information gain	Yes	Partial	Yes
[16]	Prediction-informed motion planning	No	Yes	Yes
[23]	RL-based object-goal navigation	No	No	No
[24]	Amodal map prediction for room navigation	No	No	No
[11]	Local prediction-based frontier extraction	Yes	Yes	Yes
[12]	Uncertainty-driven frontiers	Yes	No	No
[20]	No exploration (SLAM post-processing only)	No	Yes	Partial
[17]	Multi-agent generative entropy	No	Yes	No
[21]	Offline room completion	No	No	No
[9]	Door-based expansion	No	No	No

In addition to methodological and architectural differences, the choice of dataset plays a critical role in shaping both the problem formulation and the evaluation of map completion and exploration systems. The reviewed studies employ a range of datasets with varying levels of realism, from synthetic floor plans and procedurally generated environments to real-world RGB-D scans and simulation-based SLAM trajectories. Tables 2.4 and 2.5 provide a unified reference: Table 2.4 summarises dataset properties, and Table 2.5 maps each reviewed paper to the datasets it uses. Individual paper sections describe only how each dataset was specifically processed or partitioned for that work.

Table 2.4 Datasets used in reviewed studies: properties

Dataset	Type	Brief Description
KTH [31]	Real-world 2D	149 campus floor plans as XML wall/door annotations; converted to 0.1 m/cell occupancy grids
HouseExpo [3]	Synthetic 2D	≈35,000 2D indoor layouts derived from SUNCG; used with PseudoSLAM or Gazebo/ROS for exploration
LIFULL HOME'S [32]	Real-world 2D	Real Japanese apartment layouts converted to occupancy grids
Gibson [33]	3D RGB-D sim.	Photorealistic 3D indoor reconstructions used inside the Habitat simulator
Matterport3D [34]	3D RGB-D real	Real-world RGB-D scans of 90 indoor environments; widely used for embodied AI
SUNCG [35]	3D synthetic	>45,000 synthetic 3D indoor scenes with semantic labels; voxelized for scene completion tasks
ScanNet [36]	3D RGB-D real	1,513 real-world RGB-D indoor scans with semantic annotations
SUN360 [37]	Panoramic RGB	360° spherical panoramas from 26 scene categories (6,174/1,013/1,805 train/val/test)
ModelNet [38]	3D objects	CAD object models rendered as multi-view grayscale images; 40- and 10-class variants
S3DIS [39]	3D RGB-D real	Stanford Large-Scale 3D Indoor Spaces; 6 areas, 271 rooms
MRPB 1.0 [40]	2D sim.	Unified benchmark for mobile robot local planning

The following sections analyze these works in detail, comparing their methodologies, architectural choices, and limitations.

2.1 MCN-SLAM: Map Completion with Deep Generative Models for SLAM [19]

Tasks and Goal

The primary task addressed in [19] is improving SLAM performance under partial observation by completing unobserved regions of occupancy grid maps. The authors propose MCN-SLAM, a SLAM framework that extends FastSLAM by

Table 2.5 Datasets used in reviewed studies: paper–dataset mapping

Study	Datasets Used
[19]	HouseExpo, LIFULL HOME'S
[22]	Gibson (train), Matterport3D (transfer test)
[18]	Matterport3D
[27]	Matterport3D via VLN-CE / R2R benchmark
[14]	Custom (2 SolidWorks environments; TurtleBot2 physical)
[13]	Custom (ROS bags + Google Cartographer)
[15]	KTH
[28]	SUNCG (train), Matterport3D (fine-tune & eval)
[29]	SUNCG
[26]	SUNCG, Matterport3D, ScanNet
[25]	SUN360, ModelNet
[16]	Custom (75 procedural maze environments)
[30]	Matterport3D, S3DIS
[23]	Matterport3D
[24]	Matterport3D
[11]	KTH (train), HouseExpo, MRPB 1.0 (cross-domain test)
[12]	KTH
[20]	HouseExpo via modified PseudoSLAM
[17]	Custom (procedural urban scenes)
[21]	Real-world Radish/Intel Lab maps; custom Stage simulation

integrating a pre-trained map completion network. The goal is to infer full environment maps from limited sensor data using prior knowledge learned from other maps, enhancing the mapping output while maintaining localization through particle filtering.

Role of Map Prediction

In MCN-SLAM, the map prediction module is used to complete the occupancy map after each particle generates its own partial map. This module does not influence the FastSLAM particle filter directly—it does not affect the motion model or the weight update process. Instead, the generated full map serves as a post-processing enhancement, refining the global map for evaluation or planning. The authors suggest that, in future work, the completed maps could be used to influence particle weights or be incorporated into SLAM feedback loops for deeper integration.

Dataset and Its Preparation

Two datasets are used for training and evaluation:

- **HouseExpo** [3]: A large-scale synthetic dataset of 2D indoor floor plans.

Used to generate 5,000 training, 200 validation, and 100 test samples.

- **LIFULL HOME'S** [32]: Real apartment layouts converted into occupancy grid maps, with 3,500 training, 200 validation, and 100 test samples.

Ground-truth maps represent complete environments, while partial maps are created either by removing regions or simulating initial SLAM outputs (using PseudoSLAM). These input-output pairs are used to train the generative models offline.

Models and Approaches

Map Prediction Model:

- **U-Net:** Trained with L2 loss for direct image reconstruction.
- **pix2pix (U-Net + PatchGAN):** Trained with a combination of L2 loss and adversarial loss. A **PatchGAN** discriminator focuses on local consistency, and **Spectral Normalization (SNGAN)** is applied to stabilize training.

The input consists of a two-channel image: one channel for the partial map and one binary mask indicating unknown areas. Output values are continuous but post-processed into discrete occupancy labels.

Evaluation Protocol

Scope: FULL — metrics computed over the entire generated map (m_s vs. ground truth m_T).
Approach: Tip B (blending) — the U-Net generator preserves observed cells via skip connections; unobserved cells are filled by the network. Observed pixels are carried forward without modification, making the known region trivially correct and inflating FULL metrics.
Metrics: Accuracy, Precision, Recall, F-measure (binary occupancy class classification).

Novelties, Strengths, and Limitations

Novelties:

- First integration of a generative map completion module into a FastSLAM framework.

- Use of spectral-normalized PatchGAN for stable and high-quality map prediction.

Strengths:

- Significant improvement in map completeness (e.g., F-measure) over baseline methods.
- Demonstrates generalization to real and simulated maps (HOME'S and HouseExpo).
- Flexible architecture supporting U-Net and GAN variants.

Limitations:

- The model is trained offline and does not update during exploration.
- Map completion does not influence localization (particle weighting) or robot planning.
- Requires full occupancy maps and masks for training and inference, making deployment less modular.

Comparative Analysis

[19] embeds map prediction directly inside the FastSLAM particle filter, completing each particle's partial map before the SLAM update. The present work takes the opposite architectural stance: SLAM (GMapping) and prediction are fully decoupled. Completed 32×32 frontier patches from the residual U-Net (see Chapter 3) act as an informative prior only for frontier ranking via `explore_lite`, without touching SLAM internals. This modularity ensures compatibility with any ROS navigation stack. While [19] suggests that completed maps could influence particle weighting in future work, the current design prioritises system-level stability over tighter coupling.

2.2 Occupancy Anticipation for Efficient Exploration and Navigation [22]

Tasks and Goal

The primary task addressed in [22] is goal-directed navigation in partially observed 3D environments, specifically within the context of the Habitat PointNav Challenge.

The objective is to enable a robot to reach a designated goal location efficiently by predicting the occupancy of regions beyond its current field of view. To do this, the authors propose an occupancy anticipation framework that supports both navigation and exploration by enhancing spatial awareness.

Role of Map Prediction

In the proposed framework, map prediction is central to the overall policy. The system predicts local 2D top-down occupancy maps from egocentric RGB-D observations. These predicted maps are not used to update the SLAM map directly but are instead provided as inputs to both the global and local policy networks. The predicted occupancy helps determine subgoals and navigation decisions in real time, guiding the robot through unobserved or occluded regions.

Dataset and Its Preparation

Training uses 72 Gibson scenes, with validation split into "small" and "large" subsets; Matterport3D is used for transfer testing. Approximately 1.5–2 million frames of agent experience are collected during training. Observations are RGB-D images of size 128×128; ground-truth occupancy maps are generated from the 3D scene meshes.

Models and Approaches

Map Prediction Model: A deep CNN model is used for occupancy anticipation:

- **Encoder:** ResNet18 (ImageNet pre-trained, first two blocks used for RGB features)
- **Visible occupancy map:** Extracted from projected depth using geometry
- **Decoder:** A U-Net-style decoder with skip connections merges RGB and depth features to predict a $2 \times V \times V$ occupancy tensor (for occupied and explored probabilities).

Policy Learning:

- **Global Policy:** Trained with PPO to select long-term subgoals based on predicted occupancy.
- **Local Policy:** Also trained with PPO to move toward subgoals using short-term action decisions.

Evaluation Protocol

Scope: FULL — metrics over the entire $V \times V$ anticipated occupancy patch.
Approach: Tip A (raw reconstruction) — the model predicts the complete anticipated map including both visible and previously unobserved regions; all pixels contribute to metric computation.
Metrics: IoU (%), F1 score (%) for free and occupied classes; Map Accuracy (m^2) and Coverage (m^2, %) for system-level evaluation.

Novelties, Strengths, and Limitations

Novelties:

- Introduces the concept of *occupancy anticipation* for inferring unobserved regions from RGB-D input.
- Incorporates an *anticipation-based reward* to train exploration policy.
- Achieved state-of-the-art results in the Habitat PointNav Challenge 2020.

Strengths:

- Demonstrates strong generalization to unseen environments.
- Learns a navigation policy that adapts to partial observability.
- Achieves high exploration and navigation efficiency.

Limitations:

- Requires RGB-D input, which may not always be available on low-cost robots.
- Depends on large-scale training (millions of frames) in simulation.
- End-to-end nature makes the system less modular and harder to deploy with existing ROS stacks.

Comparative Analysis

Unlike [22], which targets 3D point-goal navigation using RGB-D input within the Habitat simulator, our system operates on 2D occupancy grids derived from SLAM.

Where their end-to-end policy requires millions of training frames and RGB-D sensors, our approach decouples prediction from control: generative models score frontiers locally, while standard ROS planners handle motion — making the system easier to deploy on low-cost robots with 2D LiDAR only.

2.3 Uncertainty-Driven Planning for Exploration and Navigation [18]

Tasks and Goal

The work presented in [18] addresses two primary robotic tasks: autonomous exploration and point-goal navigation in previously unseen indoor environments. The goal is to enable an agent to make informed path decisions under partial observability by learning occupancy priors beyond its current field of view. The system selects paths based on both predicted occupancy maps and model uncertainty to improve navigation efficiency and map coverage.

Role of Map Prediction

Map prediction is central to the planning strategy. An ensemble of U-Net models is trained to predict local occupancy maps from ground-projected egocentric depth images. These predicted maps serve two purposes: (1) they estimate occupancy probabilities for unobserved areas, and (2) they allow computation of epistemic uncertainty by measuring the variance across ensemble outputs. These occupancy predictions and uncertainty maps are then used to evaluate and rank candidate paths generated by a sampling-based planner.

Dataset and Its Preparation

The system is trained and evaluated using Matterport3D (90 scenes) within the Habitat simulator, following the standard train/val/test split. Training pairs are sampled across shortest paths between randomly selected locations; the exact number of training samples is not reported. The ensemble of 4 models is evaluated across multiple simulated episodes at test time.

Models and Approaches

Map Prediction Model: The system uses an ensemble of U-Net models, each with a ResNet18 encoder, trained via pixel-wise cross-entropy loss. Each model predicts a local occupancy map given a partial observation. The mean of the ensemble

outputs is used for occupancy estimation, and the variance is used as a proxy for epistemic uncertainty.

Planning and Navigation:

- **Path Generation:** Rapidly-Exploring Random Trees (RRT) are used to sample candidate paths in the environment.
- **Path Selection:** For exploration, the path with the highest average model uncertainty is selected to maximize information gain. For point-goal navigation, an upper-confidence-bound (UCB) style objective combines path traversability, uncertainty, and length.
- **Execution:** The selected path is passed to a DD-PPO (Distributed Proximal Policy Optimization) controller, which acts as a local path follower.

Evaluation Protocol

Scope: FULL — metrics over the accumulated global map.
Approach: Tip B (blending) — predictions are registered to the global map via Bayesian updates; direct sensor observations override predictions, making observed regions trivially correct and inflating FULL metrics.
Metrics: Map Accuracy (m^2), IoU (%), Coverage (m^2, %).

Novelties, Strengths, and Limitations

Novelties:

- Introduces an uncertainty-aware path planner that uses model disagreement (epistemic uncertainty) to guide exploration and navigation.
- Integrates learned occupancy priors with classical RRT sampling and modular control via DD-PPO.

Strengths:

- Provides a principled method to reason about unobserved regions using learned priors.
- Demonstrates strong performance improvements in both exploration and point-goal navigation in simulation.

- Modular: can be integrated with other navigation systems like DD-PPO without retraining.

Limitations:

- Relies on idealized, noise-free environments provided by the Habitat simulator; no SLAM or sensor noise is considered.
- Uses perfect ground-truth pose for map registration, which is unrealistic in real-world scenarios.
- Requires ensemble inference at runtime, which can be computationally expensive.

Comparative Analysis

Both approaches use learned occupancy priors to support exploration under uncertainty. The key distinction is that [18] requires ensemble inference at runtime (U-Net $\times$ N) and relies on perfect ground-truth poses in Habitat — introducing computational overhead and unrealistic localization assumptions. Our approach uses a residual UNet requiring only a single forward pass per frontier, and operates on noisy SLAM maps without pose oracle access or ensemble overhead. As a supplementary observation, this thesis also includes a GT oracle navigation study (Chapter 5) that isolates the effect of ideal local completion on navigation efficiency. While not directly comparable to [18], this exploratory study establishes a foundation for future work on tighter navigation coupling within the proposed pipeline.

2.4 Cross-Modal Map Learning for Vision and Language Navigation [27]

Tasks and Goal

The work presented in [27] addresses the task of vision-and-language navigation (VLN) in previously unseen environments. The system aims to guide an agent toward a goal location by following natural language instructions, using only egocentric RGB-D images as input. Unlike purely reactive methods, their framework explicitly builds and utilizes internal semantic map representations guided by language. The goal is to learn to associate spatial and semantic concepts in language with visual cues to improve instruction-based navigation.

Role of Map Prediction

Map prediction is a central component in the system. It enables the agent to extrapolate beyond its field of view by constructing a semantic occupancy map using depth input and instruction-guided attention. This predicted map is then used to infer a trajectory as a series of waypoints. Two stacked U-Net modules are used: the first hallucinates unobserved occupancy layout, and the second produces semantic predictions conditioned on the instruction. The cross-modal attention output is concatenated at the bottleneck, guiding the decoding process in a way similar to how partial maps are used as conditioning in CVAE-based models. However, unlike our variational models that use a stochastic latent bottleneck, their bottleneck remains deterministic, focusing on semantic alignment rather than uncertainty modeling.

Dataset and Its Preparation

The system is trained and evaluated using the VLN-CE benchmark, derived from the Room-to-Room (R2R) dataset within the Matterport3D environment using the Habitat simulator. The dataset contains 16,844 path-instruction pairs across 90 realistic indoor scenes. During training, the model receives egocentric RGB-D frames and ground-truth occupancy/semantic maps. Ground-projected maps from RGB-D inputs serve as input to the map predictor, while the instruction text is used to guide prediction. At test time, the system infers unseen map regions and navigation paths solely based on language and egocentric perception without prior exploration.

Models and Approaches

Map Prediction: The map prediction model is composed of two stacked U-Net architectures. The first U-Net hallucinates the occupancy layout based on the partial depth-based projection, and the second refines this into a semantic prediction using instruction-conditioned attention. Cross-modal attention modules are applied at the bottleneck of both U-Nets. Visual input features are extracted with a ResNet18 encoder, and instructions are encoded using a BERT-based transformer.

Path Prediction: The path is predicted as a sequence of 2D waypoints over the generated semantic map. A second attention-guided U-Net takes the predicted semantic map and instruction embeddings to generate heatmaps for waypoint locations. The path prediction module also uses auxiliary loss terms to model traversal status and starting position, helping the agent determine its progress along the instruction path.

Evaluation Protocol

Scope: FULL — metrics over the entire predicted egocentric semantic map.
Approach: Tip A (raw reconstruction) — the model outputs a full semantic map; all pixels (both observed and predicted) contribute to metric computation.
Metrics: IoU (%), F1 (%) for semantic classes; Percentage of Correct Waypoints (PCW %) for path prediction quality.

Novelties, Strengths, and Limitations

Novelties:

- Introduces a modular two-stage system with cross-modal attention at both the map and path prediction levels.
- Leverages natural language as conditioning input for both hallucination and path grounding.
- Demonstrates semantic extrapolation from RGB-D observations guided by language.

Strengths:

- Integrates semantic reasoning and instruction-following in a structured way.
- Shows strong performance on unseen environments in the VLN-CE benchmark.
- Uses modular architecture that separates map prediction from path generation.

Limitations:

- Designed for semantic navigation, not exploration; the approach does not integrate frontier-based decision-making.
- The deterministic bottleneck may limit uncertainty modeling and generative diversity.
- Uses panoramic-free egocentric RGB-D images, which may not capture long-range layout features as effectively as global maps.

Comparative Analysis

The shared insight between [27] and our work is that predicting unseen spatial structure improves downstream decisions. However, their system follows language instructions toward a known goal using semantic RGB-D maps, whereas ours performs open-ended exploration on discrete occupancy grids without semantic labels, language input, or goal specification — a fundamentally different task requiring no depth sensor or instruction corpus.

2.5 Occupancy Map Prediction using Generative and Fully Convolutional Networks for Vehicle Navigation [14]

Tasks and Goal

This study focuses on the task of predicting extended occupancy maps from sparse local observations to support high-speed autonomous navigation. The goal is to extrapolate unobserved regions of the environment based on the robot's current sensor data, enabling more informed trajectory planning under limited field of view.

Role of Map Prediction

Occupancy map prediction is the central task of this study. The model learns to expand a known partial 2D occupancy map into a larger, plausible map that includes unobserved regions. Although the authors suggest that these predicted maps could be used in downstream tasks such as model predictive control (MPC) for navigation, the paper does not implement or evaluate any active planning or control system based on the predictions.

Dataset and Its Preparation

Two custom SolidWorks environments are voxelized and converted to OctoMap (see Table 2.5). A simulated Dubins car with particle-filter SLAM generates $5\,\text{m} \times 5\,\text{m}$ input maps paired with $10\,\text{m} \times 10\,\text{m}$ ground-truth targets along four trajectories, yielding 5,221 training and 1,090 test samples. A small physical dataset collected with a TurtleBot2 + Hokuyo LiDAR is used for testing only. The limited diversity (two environments, four trajectories) causes a notable drop in performance on physical data.

Models and Approaches

The authors evaluate three deep learning architectures for map prediction:

- **U-Net:** Performs best for smaller expansion factors; preserves structure well.
- **ResNet:** Less effective overall; fails to capture detailed geometry.
- **Conditional GAN (pix2pix):** Performs comparably to U-Net in early stages but outperforms it for larger expansion sizes, producing sharper and more plausible completions.

Each model is trained on the simulated dataset using an L_1 loss (for U-Net/ResNet) or a combination of adversarial and L_1 loss (for the GAN).

Evaluation Protocol

Scope: FULL — metrics over the entire expanded output map (5 m$\times$5 m input expanded to 10 m$\times$10 m prediction).
Approach: Tip A (raw reconstruction) — the model predicts the full expanded region; no separation between known and predicted areas is applied. Both contribute to the metric.
Metrics: SSIM (structural similarity; single metric reported).

Novelties, Strengths, and Limitations

Novelties:

- One of the first works to frame occupancy map extrapolation as an image completion task.
- Explores the use of GANs (pix2pix) for robotic map prediction.

Strengths:

- Simple and effective architectures; direct comparison of feedforward vs. adversarial methods.
- Demonstrates that GANs offer more detailed completions in sparse views.

Limitations:

- The dataset is limited—only two environments and four trajectories are used for training.

- Generalization to physical data is poor due to lack of diversity in the training set.
- No actual integration of map prediction into navigation or planning.

Comparative Analysis

[14] shares the same core premise — extrapolating occupancy maps from partial observations — but stops short of planning integration and trains on only two environments. Our dataset spans a large and diverse set of environments generated during SLAM-based simulation exploration, capturing noise and occlusion that clean simulated inputs miss. Crucially, our predictions are not an endpoint: they directly score frontiers and drive goal selection in a closed exploration loop.

2.6 Uncertainty-Aware Occupancy Map Prediction Using Generative Networks [13]

Tasks and Goal

The study presented in [13] aims to support autonomous exploration by predicting unobserved regions in a 2D occupancy map and identifying areas of high uncertainty to guide exploration. The goal is to enable the robot to reason about unknown parts of the environment and proactively move toward regions where prediction confidence is low, thus maximizing information gain.

Role of Map Prediction

Map prediction is central to the framework. A deep neural network is trained to complete partial occupancy maps. To address uncertainty, the model is designed to produce multiple possible completions (multi-hypotheses), from which pixel-wise variance maps are derived. These variance maps indicate where the model is most unsure and are used to guide exploration by selecting the most ambiguous region as the next navigation target.

Dataset and Its Preparation

The dataset is generated offline using public ROS bag files and the Google Cartographer SLAM system. Full 2D occupancy maps are constructed first. Then, during a second pass through the bag files, 4 m $\times$ 4 m partial maps (centered at the robot's pose) are cropped from these full maps and paired with corresponding 6 m $\times$ 6 m ground truth maps. This process yields 4997 training samples and 1368 test

samples. Because the data is derived from fully mapped environments, the input patches are noise-free and idealized, lacking realistic SLAM artifacts, occlusions, or mapping inconsistencies.

Models and Approaches

Map Prediction Model: The core architecture is a U-Net extended with multiple output heads, each producing a different map completion hypothesis. During training, a best-match loss selects the head closest to the ground truth. At inference time, the variance across these heads is computed as a proxy for epistemic uncertainty.

Planning and Exploration: The variance map is interpreted as an uncertainty map. The robot selects the region with the highest total uncertainty score as the next target for exploration. This results in a closed-loop strategy where prediction informs planning by highlighting where new sensor data would be most valuable.

Evaluation Protocol

Scope: MASKED — metrics restricted to the *predicted* (expanded) region only; the known $4\,\mathrm{m} \times 4\,\mathrm{m}$ input region is excluded from evaluation.
Approach: The paper explicitly states evaluation "between the predicted regions of the image and the ground truth", directly measuring prediction quality over inferred cells only.
Metrics: SSIM, PSNR.

Novelties, Strengths, and Limitations

Novelties:

- Introduces a multi-hypothesis deep model for occupancy map prediction.
- Uses pixel-wise prediction variance to drive exploration in an uncertainty-aware fashion.

Strengths:

- Fully integrates map prediction with exploration strategy.
- Operates online and supports dynamic decision-making.

Limitations:

- The dataset is constructed from offline-complete maps, making inputs unrealistic for online deployment.
- The model is deterministic and needs architectural modifications (multi-heads) to simulate uncertainty.
- The region-selection mechanism is global and coarse, lacking frontier-level granularity.

Comparative Analysis

The study by [13] represents a significant predecessor to the exploration integration developed in this thesis, as both approaches utilise map prediction to inform target selection. However, the two systems diverge in terms of granularity and operational realism. While the framework in [13] computes uncertainty scores globally across the entire map, the current approach evaluates each frontier candidate independently through localised predicted patches, thereby offering finer navigational control. On patch scale: their 4 m $\times$ 4 m input (paired with a 6 m $\times$ 6 m ground truth) is broadly comparable in physical extent to our 32×32 patch at 0.1 m/cell (3.2 m), though the representation and prediction formulation differ substantially.

Furthermore, their multi-head architecture simulates uncertainty through a deterministic ensemble; in contrast, the residual UNet employed in this work eliminates the need for multi-head designs, providing a more streamlined and computationally efficient alternative without additional architectural overhead. It is also worth noting that a pure uncertainty-maximising strategy—always selecting the region with the highest uncertainty regardless of distance—can cause the robot to travel to distant high-uncertainty areas and then backtrack to complete nearby regions, potentially increasing total travel time. The cost function used in this thesis explicitly weights distance against predicted free-space gain to mitigate this over-commitment risk.

Relation to [14]

Compared to its predecessor [14], the study in [13] introduces a more advanced prediction model and is the first to actively use map prediction outputs in exploration. While [14] stops at evaluating map completion quality and only suggests possible integration into planning, [13] carries this forward by implementing an uncertainty-driven region selection strategy. However, both

studies share a common limitation: they rely on clean, limited datasets derived from a small number of environments. Specifically, [14] uses only two environments for four trajectories in training, and [13] crops from full maps generated offline, thus avoiding the challenges of online SLAM. In contrast, our dataset is generated during realistic robot exploration across a large set of diverse environments and forms a more robust foundation for learning and evaluation. Additionally, our approach combines prediction with localized frontier evaluation, offering a modular and efficient alternative to global uncertainty scanning.

2.7 Learned Map Prediction for Enhanced Mobile Robot Exploration [15]

Tasks and Goal

The study in [15] focuses on enhancing autonomous exploration in indoor environments using predictive map completion. The core goal is to improve frontier-based exploration efficiency by leveraging generative models that estimate the layout of unknown regions beyond current sensor observations. These predictions are then used to guide path planning by estimating expected information gain near frontier clusters.

Role of Map Prediction

Map prediction is central to their exploration strategy. The system uses a ResNet-based Variational Autoencoder (VAE) to predict unknown regions in occupancy maps. These predicted regions serve as inputs for computing information gain using a flood-fill algorithm. The resulting information gain scores are integrated into cost functions of two standard exploration planners: the Hector Exploration Planner and a Cost-Utility Planner. The Hector Exploration Planner builds a costmap known as an *exploration transform*, which assigns a cost to every explored free cell based on path cost to the nearest frontier and obstacle proximity; the predicted information gain is incorporated as a third term in this transform. The predicted maps help prioritize frontiers that are likely to yield high information gain, guiding the robot toward more informative regions.

The predicted maps help prioritize frontiers that are likely to yield high information gain, guiding the robot toward more informative regions.

Dataset and Its Preparation

KTH floor plans are used (see Table 2.5). Training data is generated by running Hector exploration in the Stage simulator on each floor plan, recording a 256×256 region centred on each frontier cluster. The centre 80×80 patch is masked to form the prediction target, yielding 1.5 million training samples. Ground-truth localization is assumed rather than SLAM.

Models and Approaches

Map Prediction Model: A ResNet-based VAE is used to learn map completion. The encoder processes the 256×256 partial map (with the center masked), compresses it to a latent representation, and the decoder reconstructs the missing center patch. The loss function is a combination of binary cross-entropy for reconstruction and KL divergence for latent regularization.

Exploration Strategy:

- **Information Gain Estimation:** After predicting the masked region, a flood-fill algorithm is used to estimate the number of newly observable free cells.
- **Planner Integration:** The information gain is used as a third term in the cost function of two planners—Hector Planner and Cost-Utility Planner—alongside path distance and obstacle cost. This helps select the next best frontier.

Evaluation Protocol

Scope: MASKED — metrics computed exclusively over the masked 80×80 centre region; the surrounding observed context is excluded.
Approach: The training loss and evaluation metrics are applied only to the predicted (masked) patch: "The total loss of the prediction is the sum of all the pixel-wise losses for the masked 80×80 region." Observed pixels are not included in accuracy, precision, or recall computation.
Metrics: Accuracy, Precision, Recall (free-space and obstacle classes separately).

Novelties, Strengths, and Limitations

Novelties:

- Introduces predictive map completion into classical exploration planners using a VAE.
- Proposes an information gain-based cost augmentation technique using flood-fill over predicted regions.

Strengths:

- Improves frontier selection by estimating unseen space beyond sensor range.
- Achieves higher coverage efficiency and shorter paths compared to baseline planners.
- Uses a large, diverse dataset derived from real architectural plans.

Limitations:

- The VAE decoder does not receive explicit conditioning on the observed region; spatial context passes entirely through the latent bottleneck, which can degrade boundary coherence at the known/unknown interface.
- The dataset is constructed using pseudo-SLAM: circular regions are revealed without modeling occlusion, sensor noise, or robot motion—reducing realism.
- The prediction task is limited to *inpainting* the map center, assuming sufficient surrounding observations, which may not be valid near real-world frontiers.

Comparative Analysis

Both [15] and this work use map prediction to guide frontier-based exploration, but differ on dataset realism, model selection, and cost formulation. [15] generates training data by running Hector exploration in Stage simulator on KTH floor plans, recording frontier-centered 256×256 regions with an 80×80 masked centre—without SLAM noise or realistic sensor physics; training data here comes from full Gazebo/SLAM robot trajectories with realistic noise. The architecture deployed is a residual U-Net selected by masked-region evaluation (Chapter 3), not an unconditional VAE (namely, one in which observed context passes entirely through the latent bottleneck rather than being fed directly to the decoder), whose

bottleneck compression may degrade boundary coherence at the known/unknown interface.

On information gain: [15] applies flood-fill to the predicted map to count visible free cells—an exploration strategy (not a reconstruction metric) that implicitly favours large predicted open areas; obstacle awareness enters only at the costmap level via a collision penalty term, whereas our work incorporates explicit obstacle-aware frontier filtering prior to goal selection. Inference latency is comparable: the [15] VAE processes a batch of 32 frontiers in $\approx$5 ms on a GTX980, similar to the deployed U-Net. Finally, [15] is centred on inpainting the masked centre of a map with surrounding context available—an assumption that may not hold at real-world frontiers where the context is itself incomplete. This thesis focuses on outpainting: predicting genuinely unobserved regions beyond the current frontier.

2.8 Im2Pano3D: Extrapolating 360° Structure and Semantics Beyond the Field of View [28]

Tasks and Goal

This study addresses the task of *semantic-structure view extrapolation* from partial RGB-D images. The goal is to predict both the 3D structure and semantic segmentation for the full 360° panoramic view of an indoor scene, given only a limited observation (typically 50% or less of the scene). The task integrates both inpainting (filling internal holes) and outpainting (predicting unseen regions) to provide a globally consistent representation that could enhance high-level robotic reasoning, such as navigation or next-best-view planning.

Role of Map Prediction

Map prediction is central to this work. The authors define it as a semantic and geometric extrapolation of unobserved regions based on learned priors from indoor environments. Their model jointly predicts surface normals, planar geometry, and a probability distribution over semantic categories, thereby constructing a complete, interpretable world model. While the output format (RGB-D + semantics) differs from occupancy grids, the conceptual role of prediction—to hypothesize unobserved regions for intelligent planning—is highly relevant to robotic systems.

Dataset and Its Preparation

The model is trained on the SUNCG dataset and fine-tuned and evaluated on Matterport3D.

SUNCG: A large-scale synthetic dataset of indoor scenes rendered with RGB-D and semantic labels. The authors use 58,866 panoramic images for training and 480 for testing. The dataset enables pixel-accurate supervision and covers diverse layouts but lacks real-world noise and texture variation.

Matterport3D: A real-world dataset of RGB-D indoor scans. Instead of using the raw panoramas, the authors re-render panoramic RGB-D and semantic images from the 3D mesh using skybox-style projection. They use 5,315 rendered panoramas for fine-tuning and 480 for testing. Partial inputs are simulated by masking out large portions (e.g., 180° FOV) of the full panoramic image. Although this approach aligns with the synthetic training format, it does not model realistic occlusions or actual sensor coverage in robotic systems.

Models and Approaches

The proposed model, Im2Pano3D, is a multi-stream autoencoder that processes four input-output modalities: color (RGB), surface normals, plane distance, and semantic segmentation.

Architecture:

- **Encoder:** Separate convolutional streams for each modality (multi-stream), followed by shared latent layers.
- **Decoder:** Shared decoding layers, then split into separate decoding streams for each modality.
- **PN-layer:** A geometry module that computes 3D points from predicted normals and distances to enforce consistency during training. This layer is used only for supervision and is activated after 1000 iterations.

Loss Functions: The training objective combines pixel-wise reconstruction loss, Patch-GAN adversarial loss, and high-level scene attribute loss incorporating scene category classification and object distribution matching. This multi-level loss encourages both local accuracy and global consistency.

Evaluation Protocol

Scope: MASKED — evaluation restricted to unobserved regions only; the paper explicitly states "all evaluations are performed on unobserved regions only."
Approach: Only the predicted (extrapolated) pixels are compared against ground truth; the observed input portion is excluded from metric computation.
Metrics: Pixel-wise L1, GAN loss, scene attribute loss; structural and semantic quality metrics over predicted regions.

Novelties, Strengths, and Limitations

Novelties:

- Introduces the novel task of semantic-structure view extrapolation.
- Uses a multi-stream architecture tailored to modality-specific representation learning.
- Applies a differentiable PN-layer to supervise 3D geometry through consistency between normals and planes.

Strengths:

- Provides joint predictions of semantics and structure from a single RGB-D input.
- Shows strong generalization from synthetic (SUNCG) to real (Matterport3D) data.
- Enables context-aware 3D reasoning useful for robotic planning tasks.

Limitations:

- Uses synthetically masked regions rather than realistic occlusions or sensor-limited views.
- Predicts at category-level semantics, ignoring instance-level differentiation.
- Does not isolate the effects of inpainting vs. outpainting in the evaluation.
- Input and output formats (RGB-D + semantics) differ from the discrete occupancy grid used in robotic exploration.

Comparative Analysis

Im2Pano3D and our work share the conceptual goal of predicting unseen spatial structure from partial observations, but operate in entirely different domains. Their system predicts 3D semantic panoramas from RGB-D input using synthetic masking; ours predicts discrete 2D occupancy values from SLAM-generated grids. Multiple generative model families are evaluated in this thesis (CVAE, GAN, ViT, diffusion, U-Net variants), with the best-performing model—the residual U-Net—validated through its downstream effect on live exploration decisions rather than through reconstruction metrics alone.

2.9 Deep 3D Semantic Scene Extrapolation [29]

Tasks and Goal

The primary task addressed in [29] is *scene extrapolation*, which focuses on predicting the missing half of a 3D scene when only one half is observed. Unlike conventional 3D scene completion methods that fill in occluded regions within the field of view, the goal here is to extrapolate into completely unobserved areas beyond the current observation boundary. This type of extrapolation is particularly relevant in domains such as robot navigation and exploration, where anticipating the layout of unseen regions could improve decision-making and efficiency.

Role of Map Prediction

Map prediction is central to the extrapolation task presented in this study. Given a voxelized 3D representation of the visible half of a scene, the models are trained to generate the semantic layout of the other half. This ability to predict unobserved regions aims to provide structurally and semantically consistent completions that could theoretically be used to improve robotic mapping, navigation, and path planning if integrated into a full system.

Dataset and Its Preparation

SUNCG is used (see Table 2.5). Scenes are voxelized with `binvox` at $6\,\mathrm{cm}^3$ resolution. After filtering for room size and object diversity, approximately 211,000 scenes are retained at $84 \times 42 \times 84$ voxels. 200,000 scenes are used for training. Thirteen semantic classes plus background are retained.

Models and Approaches

CNN-SE: The base model is a fully convolutional network (FCN) with no pooling or fully connected layers. It treats the 3D voxel grid as a 2D input with 42 channels (each corresponding to one voxel plane), applying a 7×7 convolution followed by two sets of residual blocks. The model ends with a 1×1 convolution and a Softmax classifier for voxel-wise prediction. A combination of cross-entropy loss, smoothness loss, L2 regularization, and focal loss is used during training.

Hybrid-SE: This model augments CNN-SE by including a top-view projection of the visible half, which is passed to an autoregressive PixelCNN. The predicted top-view of the unseen region is lifted to 3D using a separate 2D-to-3D network. The outputs of CNN-SE and the 2D-to-3D model are fused and passed through a ResNeXt-based smoother network for refinement. Each subcomponent is trained separately.

U-Net: A U-Net variant is also evaluated for comparison, using similar inputs and cross-entropy-based losses.

Evaluation Protocol

Scope: MASKED — metrics computed solely over the unobserved (hidden) half of the voxel scene.
Approach: The task is a clean half/half split: the observed half is input and the hidden half is the prediction target. Evaluation covers the predicted voxels only.
Metrics: Voxel-level Accuracy (correctly predicted voxels / total predicted) and Completeness (correctly predicted occupied voxels / total occupied ground-truth voxels).

Novelties, Strengths, and Limitations

Novelties:

- Introduces the task of scene extrapolation beyond occluded regions, distinct from conventional scene completion.
- Uses a hybrid architecture combining 2D projections and 3D voxel models.
- Demonstrates that relatively simple convolutional networks can generate plausible 3D completions from partial inputs.

Strengths:

- Achieves realistic extrapolation results using lightweight convolutional models.
- Incorporates multiple perspectives (3D and top-down 2D) for enhanced predictions.
- Highlights the potential of PixelCNN with cross-entropy and L2 loss as a viable alternative to adversarial training.

Limitations:

- Limited ability to hallucinate or infer completely unseen object types in the extrapolated region.
- No integration with downstream tasks such as robot navigation, planning, or real-time decision-making.
- Evaluation is restricted to noise-free synthetic datasets (SUNCG voxelized scenes) with no validation in real-world or embodied robotic systems.
- The multi-component architecture (CNN-SE + PixelCNN + 2D-to-3D lifter + smoother) introduces substantial computational overhead, making real-time deployment unlikely.

Comparative Analysis

[29] defines 3D semantic scene extrapolation but evaluates solely on voxel-based synthetic scenes with no robotic integration. The present work differs in three key respects. First, the target representation is 2D occupancy classes (free/occupied/unknown) on SLAM-generated grids rather than 3D semantic voxels. Second, the model families evaluated here (CVAE, CGAN, ViT, U-Net variants, etc.) predict from partially observed 2D maps; architectures with explicit conditioning (CVAE, GAN, RePaint) structurally couple the known region to the output, while feed-forward variants (U-Net, ViT, LSTM, ImageGPT) process the partial map through standard encoder-decoder pipelines. Third—and most importantly—predictions are validated through downstream exploration performance in a live ROS loop, whereas [29] identifies such integration only as future work. Structurally, the half-visible/half-extrapolated setup of [29] bears a close resemblance to the upper-to-lower local completion formulation introduced in this thesis (Section 3.4), where the upper half of a 32×32 patch serves as the conditioning context and the lower half is generated as output. Both approaches

exploit a clean directional boundary between the known and unknown regions, which provides a well-defined conditional prior but also constrains applicability: real robotic deployment exposes irregular, frontier-shaped unknown regions that such structured splits cannot represent. Furthermore, the absence of downstream validation in [29]—acknowledged by the authors as future work—combined with the substantial computational overhead of the multi-component pipeline (CNN-SE + PixelCNN + 2D-to-3D lifter + smoother), renders the approach more suitable for offline analysis than for online integration into an active exploration loop.

2.10 Extreme Relative Pose Estimation for RGB-D Scans via Scene Completion [26]

Tasks and Goal

The main task addressed in [26] is **relative pose estimation** between two RGB-D scans of the same environment, even when the scans are minimally overlapping or entirely non-overlapping. The goal is to infer the relative rigid transformation (rotation and translation) that aligns the two scans into a common coordinate frame. To achieve this, the authors propose a system that combines scene extrapolation with robust pose estimation, enabling alignment even under extreme viewpoint changes. This is particularly relevant for applications such as SLAM, loop closure detection, and 3D reconstruction.

Role of Map Prediction

Map prediction is treated as an intermediate but crucial step for enabling accurate relative pose estimation. A recurrent *scene completion module* is used to predict missing regions in each scan by incorporating information from both scans iteratively. These predicted cube-map representations include color, depth, normal vectors, semantic segmentation, and learned feature descriptors. The completed scans provide a richer representation for identifying correspondences and estimating the transformation between the two views.

Dataset and Its Preparation

The system is trained and evaluated on three RGB-D datasets: **SUNCG** (synthetic), **Matterport3D**, and **ScanNet** (real-world). In SUNCG, 9,892 bedroom scenes are used, with 80% for training and 20% for testing. For each scene, 25 camera views are rendered with a limited 90° field of view, generating partial RGB-D observations and their ground-truth completions. In Matterport3D, 925 real indoor scenes are

used with the same sampling strategy and official train/test split. In ScanNet, 1,513 real environments are sampled using every 25th frame from the recorded sequence, but due to incomplete reconstructions, the resulting cube-maps contain more missing regions. For evaluation, 1,000 scan pairs are selected from each dataset. During training, both scan completions and relative poses are supervised using ground truth.

Models and Approaches

Map Prediction Model: A U-Net-based completion module is employed. It takes as input the RGB-D scan and a transformed version of the second scan and outputs a multi-channel cube-map encoding color, depth, normal, semantic labels, and learned descriptors. The inputs are passed through separate convolutional branches for each modality (color, depth, normal), which are then concatenated and fed into the encoder-decoder network. A binary mask indicates observed vs. missing regions.

Pose Estimation Model: A hybrid matching module combines *spectral matching* and *robust fitting*:

- **Spectral matching** identifies mutually consistent feature correspondences based on pairwise geometric constraints.
- **Robust fitting** (via Iteratively Reweighted Least Squares) estimates the rigid transformation by down-weighting outliers.

These two modules alternate over multiple iterations, refining each other's outputs. This recurrent design enables the system to better infer a coordinate frame that is valid for both scans.

Evaluation Protocol

Map completion quality is not the primary evaluation target in this work. Scene completion serves as an intermediate step for pose estimation; no standalone map completion metric is reported.

Primary metrics: Rotation error (degrees), translation error; F-score for point-cloud alignment.

Note: Completion quality is assessed implicitly through pose estimation accuracy, not through IoU, SSIM, or equivalent reconstruction metrics.

Novelties, Strengths, and Limitations

Novelties:

- A recurrent mutual refinement framework where scene completion and pose estimation improve each other.
- A novel representation combining geometry and semantics for matching.
- Integration of spectral matching and robust fitting in a unified optimization framework for relative pose estimation.

Strengths:

- Handles extreme cases with little or no overlap, where traditional methods fail.
- Outperforms state-of-the-art baselines on SUNCG, Matterport3D, and ScanNet.
- Modular structure allows potential substitution of components, e.g., using GANs for extrapolation.

Limitations:

- Computationally expensive due to the recurrent structure and alternating modules.
- Not directly applicable to 2D top-down occupancy maps without modification.
- Focused on RGB-D inputs and relative pose, not map-based planning or decision-making.

Comparative Analysis

While [26] uses scene extrapolation to refine relative pose estimation between RGB-D scans, the present work uses it to predict unobserved occupancy and guide autonomous exploration. Their completion module operates on cube-map representations combining colour, depth, normals, and semantics; ours processes top-down 2D occupancy patches (free/occupied/unknown). Their output is a rigid transformation aligning two 3D scans; ours is a per-frontier free-space gain score

derived from completed 32×32 patches, used solely to augment the frontier cost function without modifying the underlying SLAM map. Their system is a pose-estimation backend with no interface to planning; ours directly drives goal selection in a ROS exploration loop. Both demonstrate the utility of scene extrapolation, but in fundamentally different domains.

2.11 Emergence of Exploratory Look-Around Behaviors through Active Observation Completion [25]

Tasks and Goal

The work in [25] focuses on learning intelligent exploration strategies for visual agents that operate in novel environments. The core task is *active observation completion*, where an agent sequentially selects viewpoints to observe the environment efficiently and reconstruct the unobserved regions. The goal is to maximize reconstruction quality while minimizing the number of observations. Although navigation is not directly handled in this work, the authors suggest that their approach could be extended to target-driven navigation or related tasks. The model is designed to be adaptable to various visual perception tasks (e.g., categorization, pose estimation, surface estimation) using the same look-around policy.

Role of Map Prediction

Map prediction (viewgrid reconstruction) is central to this approach. The system uses the predicted full viewgrid to reduce uncertainty and select the next most informative action. The better the model is at predicting unseen views, the more accurate its internal representation of the environment becomes, which in turn improves policy learning. Map completion is also used as a training signal—rewarding the agent based on the quality of its predictions to learn a reinforcement-based look-around policy that generalizes to unseen environments and new tasks.

Dataset and Its Preparation

Two datasets are used (see Table 2.5).

- **SUN360**: RGB images of size $32 \times 32 \times 3$ extracted from discrete viewpoints (4 elevations $\times$ 8 azimuths) within each panorama; 6,174/1,013/1,805 train/val/test scenes.

- **ModelNet**: Rendered as grayscale views ($32 \times 32 \times 1$) at 6 elevations $\times$ 10 azimuths per object. ModelNet-40: 6,085/327/1,310 train/val/test; ModelNet-10: 3,991/181/727.

During training, the agent sees full viewgrids and learns to predict the missing parts using only a few glimpses. At test time, the model must complete the scene from limited observations.

Models and Approaches

Map Prediction Model: A recurrent neural network consisting of five modules is proposed:

- **SENSE**: processes input images and camera metadata.
- **FUSE**: fuses visual and pose data.
- **AGGREGATE**: an LSTM that maintains a belief over time.
- **DECODE**: generates the predicted full viewgrid.
- **ACT**: a policy network $\pi(a_t|s_t)$ trained via reinforcement learning.

The agent is rewarded for minimizing the pixel-wise MSE between predicted and true viewgrids. To address sparse rewards, two *sidekick* learning strategies are proposed:

- **Reward-based sidekick**: identifies individually informative views based on one-view reconstruction error.
- **Demonstration-based sidekick**: generates trajectories of views that maximize overall information coverage.

Evaluation Protocol

Scope: FULL — metrics over the complete predicted viewgrid (all viewpoints, observed and unobserved).
Approach: Tip A (raw reconstruction) — the recurrent model outputs the full viewgrid; all predicted viewpoints contribute to metric computation. Note: this work operates on RGB viewgrids rather than 2D occupancy grids; direct comparison with occupancy map completion results is not appropriate.
Metrics: Pixel-wise MSE between predicted and ground-truth viewgrids; downstream task accuracy for scene categorisation and pose estimation.

Novelties, Strengths, and Limitations

Novelties:

- Introduction of the active observation completion task with unsupervised training.
- Sidekick policy learning using full observability during training.
- Transferable look-around policy applicable to various visual tasks without retraining.

Strengths:

- General-purpose policy adaptable to unseen environments and new tasks.
- Efficient use of data: prediction helps reduce required observations.
- Strong quantitative results in reducing reconstruction error and generalization to tasks like categorization, pose estimation.

Limitations:

- Outputs only 2D viewgrids — does not produce 3D geometric structures like point clouds or meshes.
- Uses discrete camera rotations; does not handle translations.
- Trained and tested in idealized settings with rendered scenes.

Comparative Analysis

Both the current research and the study in [25] aim to reconstruct unobserved environmental segments to facilitate intelligent exploration. However, the methodology presented in this thesis deploys a residual U-Net as the completion model, selected from a comparative evaluation of generative and convolutional families (CVAE, ImageGPT, GAN, ViT, diffusion, LSTM), to predict unknown regions within occupancy grids, whereas [25] utilises an LSTM-based recurrent model for 2D viewgrid prediction. While their approach involves training an agent via reinforcement learning to derive an exploration policy, the current framework avoids this complexity by integrating spatial predictions into a heuristic cost function. This function ranks frontier candidates within a standard frontier-based

planner, shifting the focus from direct action selection to the identification of the most informative exploration targets, while ROS navigation modules manage the underlying motion planning. Consequently, this modular design ensures high compatibility with the ROS ecosystem and greater practicality for real-world deployment.

Furthermore, this work operates directly on occupancy grid maps derived from realistic SLAM processes. It is worth noting that while [25] requires full viewgrids during *training* as supervision signal, at test time it operates from partial observations only—as does our system. The key distinction lies in domain: our approach handles 2D occupancy grids with SLAM noise, whereas [25] operates on idealised rendered viewgrids with no sensor-induced artifacts.

2.12 Map-Predictive Motion Planning in Unknown Environments [16]

Tasks and Goal

The study presented in [16] addresses the task of motion planning in unknown environments. The goal is to enable a robot to safely and efficiently reach a given goal location by planning dynamically-feasible trajectories through unobserved regions. To achieve this, the system predicts the occupancy of the unknown parts of the map from partial observations and uses these predictions to plan full trajectories without relying on traditional frontier selection.

Role of Map Prediction

Map prediction is at the heart of the planning strategy. Instead of using frontier points as incremental sub-goals, the robot directly plans toward the final goal using occupancy probabilities predicted for the unknown space. The predicted values serve as soft constraints in the trajectory optimization process, encouraging paths through likely free regions while penalizing potentially occupied areas. This allows the planner to anticipate and adapt to unseen corners and obstacles.

Dataset and Its Preparation

A custom dataset of 75 procedurally generated maze-like environments (25 m$\times$25 m, 2.5 m-wide corridors) is used for training, with 30 held out for testing (see Table 2.5). From each map, 504 unoccupied points are sampled; a simulated 5 m LiDAR scan defines the observed context, and target occupancy values are taken

within a 7.5 m radius around frontiers.

Models and Approaches

Map Prediction Model: Map prediction is formulated as a conditional stochastic process modeled via a Conditional Neural Process (CNP). The CNP consists of an encoder, an aggregation module, and a decoder—each implemented as multilayer perceptrons (MLPs). The encoder transforms observation-location pairs into embedding vectors, which are averaged to create a global context representation. The decoder then outputs the predicted occupancy probability for each unobserved target point. The model is trained with negative log-likelihood loss to predict Bernoulli-distributed occupancy values.

Planning and Path Generation: A modified global planner constructs a full path to the goal using predicted occupancy. First, A* is applied to generate a reference path over a cost map, where unknown cells are weighted by predicted occupancy probabilities. This reference is then smoothed via convex optimization to produce a dynamically-feasible trajectory within a safe corridor (bubble-based tube). Finally, the speed profile is optimized for minimum traversal time, ensuring the robot stops before entering highly uncertain or unseen regions.

Evaluation Protocol

Map completion quality is not evaluated in isolation. The CNP map predictor is integrated into a trajectory planner; evaluation focuses on planning performance rather than reconstruction accuracy.

Primary metrics: Trajectory completion rate, planning speed ($\approx$2 Hz), path safety in maze environments.

Note: No pixel-level completion metric (IoU, SSIM, etc.) is reported; prediction quality is assessed implicitly through downstream planning success.

Novelties, Strengths, and Limitations

Novelties:

- Proposes a real-time, prediction-informed planning pipeline that avoids explicit frontier selection.
- Introduces a CNP-based map predictor that can handle variable input sizes and efficiently infer occupancy distributions.

Strengths:

- Achieves high planning speed ($\sim$2 Hz), suitable for real-time operation.
- Produces smooth and anticipatory trajectories, especially beneficial in tight maze-like environments.
- Outperforms naive frontier pursuit in both speed and safety.

Limitations:

- Focuses solely on maze environments with a single dominant path structure.
- Does not evaluate multiple frontiers in parallel, which may limit robustness in open or multi-path environments.
- Applicability to real-world, cluttered environments remains to be validated.

Comparative Analysis

[16] is tailored for motion planning in narrow maze-like corridors, whereas the present work targets generalised coverage-oriented frontier exploration across diverse indoor layouts. Both involve predicting unobserved regions, but [16] uses a CNP to estimate occupancy probabilities per point for trajectory optimisation, while this thesis selects a residual U-Net to score 32×32 frontier patches via predicted free-space gain within the `explore_lite` framework. [16] is validated only on procedural mazes with idealised sensor models; the present system operates on noisy SLAM-derived maps and supports generalisation to varied floor plan topologies.

2.13 Unifying Map and Landmark Based Representations for Visual Navigation [30]

Tasks and Goal

The study in [30] addresses the goal-driven visual navigation problem under partial observability using only a sparse set of RGB images and the known start and goal poses. The core objective is to enable an agent to reach a specified target pose in a novel environment by synthesizing internal spatial representations and using past visual memories to overcome uncertainty in actuation and environmental variation.

Role of Map Prediction

Map prediction is not used for explicit reconstruction but for generating an abstract spatial representation that supports path planning. The learned mapping module produces allocentric feature maps from sparse egocentric images by predicting what the robot should see from unseen locations, thereby allowing planning through partially unobserved areas. Synthesized features at waypoints provide landmark-like visual anchors to aid robust closed-loop execution. The predicted feature maps are therefore used indirectly to reason about spatial structure and guide action decisions.

Dataset and Its Preparation

The system is trained and evaluated using the Matterport3D and Stanford Large-Scale 3D Indoor Spaces (S3DIS) datasets, which contain photorealistic 3D reconstructions of real-world indoor environments. Sparse RGB images are sampled from various poses within simulated scenes to create training episodes. Each episode includes a small number of registered images and their corresponding poses, from which the agent learns to generate a spatial representation and plan a path. Training and evaluation use S3DIS (areas 1, 5, 6 for training, area 3 for validation, area 4 for testing). Exact per-split episode counts are not reported; the policy is trained for 60K iterations using DAGGER and evaluated over 8K episodes with varying numbers of reference images (5–40).

Models and Approaches

Map Prediction: A CNN-based encoder-decoder network processes each egocentric RGB image to produce an egocentric top-down feature map. This is transformed into an allocentric map using a differentiable spatial transformer based on pose metadata. Allocentric maps from different views are fused (via weighted averaging), and an upconvolutional network upsamples the result into a final spatial representation. Unlike explicit occupancy maps, the output is a dense feature map optimized for planning rather than direct visualization.

Path Planning: A convolutional value iteration network (VIN) is used to predict optimal action sequences on the abstract map. It learns to generate plans from start to goal based on map features, goal location, and value propagation over several iterations.

Path Execution: The system synthesizes feature expectations (path signatures) for each waypoint along the planned path using a fusion module that aggregates

reference image features and relative poses. A recurrent neural network (RNN) with soft attention over the path signature is trained to follow the plan under actuation noise. The controller is trained using the DAGGER (Dataset Aggregation) imitation learning algorithm, enabling the agent to recover from states not encountered during expert demonstrations.

Evaluation Protocol

This work does not perform explicit map completion evaluation. The predicted feature maps are internal representations for navigation and are not assessed against ground-truth occupancy grids.

Primary metrics: Navigation success rate, path efficiency, policy performance in simulated environments.

Note: No reconstruction metric is reported; the focus is entirely on navigation performance.

Novelties, Strengths, and Limitations

Novelties:

- A unified framework combining learned mapping, planning, feature synthesis, and path execution using visual memories.
- Use of path signatures with synthesized features at planned waypoints to enable robust closed-loop control under noisy actuation.
- Fully end-to-end differentiable architecture learned from sparse image observations without explicit metric map construction.

Strengths:

- Capable of planning in unseen environments using a small number of sparse images.
- Avoids explicit localization or global SLAM; instead relies on learned feature representations and visual cues.
- Demonstrates strong navigation performance with minimal supervision and noisy actuation.

Limitations:

- No explicit occupancy map or ground-truth reconstruction evaluation.
- Navigation-focused design; not suitable for standalone map completion tasks.
- Limited to simulated environments with ideal RGB inputs and known poses.

Comparative Analysis

While [30] focuses on visual navigation using sparse RGB inputs and synthesized memory representations to improve path execution, the framework presented in this dissertation centres on explicit **map completion** utilising 2D occupancy grids derived from SLAM. Multiple model families were evaluated offline—including CVAE, GAN variants, ViT, ImageGPT, and U-Net architectures—with the residual U-Net selected for deployment based on superior masked-region accuracy and real-time throughput. Models are trained on patches extracted from SLAM-generated occupancy grids; during exploration, predictions are integrated into a frontier-based planner to prioritise informative targets. Distinct from the methodology in [30], where map prediction is employed implicitly to support planning toward a fixed goal pose without autonomous map growth, the current method directly quantifies completion quality through a masked-region evaluation protocol and validates the selected model through system-level exploration experiments. Furthermore, the proposed system is ROS-compatible and SLAM-aware, rendering it more suitable for generalised robotic exploration tasks in realistic environments.

2.14 SSCNav: Confidence-Aware Semantic Scene Completion for Visual Semantic Navigation [23]

Tasks and Goal

The SSCNav framework addresses the task of **object-goal semantic navigation** in unknown indoor environments using egocentric RGB-D observations. The agent is expected to navigate toward an instance of a specified object category (e.g., toilet), without knowing its exact location. To achieve this, the system must simultaneously infer where the target object might be and plan an efficient collision-free path toward it. This makes the task substantially more challenging than traditional point-goal navigation, as it requires semantic reasoning about unseen areas in addition to spatial planning.

Role of Map Prediction

Map prediction plays a central role in the proposed navigation pipeline. A semantic scene completion module is used to extrapolate the environment's full semantic layout from a partial top-down map generated from past observations. This module predicts semantic labels in unobserved regions and outputs a corresponding confidence map that reflects the trustworthiness of each prediction. The predicted semantic map and the confidence map are then passed to a navigation policy, which produces a spatial action map to select the next movement. The map completion result directly affects navigation performance, and the inclusion of the confidence map is shown to significantly reduce the impact of misleading predictions.

Dataset and Its Preparation

The system is trained and evaluated using the **Matterport3D** dataset within the Habitat simulator. RGB-D images are collected as the agent explores, and semantic segmentation is obtained using ACNet trained on 209,200 RGB-D samples. From these, 52,300 ground-truth semantic top-down maps are created. Each is paired with four incomplete input variants by randomly removing observed views. During training, losses are computed only on unobserved regions that are known in the ground truth. For testing, 715 samples from unseen environments are used to evaluate scene completion, and 687 navigation episodes are executed to test the complete pipeline.

Models and Approaches

Semantic Scene Completion: The scene completion network is a fully convolutional neural network with one max-pooling layer, four downsampling residual blocks, and five upsampling residual blocks. It is trained with cross-entropy loss to predict semantic labels of unobserved areas.

Confidence Estimation: A second CNN branch predicts a confidence map trained in a self-supervised manner using pixel-wise mean squared error between predicted correctness and ground truth. This allows the system to filter unreliable predictions during navigation.

Navigation Policy: A reinforcement learning-based policy uses a Double DQN setup to output a dense spatial action map. Each pixel in this map represents a navigational direction. The agent selects the pixel with the maximum Q-value and moves accordingly. A stop checker module determines task completion based on proximity, visibility, and category match.

Evaluation Protocol

Scope: MASKED — IoU is computed exclusively over regions that are unobserved in the input top-down map but available in the ground-truth semantic map; observed cells are excluded.

Approach: The paper restricts scene completion evaluation to regions the model must actually infer, directly measuring prediction quality over unknown cells. This matches the MASKED criterion used as the primary evaluation in this thesis.

Metrics: IoU (%) per semantic class; Success rate, SPL for system-level navigation evaluation.

Novelties, Strengths, and Limitations

Novelties:

- Introduces a **confidence-aware** semantic scene completion module for improving navigation under partial observability.
- Uses spatial action maps aligned with the predicted semantic map, enabling more intuitive and spatially grounded decisions.

Strengths:

- Outperforms prior baselines such as SAVN and GOSE in semantic navigation tasks.
- Confidence maps act as trust filters that improve planning robustness.
- General performance improves when better semantic segmentation models are used.

Limitations:

- Scene completion is limited to a 6m × 6m **local region**, not integrated into a global map.
- Evaluation assumes access to accurate pose estimates; real-world integration may be more challenging.
- System focuses only on object-goal navigation, not general-purpose exploration or map coverage.

Comparative Analysis

SSCNav integrates semantic scene completion into an RL-based object-goal navigation system, predicting dense object-category maps in a local 6 m$\times$6 m region (comparable in scale to a 64×64 patch at 0.1 m/cell) and filtering predictions with a CNN-based confidence map. The present thesis differs in goal and representation: targets are 2D occupancy classes (free/occupied/unknown) for coverage-oriented frontier exploration, not semantic object categories for goal-directed navigation. Rather than a dynamic confidence map, a fixed penalty is applied to frontiers predicted to lie inside a wall, and a free-space gain reward to reachable ones—a conservative approach that preserves all frontiers as fallback targets and requires no semantic segmentation.

2.15 Seeing the Un-Scene: Learning Amodal Semantic Maps for Room Navigation [24]

Tasks and Goal

This work focuses on the task of **room navigation** in previously unseen indoor environments. The objective is to enable an agent to reach a specified room type (e.g., kitchen, bedroom) using only partial RGB observations, without requiring a full map of the environment. Instead of relying on brute-force reinforcement learning or traditional SLAM, the agent learns to reason about unobserved regions using predicted semantic maps that capture the layout and structure of rooms beyond its current field of view.

Role of Map Prediction

Map prediction plays a central role in the system's decision-making pipeline. The agent predicts **amodal semantic top-down maps**—meaning that the maps include both observed and extrapolated regions, even those beyond the robot's current view. These belief maps estimate the likely layout, location, and size of various room types based on visual cues and architectural priors. The predicted semantic map is then used to generate a plausible target point inside the desired room, which the agent attempts to reach using a point navigation controller.

Dataset and Its Preparation

The system is trained and evaluated using a subset of the **Matterport3D** dataset within the Habitat simulator. While Matterport3D provides 3D reconstructions

and per-room semantic annotations, it does not contain pre-defined trajectories or room-goal pairs. These are generated during preprocessing as follows:

- The agent is randomly spawned in a navigable location that does not belong to the target room.
- A **target room type** is selected, and a goal point is defined as the geodesically closest point within the target room.
- Shortest-path trajectories are computed between start and target, and sequences of RGB images and egocentric semantic map crops are collected along the way.
- Ground truth semantic maps are constructed using 3-class labels: (1) outside the house, (2) inside house but not the target room, and (3) inside the target room.

The dataset includes approximately 2.6 million training episodes across 32 houses, 200 validation episodes in 4 houses, and 500 test episodes across 10 unseen houses. During inference, no ground-truth semantic maps are provided; the model starts with a zero or random map and uses its previous predictions recursively.

Models and Approaches

Map Prediction: The map generation model is a sequence-to-sequence LSTM architecture that predicts egocentric semantic maps. It uses the current RGB frame, the previously predicted map, and the previous action as input. A ResNet50 encoder extracts features from the RGB image and semantic map, which are fed into a 2-layer LSTM. The LSTM output is decoded via transpose convolution layers into a 2D semantic map. During training, scheduled sampling is used to progressively rely on the model's own predictions.

Point Prediction: A multilayer perceptron (MLP) predicts a 2D target point inside the specified room based on the predicted semantic map, the RGB input, and a one-hot room-type embedding. The model is trained using mean squared error (MSE) with respect to the ground-truth target point.

Navigation Policy: A separate navigation policy is trained using PPO with Generalized Advantage Estimation (GAE). This controller receives as input the predicted target point, the previous action, and a depth-encoded input via a ResNeXt encoder. The policy outputs discrete navigation actions (e.g., move forward, turn left) and is fine-tuned on points predicted by the point prediction module.

Evaluation Protocol

Scope: FULL — metrics computed over the entire egocentric top-down semantic crop, including both observed and predicted regions.
Approach: Tip A (raw reconstruction) — the LSTM model outputs a full semantic map; all pixels contribute to mIoU and Pixel Accuracy computation.
Metrics: mIoU (%), Pixel Accuracy (%) for semantic class prediction; SPL and Coverage for system-level navigation evaluation.

Novelties, Strengths, and Limitations

Novelties:

- Introduces amodal semantic map prediction for room navigation, extending reasoning beyond the agent's current observations.
- Demonstrates a modular architecture with separate map generation, target prediction, and navigation components.
- Uses egocentric RGB input only, without explicit SLAM or geometric mapping.

Strengths:

- Achieves generalization to unseen environments using architectural and stylistic room priors.
- Improves over brute-force PPO-based learning by integrating learned priors via intermediate maps.
- Ablation studies show contributions of each module and validate LSTM-based map generation.

Limitations:

- Map prediction quality is a bottleneck; performance improves significantly when using ground-truth semantic maps.
- Requires accurate room segmentation annotations in the dataset, which may not be available in all domains.
- Designed for semantic goal-driven tasks, not for full environment exploration or coverage.

Comparative Analysis

[24] targets room navigation: it predicts amodal semantic top-down maps from RGB images to infer likely room types and generate goal targets, without building or completing an occupancy grid. The present thesis targets coverage-oriented autonomous exploration: it completes the occupancy grid (free/occupied/unknown) from SLAM-derived patches and uses those predictions to score frontiers, not to reach a named goal. Key differences are (1) input modality (2D SLAM occupancy vs. RGB images), (2) prediction output (occupancy classes vs. semantic room layout), and (3) task (maximum coverage exploration vs. goal-directed room navigation). The modular architecture in [24] is, however, a useful precedent for future structure-aware extensions such as door-prediction-guided exploration.

2.16 SenseExpo: Efficient Autonomous Exploration with Prediction Information from Lightweight Neural Networks [11]

Tasks and Goal

The SenseExpo framework aims to improve autonomous robotic exploration efficiency in unknown environments. The primary task is to enable a robot—or a team of robots—to navigate and explore environments rapidly by predicting unobserved regions of the map and using these predictions to guide exploration. The goal is to minimize total exploration time and model complexity while maintaining high prediction accuracy and generalization capability. The system supports both single-robot and multi-robot settings and is integrated into a ROS-compatible pipeline for practical deployment.

Role of Map Prediction

Map prediction is central to the system. A lightweight deep neural network predicts local occupancy maps from partial observations. These predictions are incrementally fused to form a global predicted map, which is then used to extract frontiers and plan the next goal. Unlike traditional frontier-based exploration that relies on boundaries between known and unknown areas, SenseExpo uses the predicted free/obstacle boundaries—effectively assuming the predicted map is complete and reliable for decision-making. This enables faster coverage but depends heavily on the accuracy of the predicted map.

Dataset and Its Preparation

KTH is used for training: 24,000 samples for training and 6,000 for testing. Cross-domain robustness is evaluated on HouseExpo and MRPB 1.0 through both offline prediction metrics and full robot exploration simulations using the authors' own simulator (not Gazebo).

Models and Approaches

Map Prediction Model: The system uses a lightweight hybrid model combining a reduced U-Net, Transformer encoder blocks, and Fast Fourier Convolution (FFC). The U-Net captures local features, the Transformer models long-range dependencies, and the FFC component provides global structural context using spectral transformations. The model is trained using a combination of adversarial loss, ResNet50-based perceptual loss, and L1 loss. The input maps are preprocessed into three-channel representations rather than using raw trinary occupancy values.

Exploration System:

- **Frontier Extraction:** Frontiers are extracted from the global predicted map—not from observed data—by identifying boundaries between predicted free and obstacle regions.
- **Goal Selection:** A cost function based on the Euclidean distance between the robot and frontier candidates is used. Multi-robot maps are fused via Bayesian updating to enhance global map completeness.
- **Stopping Criterion:** Exploration is marked complete when no further frontiers are found in the predicted global map.

Evaluation Protocol

Scope: FULL — offline metrics over the entire predicted local patch; system-level evaluation via exploration simulation.

Approach: Tip A (raw reconstruction) — at test time, the model's full output is compared against the complete ground-truth image. Note that during training, adversarial and perceptual losses are applied only to uncertain regions, creating a mismatch between training emphasis and evaluation scope.

Metrics: PSNR, SSIM, LPIPS, FID for offline evaluation; exploration coverage and time for system-level evaluation.

Novelties, Strengths, and Limitations

Novelties:

- Introduces a highly lightweight map prediction model (as small as 709k parameters) integrating Transformer and FFC components.
- Uses a fully ROS-compatible, plug-and-play prediction node to guide exploration in real time.
- Proposes frontier extraction and goal selection directly on the predicted global map.

Strengths:

- Achieves significantly faster exploration compared to baseline and state-of-the-art methods (up to 13× improvement).
- Demonstrates strong generalization ability in cross-domain testing.
- Suitable for deployment on resource-constrained robots due to low parameter count.

Limitations:

- Fully trusts predicted maps—no uncertainty handling or fallback to sensor-based validation.
- Does not preserve the unknown class in outputs: by replacing all unknown cells with free or obstacle predictions, the system cannot distinguish between genuinely explored regions and regions whose status is only predicted, potentially masking areas that still need direct sensor observation.
- Evaluates prediction performance on static images rather than simulating full exploration in realistic environments like Gazebo.
- Multi-robot capability contributes to performance gains in their experiments, making it difficult to isolate the contribution of the map prediction model in single-robot settings.

Comparative Analysis

While SenseExpo focuses on accelerating exploration by utilising a fully predicted binary map, the framework presented in this thesis maintains a clear distinction between observed and unobserved regions, integrating map prediction through a more conservative methodology. In the SenseExpo approach, the completed map is used directly for frontier extraction without uncertainty handling or fallback to observed data. In contrast, the current system identifies frontiers based on the observed occupancy data and subsequently computes auxiliary scores from both the predicted free-space ratio and a fixed penalty for frontiers whose neighbourhood is predicted to lie inside a wall, to refine frontier selection within a conventional planner.

The architectures evaluated in this research include CVAE, CGAN, Vision Transformers, ImageGPT, and a residual U-Net (selected for deployment), all trained on 32×32 patches extracted from realistic, SLAM-based Gazebo environments. Distinct from SenseExpo, which transforms input data into abstract three-channel representations, this work operates directly on trinary occupancy values (free, occupied, unknown). Furthermore, whereas SenseExpo evaluates exploration in simulation using its own lightweight simulator rather than a full SLAM-based Gazebo pipeline, the present study simulates full autonomous navigation and exploration using ROS and Gazebo, thereby incorporating realistic sensor noise and mapping dynamics. Additionally, whereas SenseExpo relies on multi-robot fusion to enhance performance—making the single-robot contribution hard to isolate—the current approach uses single-robot exploration to ensure rigorous experimental control. Ultimately, the proposed system is engineered for robustness and operational reliability in real-world deployments, contrasting with the aggressive prediction-trust strategy favoured by SenseExpo.

2.17 MapEx: Indoor Structure Exploration with Probabilistic Information Gain from Global Map Predictions [12]

Tasks and Goal

The MapEx framework addresses the task of autonomous exploration in unknown indoor environments. The goal is to enable a robot to intelligently plan its next exploration steps by reasoning not only about what it has observed but also about what it might see next. Specifically, MapEx aims to improve exploration efficiency by calculating a probabilistic information gain at each candidate frontier. This is achieved by jointly considering predicted environment structure and the uncertainty associated with those predictions, guiding the robot toward regions that maximize

expected knowledge gain.

Role of Map Prediction

Map prediction is central to the MapEx framework. A global inpainting model is used to extrapolate unknown areas of the occupancy grid based on partial observations. The predicted maps serve two key roles: (1) to estimate which areas are likely visible from each frontier via probabilistic raycasting, and (2) to provide pixel-wise uncertainty by measuring variance across predictions from an ensemble of models. This variance map directly feeds into the information gain computation that scores each frontier. Thus, map prediction not only completes the visual structure but also drives exploration decision-making.

Dataset and Its Preparation

KTH floor plans are used (see Table 2.5), downsampled to 0.1 m/pixel. An 80:20 building-level split yields 2,367 training pairs (observed + ground-truth map, generated via nearest-frontier exploration) and 10 held-out floor plans. Robot simulations from 4 start positions per floor plan give 40 evaluation conditions.

Models and Approaches

Map Prediction Model: MapEx uses the LaMa (Large Mask Inpainting) network as its global map prediction model. This architecture is designed to handle large missing regions in images, leveraging Fourier convolutions and a wide receptive field for global context reasoning. To estimate prediction uncertainty, MapEx uses an ensemble of three independently fine-tuned LaMa models, each trained on a subset of the training data. The mean output across these models forms the predicted map, and the pixel-wise variance forms the uncertainty map.

Exploration Planner:

- **Frontier Extraction:** Frontiers are extracted from the observed map (not the predicted map) as the boundary between known and unknown space.
- **Probabilistic Raycasting:** For each frontier, visibility is estimated via raycasting over the predicted mean map, simulating 360° LiDAR beams up to 20 m range.
- **Information Gain Computation:** The information gain is computed by summing the uncertainty values within the visibility mask derived from

raycasting. This value is then normalized by Euclidean distance to prioritize nearby informative frontiers.

- **Goal Selection:** The frontier with the highest score is selected as the next goal, and an A* local planner generates a path to that location.

Evaluation Protocol

Scope: FULL — metrics over the entire predicted global occupancy map.
Approach: Tip B (blending/inpainting) — LaMa inpainting preserves known (observed) pixels and fills unknown regions only; observed pixels are trivially correct in the full-map metric, inflating Coverage and IoU scores relative to a pure reconstruction baseline.
Metrics: Coverage (%), IoU (% occupied class), Topological Understanding (TU) for system-level exploration evaluation.

Novelties, Strengths, and Limitations

Novelties:

- Introduces a global map prediction framework for exploration using LaMa.
- Uses an ensemble of models to compute a probabilistic uncertainty map.
- Incorporates probabilistic raycasting to better estimate sensor coverage.

Strengths:

- Realistically simulates LiDAR visibility during planning via probabilistic raycasting.
- Achieves significant performance improvements over previous map-prediction-based methods and classical frontier strategies.
- Demonstrates strong results using real-world building layouts from the KTH dataset.

Limitations:

- The training dataset is relatively small (2,367 examples), limiting the model's generalization potential.

- Each LaMa ensemble branch is trained on a subset of the data, reducing training data per model.
- The use of variance alone for information gain can undervalue accurately predicted but unobserved regions, and overvalue structurally unimportant high-variance areas.
- Map predictions are not directly used to extract new frontiers or perform SLAM.

Comparative Analysis

While MapEx is built on global map prediction using ensemble-based uncertainty from LaMa models, our work emphasises a broader comparative evaluation of model families—including CVAE, CGAN, ViT, ImageGPT, LaMa, diffusion, LSTM, and U-Net variants—selecting a lightweight residual U-Net for online deployment based on masked-region accuracy and real-time throughput. Rather than requiring multiple forward passes from an ensemble, the deployed U-Net performs a single inference call per frontier batch, reducing computational cost and enabling practical deployment on standard robot hardware.

Regarding information gain estimation: MapEx uses pixel-wise variance across ensemble outputs as a proxy for uncertainty. This has two known limitations, both listed in the MapEx paper itself: (1) accurately predicted but structurally uncertain regions (e.g., a reliably inferred corridor that the model consistently predicts) will show low variance and be undervalued, while (2) noisy or irrelevant high-variance areas unrelated to exploration utility may be over-prioritised. In contrast, the present system scores frontiers using the predicted free-space gain r_f directly—rewarding predicted open space—and applies a fixed cost penalty when the frontier neighbourhood is predicted to lie inside a wall, providing both a positive and a negative signal.

MapEx's limitation that "map predictions are not directly used to extract new frontiers or perform SLAM" means that frontiers are still discovered from the observed map only; the predicted map contributes to scoring but cannot reveal previously unknown frontier candidates. Our system shares this conservative design: frontiers are extracted from the SLAM-observed map, and predictions influence ranking only—a deliberate choice to avoid acting on potentially erroneous completions.

The training sets are not directly comparable in granularity—32×32 local patches

(9,000–10,000 per SDS/BDS configuration) vs. full-map pairs (2,367)—but the present work's data is derived from live SLAM runs in Gazebo rather than from static floor-plan vector files, providing richer noise and topology variety.

This thesis additionally reports a supplementary oracle navigation study (Chapter 5) examining the upper-bound benefit of perfect local completion integrated into the frontier-based pipeline. Though a direct comparison with MapEx is not appropriate, the study provides a reference point for future extensions toward full navigation-level integration.

It should also be noted that this thesis evaluates global map completion models operating on full 128×128 SLAM snapshots (Chapter 3); however, these were not selected for deployment due to throughput and modularity constraints in the live frontier-scoring pipeline, where per-frontier 32×32 patch inference proved more practical.

2.18 Occupancy Grid Artefact Removal and Error Correction using GANs [20]

Tasks and Goal

This study focuses on improving the quality of occupancy grid maps produced by SLAM systems by removing artifacts, noise, and structural errors. The main task is formulated as a map cleaning problem using image-to-image translation, where noisy occupancy maps are transformed into cleaner, more accurate representations. Although the paper occasionally refers to this task as "room completion" or "partial observation correction," the actual goal is not to infer or extrapolate unobserved regions, but to identify and eliminate artifacts introduced during SLAM mapping—such as ghost walls caused by LiDAR passing through windows or angular misalignments.

Role of Map Prediction

In this study, map prediction serves a denoising purpose: it refines already observed but distorted map regions by learning to map from noisy to clean domains. The models are trained to preserve true room structures while removing fake ones. If a room is only partially explored and genuinely incomplete, the model retains the visible portion without predicting the missing parts. Therefore, the prediction operates as a correction mechanism, not a generative extrapolation of unseen areas.

Dataset and Its Preparation

HouseExpo layouts are used via a modified PseudoSLAM simulator (see Table 2.5). A DRL agent explores each environment while noise is injected programmatically (sensor errors, angular drift, LiDAR leakage), producing 35,000 paired 256×256 grayscale images: a noisy map and its clean ground-truth reference.

Models and Approaches

CycleGAN: This architecture includes two generators (G, F) and two discriminators (D_X, D_Y). It employs adversarial losses to align noisy-to-clean and clean-to-noisy translations, and a cycle-consistency loss to ensure that round-trip translations preserve structure ($x \to G(x) \to F(G(x)) \approx x$). Generators consist of three downsampling convolutional layers, nine residual blocks, and two transposed convolutional layers for upsampling. PatchGAN discriminators evaluate 70×70 patches across the image.

CUT (Contrastive Unpaired Translation): CUT simplifies the framework by using a single generator (split into encoder and decoder) and a single discriminator. Instead of cycle-consistency, it uses **patch-wise contrastive loss** (PatchNCE), which aligns feature patches from input and output to preserve local structural information. This method avoids structural hallucinations, making it more suitable for layout-sensitive tasks like map cleaning.

Evaluation Protocol

Scope: FULL — metrics over the entire transformed (cleaned) occupancy map.
Approach: Tip A (full map transformation) — CycleGAN/CUT transforms the entire noisy input map into a clean output; all pixels of the generated map are compared to the ground truth. Note: this work performs SLAM artefact removal rather than extrapolation of genuinely unobserved regions.
Metrics: FID (distributional quality), Room Completion (segment-level MSE), Wall Placement Accuracy, Wall Offset.

Novelties, Strengths, and Limitations

Novelties:

- Novel dataset generation using DRL-based exploration in HouseExpo environments with configurable noise injection.
- Application of CycleGAN and CUT to occupancy grid artefact removal.

- Introduction of custom metrics for evaluating wall placement accuracy and segment-wise room consistency.

Strengths:

- Successfully removes artifacts and aligns misrepresented room structures.
- Uses well-established GAN frameworks and adapts them for a SLAM post-processing task.
- Demonstrates generalization to real-world Radish SLAM data despite training on synthetic maps.

Limitations:

- Does not perform real map completion—i.e., it does not predict or extrapolate genuinely unobserved regions.
- Only handles binary or grayscale maps, making direct integration with semantic or 3-class occupancy maps nontrivial.
- The models are not integrated into a real-time SLAM or exploration pipeline.

Comparative Analysis

While the study in [20] focuses on mitigating artifacts within fully observed occupancy maps using CycleGAN and CUT architectures, the research presented in this thesis addresses the more complex generative challenge of **scene extrapolation**—specifically, the prediction of unobserved regions within partially mapped environments. It is worth noting that the dataset in [20] is generated using a modified PseudoSLAM simulator in which a DRL agent navigates and noise (sensor errors, angular drift, LiDAR leakage) is injected programmatically; this produces controlled artifacts but does not involve a real SLAM stack or physics-based sensor simulation. The present work instead derives training data from a TurtleBot3 navigating Gazebo environments with GMapping SLAM, capturing noise that arises organically from sensor physics and map-building dynamics.

Distinct from the binary map correction performed in [20], occupancy grids are modelled here using discrete classes (free, occupied, unknown). Multiple architectural families—CVAE, CGAN, ViT, ImageGPT, diffusion, LaMa, LSTM, and residual U-Net—are trained to infer missing spatial data, with the U-Net

selected for deployment based on superior masked-region evaluation. The resulting predictions are integrated into a ROS-based frontier exploration framework, augmenting the frontier cost function with predicted free-space gain and a wall-proximity penalty to guide exploration proactively. While CycleGAN and CUT are general-purpose frameworks that could potentially be adapted for map completion, their current application is restricted to post-processing of previously observed data and does not address the extrapolation of genuinely unobserved regions.

From a system-level perspective, such methods may still be relevant in the later stages of exploration. As observed in our experiments, beyond the 95% coverage threshold, the remaining unknown regions tend to form geometrically isolated pockets that are deliberately deprioritized by the frontier-based cost function. In these cases, local refinement or planning-aware completion strategies similar to those in [20] could provide additional benefits.

2.19 Map Prediction and Generative Entropy for Multi-Agent Map Prediction [17]

Tasks and Goal

The study in [17] focuses on autonomous multi-agent exploration in outdoor urban environments. The primary task is to reduce the time needed to build a high-quality global occupancy map by efficiently assigning exploration targets to a team of robots. Instead of relying on classic information-theoretic strategies, the authors propose a novel task allocation method that prioritizes regions with high *generative entropy*—areas of high uncertainty in the predicted map—to guide robots toward parts of the environment that are harder to infer and thus more valuable to explore.

Role of Map Prediction

Map prediction is a central component of the framework. A global occupancy grid, fused from all agents' observations, is periodically input to an inpainting model that completes the unknown regions. These predictions are treated as noisy observations and used to incrementally update the posterior occupancy probability of each cell. The per-cell binary entropy is then calculated to form a *generative entropy map*. Regions with high entropy are prioritized during task allocation, under the hypothesis that resolving these uncertain areas will yield faster convergence to an accurate global map.

Dataset and Its Preparation

A custom procedural map generator was developed to create urban scenes with a central street and randomly placed buildings. Using a simulated robot platform, three agents explore each scene using potential-field navigation. As they move, simulated LiDAR sensors generate occupancy maps. A total of 2,000 unique environments and 180,000 exploration snapshots were created for training, while another 2,000 environments with 30,000 snapshots were reserved for testing. During training, partial observations were sampled at 10% coverage intervals; for testing, 5 coarser intervals were used. The dataset was designed to assess both local and global structure prediction capabilities under partial observability.

Models and Approaches

Map Prediction: Three state-of-the-art inpainting models were adapted for global occupancy map completion:

- **LaMa:** Uses Fast Fourier Convolutions (FFC) for large mask inpainting with strong structural priors but lacks semantic conditioning.
- **RePaint:** Employs a pixel-space diffusion model (DDPM) with resampling, robust to unseen mask geometries but computationally expensive (30s per image).
- **Stable Diffusion:** Operates in latent space, offering faster inference (1s per image) and the potential for semantic conditioning. Fine-tuned on occupancy grid data using a fixed instance prompt of "an occupancy grid" for 250,000 steps.

Task Allocation: The Asynchronous Consensus-Based Bundle Algorithm (ACBBA) is used to assign tasks to agents based on the total generative entropy within a bounding box around each task location. Unlike traditional frontier-based exploration, task points are quasirandomly distributed or derived from frontiers and ranked by generative entropy.

Evaluation Protocol

Scope: FULL — metrics over the entire predicted global occupancy map at coverage intervals (20%–80%).

Approach: Tip B (blending) — inpainting models (LaMa, RePaint, Stable Diffusion) fill unknown regions while observed pixels are preserved; FULL metrics

are partially inflated by trivially correct observed cells.
Metrics: LPIPS, FID evaluated at multiple coverage intervals; generative entropy for task allocation scoring.

Novelties, Strengths, and Limitations

Novelties:

- Introduces *generative entropy* as a new metric for task prioritization in multi-agent exploration.
- Leverages global map prediction using fine-tuned image inpainting models to inform exploration.

Strengths:

- Outperforms classic information-theoretic methods in terms of prediction convergence speed.
- Efficiently supports multiple agents with centralized or decentralized deployment.
- Prediction quality improves monotonically with map coverage, as quantified by LPIPS and FID metrics at 20%–80% observation intervals; the generative entropy objective directs agents toward high-uncertainty regions to accelerate this convergence.
- Map prediction is used only for task scoring, not for navigation; therefore, hallucinations or mispredictions have no impact on robot safety or path execution.

Limitations:

- Evaluation is limited to static outdoor simulations; no dynamic elements or sensor noise.
- The approach lacks integration with classical frontier-based systems and is not tested on real robots.

Comparative Analysis

While the study in [17] focuses on multi-agent exploration within outdoor environments using entropy-based task scoring, this thesis targets single-robot exploration in structured indoor settings utilising 2D occupancy maps generated via realistic SLAM systems. Although both studies employ learning-based models for map completion, their objectives diverge: the model in [17] performs global prediction to inform high-level task allocation, whereas the current approach utilises localised prediction to enhance frontier selection and decision-making. Distinct from the methodology in [17], map prediction is integrated here into a ROS-compatible frontier-based exploration stack by augmenting the standard distance-size cost function with two prediction-driven terms: a free-space gain reward for reachable frontiers and a fixed cost penalty for frontiers predicted to lie inside a wall.

The model families evaluated—CVAE, CGAN, ViT, ImageGPT, LaMa, diffusion, LSTM, and residual U-Net variants—are trained on a custom dataset of 32×32 map patches extracted from Gazebo-based exploration trials, with the residual U-Net selected for online deployment. The proposed framework handles both inpainting and outpainting by conditioning predictions on partial observations, thereby enabling proactive, risk-aware planning. Additionally, the approach accounts for realistic noise and localisation uncertainty, maintaining suitability for direct deployment on physical robotic platforms. In contrast, [17] relies on idealised fusion and global prediction without integration into local planning or SLAM feedback loops. Consequently, the research presented in this thesis offers a more granular, modular, and deployable system tailored for the challenges of real-world autonomous indoor exploration.

2.20 Geometry-Based Completion via Structural Inference [21]

Tasks and Goal

The primary task addressed in [21] is to complete partially observed occupancy grid maps by inferring the layout of unobserved rooms hidden behind closed doors. The goal is to improve the spatial representation of indoor environments to support downstream robotic tasks such as offline path planning or proactive exploration. This thesis also includes a geometry-based baseline [9] developed and published as part of this research, which pursues the same objective through a simplified door-detection and region-expansion strategy. That method is described in detail in Chapter 3 (Section 3.3) and Chapter 7; it is referenced here solely to contextualise how the thesis's own geometric contribution relates to the external literature.

Role of Map Prediction

In [21], map prediction serves as the core mechanism to recover missing portions of the map that the robot could not access during SLAM-based mapping due to closed doors or occlusions. Map prediction enables the system to reason about the topology of unseen rooms and improves planning by allowing pre-computation of complete traversal paths across rooms, including those initially inaccessible.

Dataset and Its Preparation

Experiments use real-world maps from the Radish repository (e.g., Intel Lab) and custom Stage simulator environments (see Table 2.5). Up to 15 closed doors are randomly introduced in simulation, generating 3,885 partial maps corresponding to 24,045 predicted room shapes. No training is required.

Models and Approaches

Map Prediction in [21]: The method begins with edge detection using the Canny method and line extraction via the probabilistic Hough transform. Wall segments are then clustered using a two-stage process: (1) orientation clustering via mean-shift and (2) spatial clustering of collinear lines. Representative wall lines are generated from each cluster and used to partition the map into convex polygonal facets. These facets represent candidate room regions. The map is completed by expanding room facets from initial seed regions located behind known door positions. A cost function—considering area, convexity, boundary continuity, and regularity—is used to iteratively select adjacent facets for expansion. A final face-swapping optimization is performed for dependent adjacent rooms to improve shape consistency.

The geometry-based baseline developed within this thesis [9] simplifies the [21] pipeline by removing mean-shift wall segment clustering and polygonal partitioning, replacing them with threshold-based coordinate deduplication and rectangular region filling. It uses standard Hough transform to extract vertical and horizontal lines, and applies threshold-based heuristics to detect vertical and horizontal doors by scanning pixel regions. Map completion is performed by expanding rectangular regions from each detected door, bounded by the nearest wall or guide line in each direction. Two variants are introduced: Algorithm 1 relies on proximity to structural lines, while Algorithm 2 uses bounding box limits as fallback guides. Full details and quantitative evaluation of both variants are provided in Chapters 3 and 7.

Evaluation Protocol

Scope: MASKED — IoU computed solely between the predicted room layout $\hat{F}_{r^*}$ and its ground-truth counterpart F_{r^*}; the already-observed portions of the map are excluded from metric computation. **Approach:** Only predicted (completed) room regions are evaluated; no blending effect is present since observed cells are never included in the IoU denominator.

Novelties, Strengths, and Limitations

[21]:

- **Novelties:** Introduces a fully geometric, face-based map completion method using structural wall partitioning. Capable of handling multiple adjacent closed rooms through a joint optimization mechanism.
- **Strengths:** Handles complex room layouts without training; produces interpretable results; integrates into planning pipelines.
- **Limitations:** Requires accurate closed-door annotations; cited door detection methods rely on external RGB/RGB-D sensors and deep models, adding hidden cost; not real-time; not integrated with SLAM or online decision-making.

The geometry-based method published as part of this thesis [9] shares similar strengths (training-free, lightweight) and limitations (threshold-sensitive, assumes Manhattan layouts, no adaptive reasoning). Its detailed evaluation is in Chapters 3 and 7.

Comparative Analysis

[21] relies on structural cues in occupancy maps to infer missing regions through rule-based geometric reasoning. The thesis builds on this principle by including its own published geometric baseline [9] as a training-free reference method, but then advances substantially by employing learning-based models—residual U-Net, CVAE, GAN variants, ViT, diffusion, LaMa, and LSTM—that learn to complete maps from data. These models are trained offline on thousands of 32×32 frontier-centered occupancy patches derived from realistic SLAM-based exploration in Gazebo (approximately 9,000–10,000 training patches for each of the SDS and BDS configurations).

Unlike [21], the thesis integrates predictive completion directly into a frontier-based exploration pipeline. The U-Net, selected as the deployment model, scores frontiers based on predicted free space and penalises frontiers predicted to lie inside a wall, enhancing decision-making during exploration. The models generalise to irregular frontier shapes, are robust to SLAM noise, and require no predefined door annotations. Additionally, the approach is ROS-compatible, enabling deployment with standard planners such as A*, TEB, or DWA.

In summary, [21] offers a sophisticated geometry-based method, and the thesis's own geometric contribution provides a lightweight alternative; both are limited by their offline nature and reliance on strict structural assumptions. The learning-based system presented in this thesis overcomes these limitations by combining learned spatial priors with real-time planning and system integration.

Summary of Evaluation Protocols Across Reviewed Studies

Table 2.6 summarises the evaluation protocols used across all reviewed studies. The majority report metrics over the full output patch (FULL), either via raw model reconstruction (Tip A) or blending of known and predicted pixels (Tip B). A minority restrict evaluation to originally unknown cells (MASKED), which more directly measures genuine prediction quality. Among FULL-protocol studies, Tip B approaches (MCN-SLAM, UPEN, MapEx, Spinos et al.) inflate scores by including trivially correct observed pixels, while Tip A studies (OccAnt, CM2, SenseExpo, Davies et al.) require the model to reconstruct all pixels including observed ones. The present work adopts MASKED as the primary criterion for frontier-centered evaluation, consistent with [13] and [15] (though in the latter the evaluation region is not stated explicitly, the loss formulation over the masked 80×80 patch implies MASKED evaluation), and extends it to irregular frontier-shaped masks derived from live SLAM exploration — a structural distinction from the regular geometric masks used in both prior studies.

Recent advancements in map prediction and scene extrapolation have enhanced the ability of robotic systems to operate in environments with partial observability. While many studies emphasise semantic reasoning, and most—including the present thesis—conduct primary evaluations in simulation, only a limited subset demonstrate full integration into exploration pipelines with system-level validation.

In contrast, our work addresses this gap by introducing a modular, ROS-compatible framework that harnesses learning-based models for occupancy map completion—selecting a residual U-Net for deployment based on rigorous

Table 2.6 Evaluation protocols in reviewed studies

Study	Protocol	Approach	Primary Metrics
[13]	MASKED	—	SSIM, PSNR
[15]	MASKED	—	Accuracy, Precision, Recall
[29]	MASKED	—	Accuracy, Completeness
[28]	MASKED	—	L1, scene attribute loss
[23]	MASKED	—	IoU per semantic class
[14]	FULL	Tip A	SSIM
[22]	FULL	Tip A	IoU, F1, Map Acc.
[24]	FULL	Tip A	mIoU, Pixel Acc.
[27]	FULL	Tip A	IoU, F1, PCW
[20]	FULL	Tip A	FID, Room Comp., Wall Acc.
[11]	FULL	Tip A	PSNR, SSIM, LPIPS, FID
[19]	FULL	Tip B	Accuracy, F-measure
[18]	FULL	Tip B	Map Acc., IoU, Coverage
[12]	FULL	Tip B	Coverage, IoU, TU
[17]	FULL	Tip B	LPIPS, FID
[25]	FULL*	Tip A	MSE
[26]	N/A	—	Pose error
[16]	N/A	—	Planning metrics
[30]	N/A	—	Navigation metrics
[21]	MASKED	—	IoU, Coverage

* Different domain: RGB viewgrid, not 2D occupancy map.
Type A: Raw reconstruction — model predicts all pixels.
Type B: Blending — observed pixels preserved, only unknown cells filled.

masked-region evaluation—and actively guides frontier-based exploration. Compared to prior approaches, our method prioritises lightweight inference, spatial realism, and practical deployment in static simulated indoor environments.

This literature review provides the architectural foundation for the proposed system by categorizing prior research based on the functional role of map prediction and the underlying exploration strategies, as detailed in Tables 2.2, 2.3, and 2.7. Building upon this comparative analysis, a dedicated subsection is provided to justify the selection of the `explore_lite` [6] framework. This rationale demonstrates how the current approach diverges from existing global or uncertainty-aware methodologies by facilitating localized, patch-based integration with minimal computational overhead.

Furthermore, a significant gap in the existing literature regarding evaluation integrity is addressed; many current studies lack a rigorous protocol to prevent data leakage between structurally similar floorplans. The methodology presented here is contrasted with modern frontier strategies, such as *nearest-frontier* or *utility-based* selection, to demonstrate that incorporating a predictive 'free-space gain' into traditional cost-utility functions provides a more forward-looking exploration strategy.

Ultimately, this review not only guided our system design choices but also revealed critical directions for future research, particularly in predictive spatial reasoning, real-time exploration, and decision-making under uncertainty.

It is worth noting that the present system already computes a blended output — merging the model's predictions for unknown cells with the original observed pixels — as a visualisation aid during qualitative evaluation (Figures 7.9 and 7.10). Extending this blended result to quantitative FULL evaluation would require only a minor post-processing step and would place all learned models on equal footing with GEOM and RePaint, which implicitly benefit from trivially correct observed pixels in the current reconstruction-based FULL protocol. This is identified as a direction for future evaluation refinement; it would not affect MASKED metrics or any conclusion drawn from them.

Rationale for Selecting `explore_lite` as the Exploration Backbone

In this thesis, the ROS-native `explore_lite` framework is adopted as the foundation of the exploration system due to its modularity, computational efficiency, and seamless compatibility with real-world robotics platforms. The map prediction module developed herein operates on localized 32×32 patches extracted from

SLAM-generated occupancy grids, rendering high-resolution, global prediction frameworks impractical for this specific use case. While several state-of-the-art predictive exploration systems rely on large-scale global map completion and dense uncertainty modeling—such as UPEN [41], MapEx [12], and the work by Spinos et al. [17]—these architectures remain tightly coupled to their specific prediction engines and lack support for modular integration of external predictors at patch-level granularity.

SenseExpo [11], while using local predictions and offering greater modularity, does not cleanly separate frontier detection from map prediction: it extracts frontiers directly from the predicted map rather than the observed SLAM map, coupling frontier generation to prediction quality in a way that prevents using predictions solely as a scoring signal. IG-Hector [15], by contrast, detects frontiers from the observed map and uses map predictions only for information gain computation—a design closer in spirit to the present work. However, it injects this gain into planner-internal cost transforms across multiple pipeline stages, preventing clean separation of the scoring layer from the path planner.

`explore_lite` differs by exposing frontier scoring as a separable stage over the observed SLAM map, allowing the UNet batch service to augment frontier costs without modifying the path planner or navigation stack.

MapEx and Spinos et al. use global predictions requiring high GPU memory and end-to-end global reasoning. UPEN relies on ensemble-based epistemic uncertainty, introducing overhead unsuitable for lightweight patch-based inference. IG-Hector injects VAE-based information gain into planner internals across multiple pipeline stages.

Unlike [15], where predicted maps feed into both frontier scoring and planner-level cost computation across multiple pipeline stages, the present system confines all prediction-based modifications to a single layer: frontier ranking within `explore_lite`. The path planner and navigation stack remain entirely unmodified, reducing integration complexity and making the completion module trivially replaceable without affecting downstream components.

Recent advancements in map prediction and scene extrapolation have enhanced the ability of robotic systems to operate in environments with partial observability. While many studies emphasise semantic reasoning, and most—including the present thesis—conduct primary evaluations in simulation, only a limited subset demonstrate full integration into exploration pipelines with system-level validation.

Table 2.7 Comparison of exploration systems for integration with patch-based map prediction

System	Modular	ROS	Local Pred.	Patch-Based	Global Pred.	Real Robot
Ours [6]†	Yes	Yes	Yes	Yes	No	No
[41]	No	No	No	No	Yes	No
[15]	Partial	Yes	Yes	No	No	No
[12]	No	No	No	No	Yes	No
[11]	Yes	Yes	Yes	No	No	No
[17]	No	No	No	No	Yes	No

† Extended `explore_lite`: patch-based UNet completion integrated as a frontier-scoring layer; prediction is contributed by the present work, not by `explore_lite` itself.

In contrast, our work addresses this gap by introducing a modular, ROS-compatible framework that harnesses learning-based models for occupancy map completion—selecting a residual U-Net for deployment based on rigorous masked-region evaluation—and actively guides frontier-based exploration. Compared to prior approaches, our method prioritises lightweight inference, spatial realism, and practical deployment in static simulated indoor environments.

This literature review provides the architectural foundation for the proposed system by categorizing prior research based on the functional role of map prediction and the underlying exploration strategies, as detailed in Tables 2.2, 2.3, and 2.7. Building upon this comparative analysis, a dedicated subsection is provided to justify the selection of the `explore_lite` [6] framework. This rationale demonstrates how the current approach diverges from existing global or uncertainty-aware methodologies by facilitating localized, patch-based integration with minimal computational overhead.

Furthermore, a significant gap in the existing literature regarding evaluation integrity is addressed; many current studies lack a rigorous protocol to prevent data leakage between structurally similar floorplans. The methodology presented here is contrasted with modern frontier strategies, such as *nearest-frontier* or *utility-based* selection, to demonstrate that incorporating a predictive 'free-space gain' into traditional cost-utility functions provides a more forward-looking exploration strategy.

Ultimately, this review not only guided our system design choices but also revealed critical directions for future research, particularly in predictive spatial reasoning, real-time exploration, and decision-making under uncertainty.

3
MAP COMPLETION

This chapter presents the map completion component of the thesis, which constitutes the primary learning-based contribution enabling the robot to reason beyond its current sensor observations. This thesis addresses the challenge of scene extrapolation—defined as the probabilistic estimation of unmapped cells based on observed context—through four distinct formulations. A training-free *rule-based geometric baseline* provides a structural reference point. Three learning-based formulations address different completion settings: *upper-to-lower local completion*, *global map completion*, and *frontier-centered local completion*—the robotically most relevant formulation, whose best-performing model is subsequently deployed online in the exploration pipeline.

The first formulation, *upper-to-lower local completion*, provides the upper half of a 32×32 patch and requires the model to generate the lower half. This setup enables controlled evaluation of conditional generation under a structured directional prior [1]. The second formulation, *global map completion*, operates on full partial maps resized to 128×128 pixels and seeks to complete the entire unobserved region from a single exploration snapshot [2]. The third and robotically most relevant formulation, *frontier-centered local completion*, operates on 32×32 patches whose format matches those extracted around frontiers during active exploration; all models under this formulation are trained and evaluated offline on a held-out patch dataset, and the best-performing model is subsequently deployed online in the exploration pipeline. In addition, a *rule-based geometric baseline* provides a training-free reference and establishes the extent to which pure structural reasoning can approximate data-driven completion [9].

The datasets used to train and evaluate all models are described in Chapter 6; this chapter focuses exclusively on model architectures, mathematical formulations, and completion objectives. All experimental results, quantitative comparisons, and model selection rationale are reported in Chapter 7.

The chapter is organised as follows. Section 3.1 defines the map completion problem mathematically. Section 3.2 describes the evaluation protocol, including the distinction between FULL and MASKED evaluation scopes and the rationale for training on full-patch losses while evaluating primarily on masked regions. Section 3.3 presents the geometric baseline algorithms. Sections 3.4, 3.5, and 3.6 detail the model architectures for each formulation. Section 3.7 summarises the chapter.

The completion formulations studied in this chapter reflect the research progression described in Section 1.4. The rule-based geometric baseline and directional local completion formulation were developed first, serving as feasibility studies and controlled benchmarks. Global map completion followed as a model family comparison on realistic SLAM data. The frontier-centered local completion formulation — the most robotically relevant setting — emerged from practical observations made during the preceding stages and constitutes the primary contribution of this chapter.

3.1 Mathematical Problem Formulation

Let $M \in \{0, 1, 2\}^{H \times W}$ denote an occupancy grid map in which each cell is encoded as:

- 0 — free space (navigable),
- 1 — occupied (obstacle / wall),
- 2 — unknown or unobserved.

In practical robotic systems, M is constructed incrementally by a SLAM algorithm from sensor data. Let M_{obs} denote the set of observed cells and M_{miss} the unknown cells.

The central problem is to predict M_{miss} from M_{obs}, where $\mathcal{F}(\cdot)$ is a mapping from observed to predicted cells that may be a learned discriminative model (e.g., U-Net), a generative model (e.g., CVAE, GAN, diffusion), or a training-free rule-based algorithm (e.g., the geometric baseline presented in Section 3.3), as given in (3.1).

$$\hat{M}_{\text{miss}} = \mathcal{F}(M_{\text{obs}};\, \theta) \tag{3.1}$$

In all cases, the full predicted map merges observed and predicted regions, as given

in (3.2).

$$\hat{M} = M_{\text{obs}} \cup \hat{M}_{\text{miss}} \tag{3.2}$$

The exact definition of M_{miss} depends on the formulation:

(i) **Upper-to-lower:** M_{miss} is the lower half of a 32×32 patch; M_{obs} is the upper half.

(ii) **Global:** M_{miss} is the full set of unknown cells in a 128×128 SLAM map at exploration time t (paired at training time with the map at $t{+}3$ as the completion target for learning-based models; the rule-based geometric baseline also operates at this scale without a training target).

In the frontier-centered deployment setting, the completed patch $\hat{M}$ is used only to augment the frontier cost function; it is not written back to the SLAM map. The details of how $\hat{M}$ enters the cost function are described in Chapter 4.

3.2 Evaluation Protocol and the MASKED Metric

Two evaluation scopes are defined for all map completion experiments:

- **FULL**: Metrics computed over the entire output patch, capturing overall structural similarity but potentially overestimating completion quality.
- **MASKED**: Metrics computed exclusively over cells that were initially unknown in the input (M_{miss}), directly measuring the model's ability to infer new structure.

A similar focus on unknown-cell evaluation appears in [13] and [15], which restrict metrics to predicted or masked regions. The present work extends this principle to *irregular* frontier-shaped masks derived from real SLAM exploration, and applies it as the primary criterion across a systematic multi-family model comparison spanning thirteen model families. The key distinction from these prior studies lies in the mask geometry: [13] evaluates a clean geometric expansion region (4 m×4 m known, 6 m×6 m predicted), and [15] masks a regular 80×80 pixel centre patch, whereas frontier-centered patches in this thesis expose irregular, topology-dependent unknown regions that follow no fixed axis.

MASKED evaluation is the primary criterion for the frontier-centered setting. FULL metrics capture overall structural similarity across the entire patch and are

reported to enable comparison with prior work. MASKED evaluation directly measures prediction quality over the regions the model must actually infer, without conflating this with the preservation of already-observed cells.

Important: MASKED is an evaluation-only protocol.
All models in this thesis are trained using losses computed over the *full* predicted patch. A similar evaluation focus appears in [13] and [15], which restrict metrics to predicted or masked regions. The present work extends this principle to *irregular* frontier-shaped masks derived from real SLAM exploration, and applies it as the primary criterion across a systematic multi-family model comparison.

Restricting training loss to unknown cells only would prevent the model from learning to preserve observed structure, which is equally important for deployment. The MASKED protocol is therefore applied exclusively at test time to measure how well each model infers *new* information.

The sole exception is the global RePaint diffusion model (Section 3.5.3), which includes a masked L1 term ($w_{\mathrm{L1}} = 3.0$) to stabilise known-pixel preservation under iterative denoising. The frontier variant omits this term: at 32×32 scale, the RePaint re-imposition mechanism alone is sufficient to preserve known-pixel consistency, and the additional masked L1 term was found to destabilise training at this resolution without measurable benefit.

Model selection follows a two-tier criterion: for cross-family comparison, the primary selection score is the macro-average of $\mathrm{IoU}_{\mathrm{macro}}$ and $\mathrm{F1}_{\mathrm{macro}}$; for within-family UNet variant selection, an unweighted five-metric normalised average (Pixel Accuracy, F1-macro, IoU-macro, L1 inverted, normalised throughput) is used. In both tiers, models below 20 img/s are excluded from deployment selection. Supporting metrics include L1, RMSE, and SSIM. Full experimental results are in Chapter 7.

3.3 Rule-Based Geometric Baseline

Before introducing data-driven models, a rule-based geometric baseline is established [9]. This approach identifies door-like structures using image processing and completes adjacent regions by estimating structural elements such as walls and guide lines. Being training-free, it serves both as a reference point and as a component of the GEOM baseline in the global comparison.

3.3.1 Algorithm 1: Nearest-Structure Completion

Algorithm 1: Geometric Completion with Nearest Structural Elements (Algorithm 1)

Input: Partial occupancy map image
Output: Completed occupancy map
Convert image to grayscale;
Re-rotate if the image is axis-misaligned;
Detect bounding box via Canny edge detection [42];
Extract and cluster perpendicular and vertical guide lines via Hough transform [43], restricted to the bounding box;
Identify vertical and horizontal door candidates using wall–door–wall pixel intensity patterns with threshold-based counting;
foreach *detected door* **do**
  foreach *direction* $\in$ *{up-right, up-left, bottom-right, bottom-left}* **do**
    Find the nearest wall in the opposite direction;
    if *no wall found* **then**
      find the nearest guide line in the opposite direction;
    end
    Find the reference line in the same direction;
    if *both reference lines found* **then**
      Fill the bounded region with free-space pixels, skipping walls, guide lines, and door pixels;
    end
  end
end

The overall processing pipeline is illustrated in Figure 3.1, where each stage of the geometric door discovery and map completion process is visualized sequentially.

As shown in Figure 3.1(a)–(f), the pipeline begins with a re-rotation step to align the occupancy map into a canonical orientation, ensuring consistent geometric processing across all subsequent operations.

Following this alignment, vertical and horizontal door candidates are detected separately (Figure 3.1(b)–(c)) by identifying structural discontinuities within wall segments based on pixel intensity transitions.

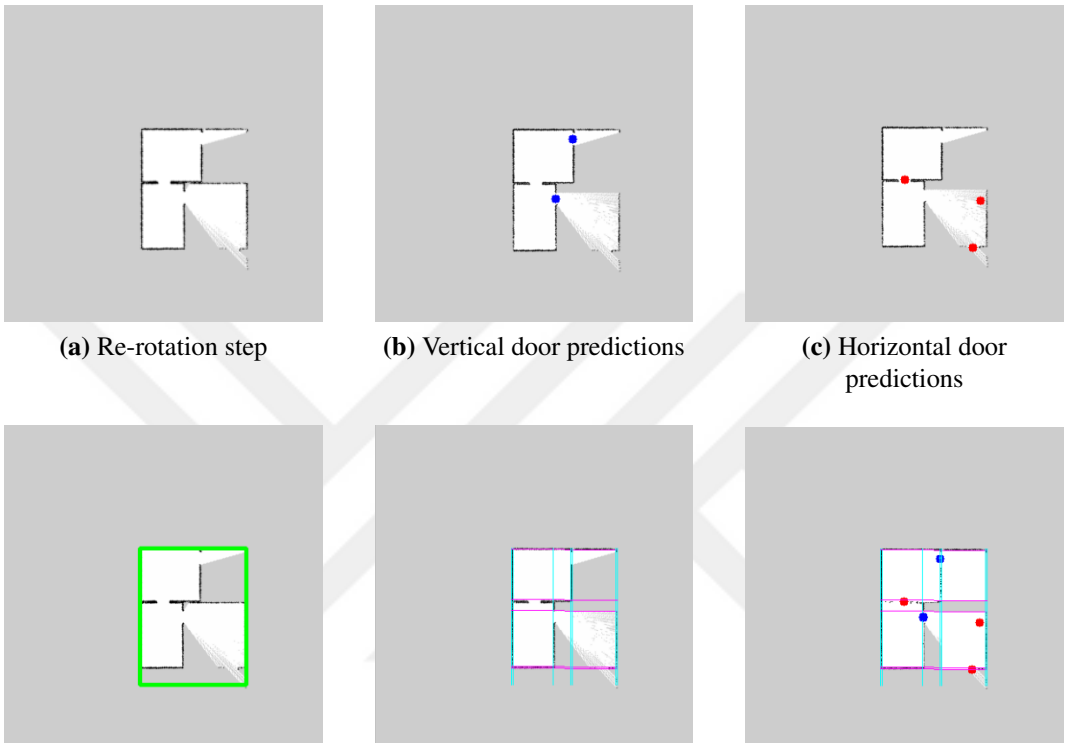
Using these detections, a bounding box is estimated to localize the region of interest (Figure 3.1(d)), which constrains and stabilizes the subsequent processing steps.

Next, dominant structural directions are captured through guide line extraction (Figure 3.1(e)), enforcing geometric consistency and supporting reliable structural inference.

Finally, detected doors and structural components are used to generate the

completed occupancy map (Figure 3.1(f)), where inferred free-space regions are filled and integrated with the original observed cells.

The pipeline combines affine transformation for alignment, Canny edge detection for bounding box estimation, and Hough transform for extracting dominant architectural structures, as illustrated in Figure 3.1.



(a) Re-rotation step **(b)** Vertical door predictions **(c)** Horizontal door predictions

(d) Bounding box detection **(e)** Extracted guide lines **(f)** Final completed map

Figure 3.1 Step-by-step execution of the geometric door discovery and map completion pipeline (Algorithm 1). See text for pipeline details.

3.3.2 Algorithm 2: Bounding-Box Fallback Completion

Algorithm 2 relaxes the reference line search by falling back to the bounding box when neither wall nor guide line is found. This increases coverage at the cost of more false positives in non-room regions.

Figure 3.2 provides an overview of both algorithms, illustrating the full pipeline from re-rotation and bounding box detection through door detection, guide line extraction, and final completion via Algorithm 1 (nearest wall/guideline) and Algorithm 2 (bounding-box fallback).

Algorithm 2: Bounding-Box-Based Completion (Algorithm 2)

Input: Partial occupancy map image
Output: Completed occupancy map
Execute the same preprocessing steps as Algorithm 1;
foreach *detected door* **do**
    foreach *direction* $\in$ *{up-right, up-left, bottom-right, bottom-left}* **do**
        Find the nearest wall to the bounding box in the opposite direction;
        if *no wall* **then**
            find nearest guide line; otherwise use bounding box;
        end
        Find reference line in the same direction;
        if *no reference found* **then**
            use bounding box as reference;
        end
        if *both reference lines found* **then**
            Fill region as in Algorithm 1;
        end
    end
end

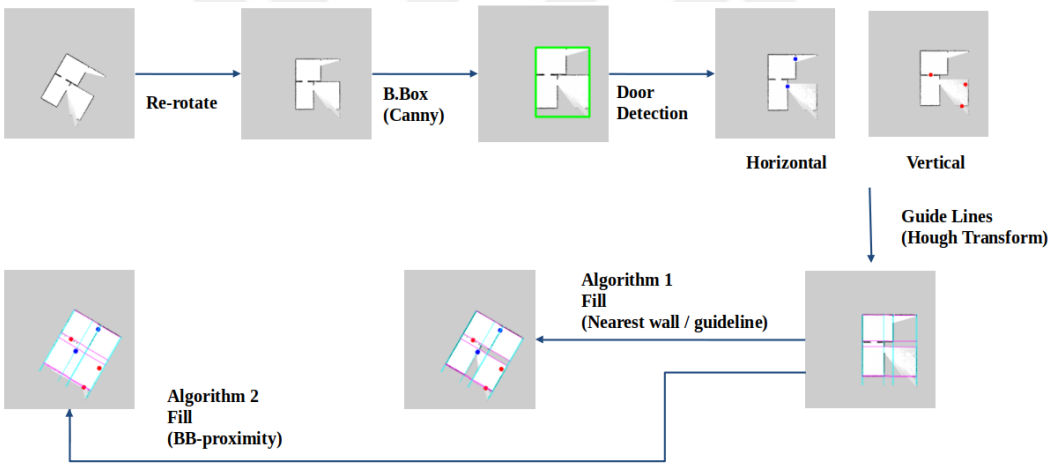


Figure 3.2 Overview of the geometric completion pipeline showing both algorithms. Starting from the input map, the pipeline proceeds through re-rotation, bounding box detection (Canny), door detection, and guide line extraction (Hough transform). Algorithm 1 fills regions using the nearest wall or guide line; Algorithm 2 falls back to the bounding box when no reference is found.

3.3.3 Limitations of the Geometric Baseline

Both algorithms are sensitive to predefined thresholds for door width detection and fail on non-Manhattan environments where walls are not axis-aligned. They also cannot generalise to the irregular unknown regions encountered in frontier-centered patches, where no structural door pattern is guaranteed to be present within the 32×32 window. These limitations motivate the learning-based approaches described in the following sections. Quantitative results, output comparisons between the two

algorithms, and their failure mode examples are presented in Chapter 7, Section 7.1.

3.4 Models for Upper-to-Lower Local Completion

The upper-to-lower formulation provides the top half of a 32×32 occupancy patch as input, where $M_{\text{upper}} \in \mathbb{R}^{1\times16\times32}$, and requires the model to reconstruct the lower half, as given in Equation (3.3).

$$\hat{M}_{\text{lower}} = \mathcal{F}(M_{\text{upper}};\, \theta) \tag{3.3}$$

This directional setup was motivated by prior work on sequential image generation, where models trained on MNIST with the upper half masked were shown to complete the lower half reliably [7]. Applying the same principle to occupancy patches—where the upper half represents the currently known region and the lower half the region below the robot's current sensor boundary—enables controlled analysis of conditional generation as a precursor to the more general frontier-centered task. This formulation was the subject of the initial generative map completion study [1]. Two models are evaluated: **ImageGPT** and **CVAE**.

3.4.1 ImageGPT

An autoregressive transformer [7] that treats each pixel as a discrete token. The model generates the lower half of the patch sequentially, token by token, conditioned on the upper half, directly operationalising the upper-to-lower completion paradigm.

Completion formulation.

Let the patch be flattened to a sequence of $T = 1024$ tokens $\mathbf{x} = (x_1, \ldots, x_T)$, where $x_i \in \{0, 1, 2\}$. Given the upper-half context $\mathbf{x}_{1:512}$, the model factorizes the lower-half distribution autoregressively, as given in (3.4).

$$p(\mathbf{x}_{513:1024} \mid \mathbf{x}_{1:512}) = \prod_{i=513}^{1024} p(x_i \mid \mathbf{x}_{1:i-1};\, \theta) \tag{3.4}$$

The architecture of the ImageGPT model adapted from [7] for upper-to-lower map completion is shown in Figure 3.3.

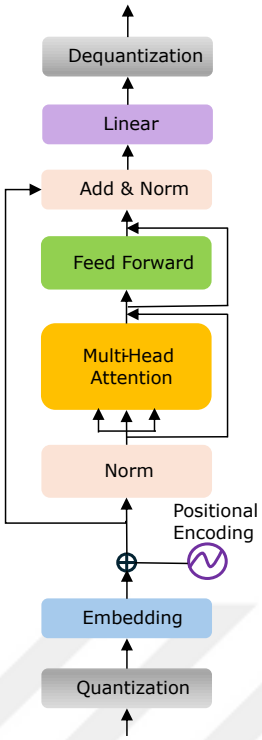


Figure 3.3 Architecture of the ImageGPT model adapted for upper-to-lower map completion.

Architecture.

The full 32×32 patch is quantised into three discrete token classes $\{0 = \text{free},\ 1 = \text{unknown},\ 2 = \text{obstacle}\}$, replacing the 16-centroid colour clustering of the original ImageGPT [7]. The sequence of 1024 tokens is processed by a lightweight GPT-style causal decoder consisting of 8 transformer blocks, a single attention head, and an embedding dimension of 16. There is no separate encoder: the upper-half tokens serve directly as the leading context, and the decoder generates the remaining 512 lower-half tokens autoregressively. The final logits are dequantised to reconstruct occupancy pixel values.

Training objective.

Cross-entropy loss over the full sequence, applied to both upper- and lower-half tokens for consistency, though only the lower half is evaluated, as given in (3.5).

$$\mathcal{L}_{\text{ImageGPT}} = -\sum_{i=1}^{1024} \log p(x_i \mid \mathbf{x}_{1:i-1};\, \theta) \tag{3.5}$$

Adam optimiser ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 10^{-9}$), initial learning rate 10^{-3}, halved if loss stagnates for three epochs; up to 250 epochs, batch size 25.

Deployment note.

Only the upper half is provided at inference; the decoder samples tokens autoregressively until the lower half is complete. Generating a single 32×32 patch requires approximately 6 seconds, making this model impractical for real-time frontier scoring but valuable as an offline quality baseline. Throughput and accuracy results are reported in Chapter 7.

3.4.2 Conditional Variational Autoencoder (CVAE)

The CVAE was selected for this formulation motivated by prior work demonstrating that VAE-based models can achieve strong structured prediction in robotic map completion [15]. Adding a conditioning mechanism—the upper-half observation—to the standard VAE framework [44, 45] was expected to improve coherence between the known and predicted regions by explicitly anchoring the latent representation to the observed context. The decoder is conditioned on the upper half and a sampled latent code z.

Completion formulation.

At inference, the decoder maps a sampled latent vector $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the observed upper half to the predicted lower half, as given in Equation (3.6).

$$\hat{M} = \mathcal{D}(z,\, M_{\text{upper}};\, \theta_D) \tag{3.6}$$

The architecture is shown in Figure 3.4. The model consists of an encoder that compresses the input into a latent representation and a decoder that reconstructs the output conditioned on both the latent vector and the input.

Convolutional and deconvolutional layers with normalization and activation functions are used to ensure stable training and effective feature learning.

Architecture.

The encoder and decoder each consist of 12 convolutional layers with batch normalisation and ReLU activations. The encoder outputs $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$ of a $d_z = 32$-dimensional latent code. The decoder concatenates z with the flattened upper half and uses transposed convolutions to reconstruct the complete 32×32 patch. Skip connections are disabled to prevent encoder–decoder information leakage at inference.

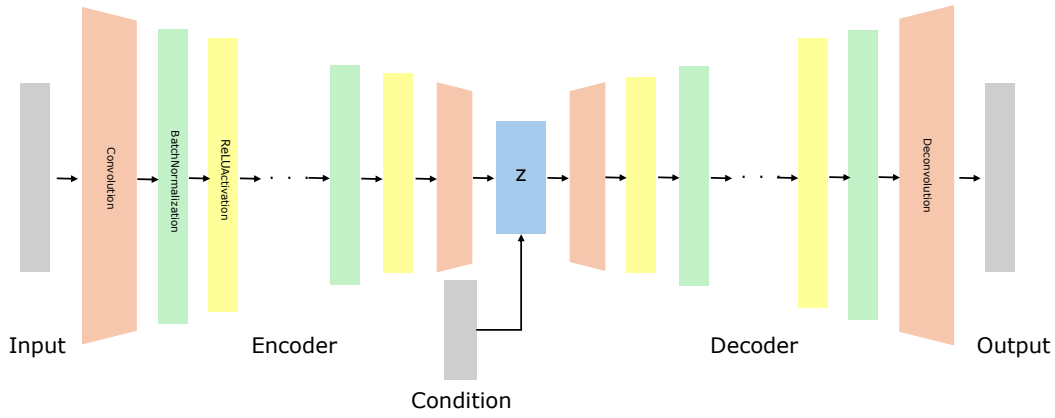


Figure 3.4 Architecture of the decoder-only CVAE for upper-to-lower completion.

Training objective.

The CVAE is trained with a combined MSE-SSIM reconstruction loss and an annealed KL divergence term, where α balances the MSE and SSIM reconstruction terms and β_{eff} is defined in (3.8) and annealed linearly from 0 to 0.5 over training, as given in (3.7).

$$\begin{aligned}\mathcal{L}_{\text{CVAE}} &= \alpha \cdot \text{MSE}(\hat{M}, M) + (1-\alpha) \cdot \big(1 - \text{SSIM}(\hat{M}, M)\big) \\ &\quad + \beta_{\text{eff}} \cdot \text{KLD}\big(q(z|M) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I})\big) \qquad (3.7) \\ \beta_{\text{eff}} &= \beta \cdot \text{anneal}(e;\, 0,\, 0.5) \qquad (3.8)\end{aligned}$$

Adam optimizer ($\text{lr} = 2 \times 10^{-4}$), batch size 25, up to 300 epochs; best results with $\beta = 0.5$, 12 convolutional layers, and latent dimension 32.

Deployment.

Encoder is discarded. A vector $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled, concatenated with the flattened upper half, and passed to the decoder. Inference time is ≈ 0.03 ms per patch.

Rationale for the Completion Formulation Progression

The global map completion study served two primary purposes: assessing whether CVAE generalises to a realistic SLAM-based dataset (DS1), and contributing a multi-model family comparison to the literature. Reviewer feedback during publication led to the inclusion of UNet and RePaint; UNet proved to be the strongest model, directly motivating its selection for the subsequent frontier-centered deployment. Global completion also represents a potential use case in its own right—for static offline completion of partially explored maps, or

as an adaptive strategy combining frontier-centered scoring in early exploration with global completion in late-stage cleanup. However, such integration carries non-trivial costs: structural information loss from downscaling, higher inference cost, reduced cache capacity for larger maps, and additional overhead from frontier detection on the predicted global map. These considerations motivated the frontier-centered patch approach as the primary deployment strategy.

Adapting the upper-to-lower formulation to live frontiers would require rotating each patch to align with the robot's heading direction, adding per-frontier computation overhead while still not producing a clean directional boundary in the general case. Furthermore, conditioning exclusively on the upper half introduces a structural bias: when the spatial pattern changes abruptly at the boundary, the model over-relies on the observed upper region and fails to extrapolate correctly, as observed in the failure cases reported in Section 7.2. The frontier-centered full-patch setting avoids these assumptions entirely, at the cost of requiring a model that is more robust to irregular mask shapes.

The implications of this structural difference for CVAE performance are discussed in Section 8.2.

3.5 Models for Global Map Completion

Global map completion operates on full partial maps resized to 128×128 pixels. The completion mapping is given in (3.9), where $M_{\text{partial}} \in \mathbb{R}^{1\times128\times128}$ is a SLAM map at exploration time t, paired at training time with the corresponding map at time $t{+}3$ as the completion target.

$$\hat{M} = \mathcal{F}(M_{\text{partial}};\, \theta) \tag{3.9}$$

where M_{partial} is a SLAM map at exploration time t, paired at training time with the corresponding map at time $t{+}3$ as the completion target. Six model families are compared under a unified evaluation protocol [2]. Unless noted otherwise, all global models use the Adam optimiser ($\text{lr} = 2 \times 10^{-4}$, batch size 32, seed 42) with ReduceLROnPlateau (patience 6, $\eta_{\min} = 10^{-6}$).

3.5.1 CVAE (Global, 128×128)

The global CVAE adopts a *pyramid conditioning* strategy in which the partial input map is average-pooled to three spatial scales and concatenated with decoder features

at the corresponding resolutions, enabling globally coherent completion without encoder–decoder skip connections (Figure 3.5).

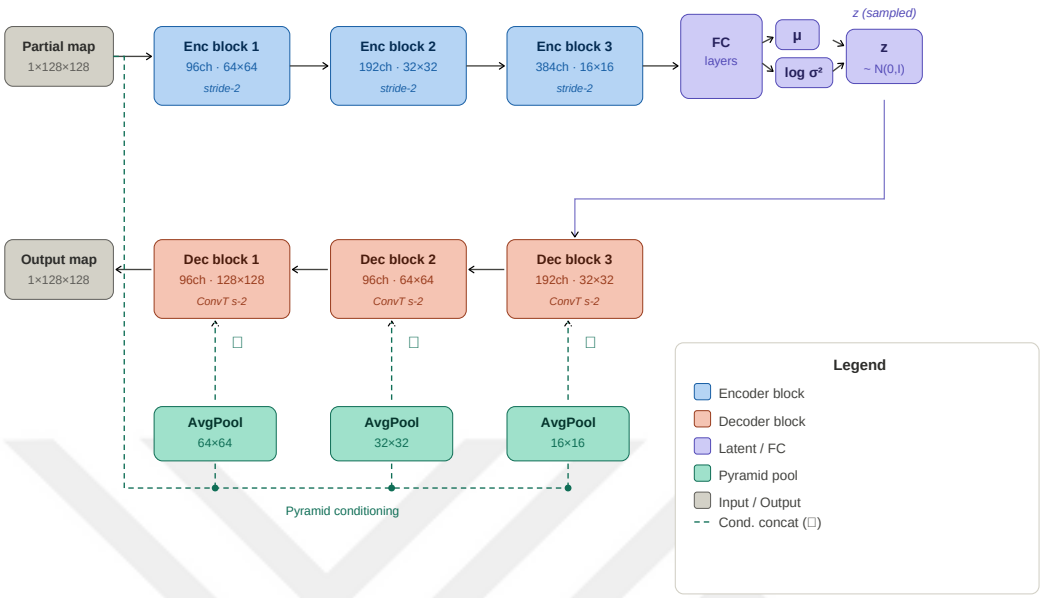


Figure 3.5 Global CVAE (128×128) with pyramid conditioning.

Architecture.

The encoder applies three strided convolutions (4×4, stride 2) with BatchNorm and ReLU, reducing the spatial resolution from 128×128 to 16×16 while expanding the channel width to 96, 192, and 384 respectively. The resulting feature map is flattened and projected to $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$ of a 128-dimensional latent code via fully connected layers. At inference, the latent code is obtained from the posterior mean $\boldsymbol{\mu} = \mathrm{Enc}(y)$; prior sampling $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is also supported. The decoder projects $z \in \mathbb{R}^{128}$ back to shape $[B, 384, 16, 16]$ via a linear layer. At each of the three upsampling stages, the partial input map is average-pooled to the current spatial resolution (16×16, 32×32, 64×64) and channel-concatenated with the decoder feature map before the transposed convolution, implementing pyramid conditioning. A final 1×1 convolution produces the output logits $[B, 1, 128, 128]$.

Training objective.

The global CVAE is trained with a BCE-logits reconstruction loss and a KL divergence term with linear annealing, where $\beta = 0.2$, as given in (3.10) and (3.11).

$$\mathcal{L}_{\text{CVAE}}^{\text{global}} = \mathcal{L}_{\text{BCE-logits}}(\hat{M}, M) + \beta_{\text{eff}} \cdot \mathcal{L}_{\text{KL}} \tag{3.10}$$

$$\beta_{\text{eff}} = \beta \cdot \text{anneal}(e;\ s = 0,\ e = 200) \tag{3.11}$$

3.5.2 Conditional GAN (CGAN / Pix2Pix)

A Pix2Pix-style CGAN [46] with a U-Net generator G and a spectral-normalised PatchGAN discriminator D, adapted from the architecture used in MCN-SLAM [19] and scaled to a lighter configuration suitable for 128×128 occupancy map completion, illustrated in Figure 3.6.

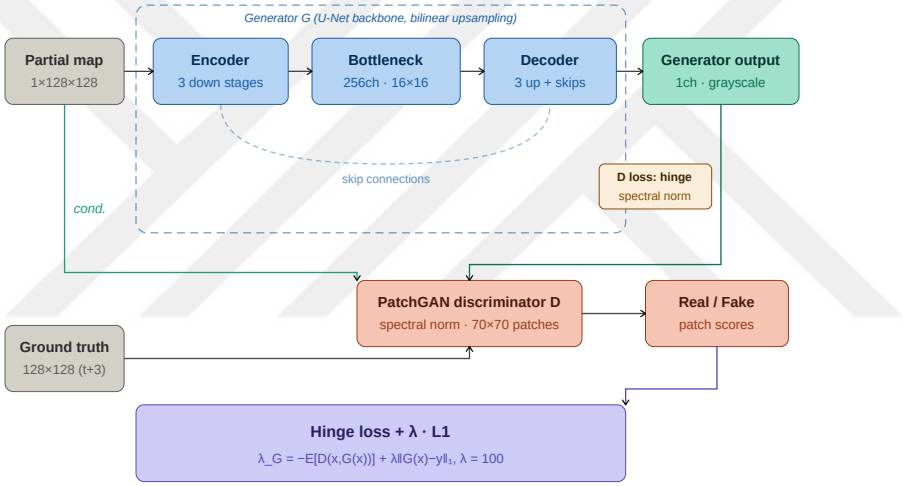


Figure 3.6 Global CGAN (Pix2Pix-hinge) for 128×128 map completion.

Architecture.

The generator follows a 3-down/3-up U-Net design with bilinear upsampling (to suppress checkerboard artifacts) and skip connections that propagate low-level spatial detail from encoder to decoder. The generator input is the 128×128 partial SLAM map; the target is the corresponding map at $t{+}3$. The discriminator operates on 70×70 overlapping patches with spectral normalisation, enforcing local structural consistency rather than global image realism.

Training objective.

Hinge-GAN adversarial loss with strong L1 regularization is used, as given in (3.12) and (3.13).

$$\mathcal{L}_D = \mathbb{E}[\max(0, 1 - D(x, y))] + \mathbb{E}[\max(0, 1 + D(x, G(x)))] \tag{3.12}$$

$$\mathcal{L}_G = -\mathbb{E}[D(x, G(x))] + \lambda \|G(x) - y\|_1 \quad \lambda = 100 \tag{3.13}$$

Training uses the Adam optimiser ($\beta_1 = 0.5$, $\beta_2 = 0.999$) with ReduceLROnPlateau (patience 4), for up to 1,003 epochs.

3.5.3 RePaint-Style Diffusion (DDPM, Global)

A conditioned DDPM [47, 48] that uses a U-Net noise predictor $\epsilon_\theta(x_t, t, x_{\text{cond}})$ operating on the concatenation $[x_t, x_{\text{cond}}]$ with learned timestep embeddings, illustrated in Figure 3.7.

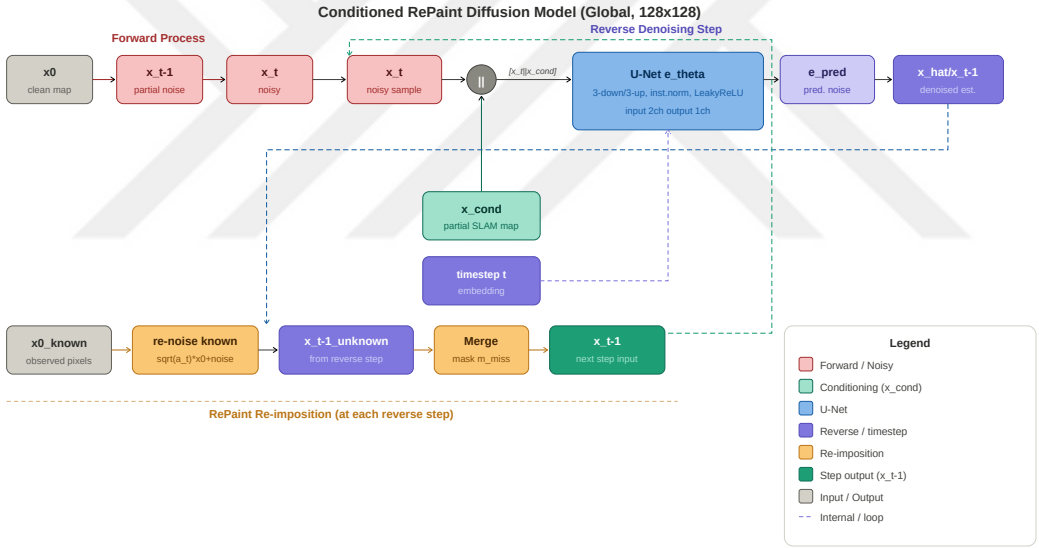


Figure 3.7 Conditioned RePaint diffusion model for global map completion (128×128).

Architecture.

The noise-prediction U-Net (3-down/3-up, instance normalisation, LeakyReLU) takes a 2-channel input formed by concatenating x_t and x_{cond}, and produces the predicted noise ϵ_{pred}. A linear β schedule ($T = 1000$, $\beta_{\text{start}} = 10^{-4}$, $\beta_{\text{end}} = 2 \times 10^{-2}$) is used. At each reverse step, the RePaint re-imposition mechanism [48] re-noises the known pixels to the current noise level and merges them with the denoised unknown region, preserving consistency with observed map areas throughout the denoising trajectory.

Training objective.

The RePaint model is trained with a noise prediction loss, an SSIM term, and a masked L1 term, where $\hat{x}_0$ is the estimated clean map, m_{miss} is the binary unknown-cell mask, and $w_{\text{SSIM}} = 2.0$, $w_{\text{L1}} = 3.0$, as given in (3.14).

$$\mathcal{L}^{\text{global}}_{\text{RePaint}} = \underbrace{\mathbb{E}\left\|\epsilon - \epsilon_\theta(x_t, t, x_{\text{cond}})\right\|_1}_{\text{noise prediction}} + w_{\text{SSIM}}\big(1 - \text{SSIM}(\hat{x}_0, y)\big) + w_{\text{L1}}\left\|(\hat{x}_0 - y) \odot m_{\text{miss}}\right\|_1 \tag{3.14}$$

The masked L1 term is applied during training only for this global variant; MASKED evaluation is applied exclusively at test time. Training uses ReduceLROnPlateau (patience 4).

3.5.4 Vision Transformer Encoder–Decoder (ViT)

A compact ViT [49] adapted into an encoder–decoder for dense map completion, processing the 128×128 input as 64 non-overlapping 16×16 patches (Figure 3.8).

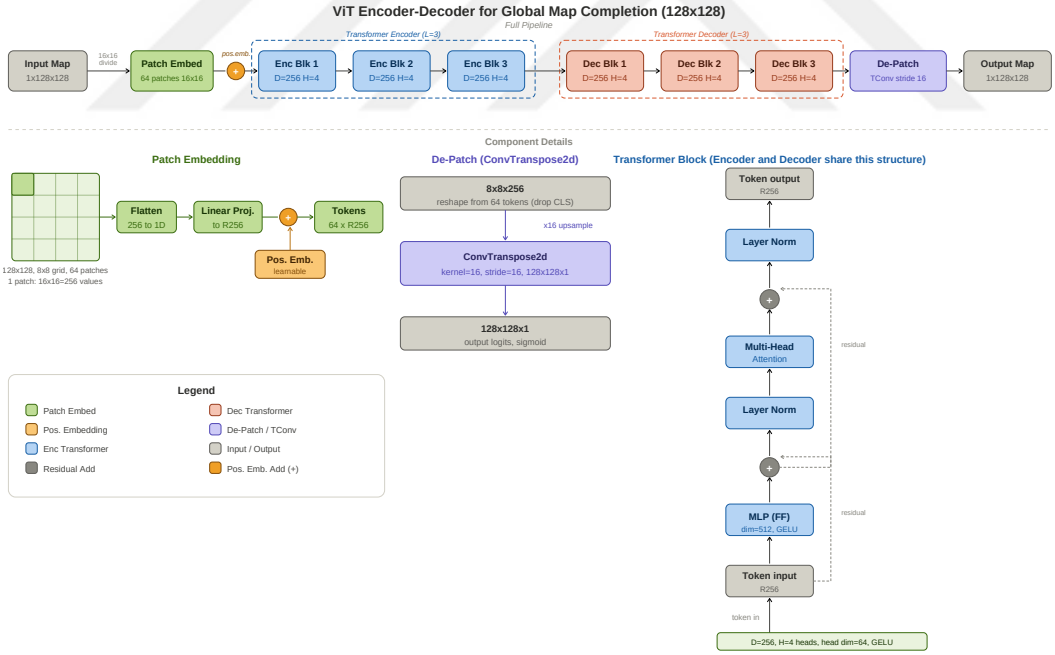


Figure 3.8 ViT encoder–decoder for global map completion (128×128).

Architecture.

The input map is divided into 64 non-overlapping 16×16 patches, each flattened and linearly projected to a $D = 256$ token embedding with learnable positional encoding. Three encoder and three decoder transformer blocks ($H = 4$ heads, head

dim 64, MLP dim 512) process the token sequence. The CLS token is dropped before a single transposed convolution (kernel 16, stride 16) restores the spatial resolution to $[B,\ 1,\ 128,\ 128]$. The coarse 16×16 tokenisation limits discrimination of thin walls and fine features; this is discussed in Chapter 7.

Training objective.

Smooth-L1 loss ($\beta = 0.1$) is computed over the full predicted map, as given in (3.15).

$$\mathcal{L}_{\text{ViT}} = \text{SmoothL1}\big(\hat{M}, M_{\text{target}};\ \beta = 0.1\big) \tag{3.15}$$

Training uses the Adam optimiser (weight decay 10^{-4}) with ReduceLROnPlateau, for up to 10,250 epochs.

3.5.5 Residual U-Net (Global, $3{+}3$ Stages)

A residual U-Net [50] with three encoder and three decoder stages, instance normalisation, and skip connections, trained with class-weighted cross-entropy and label smoothing (Figure 3.9).

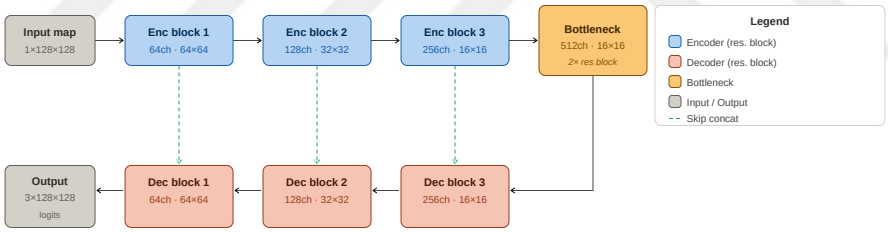


Figure 3.9 Residual U-Net (3+3 stages) for global 128×128 map completion.

Architecture.

Residual encoder blocks apply stride-2 convolutions to progressively reduce spatial resolution from 128×128 to 16×16, doubling the channel width at each stage (base=64). The bottleneck applies two residual blocks at 16×16 with 512 channels. Decoder blocks restore spatial resolution via bilinear upsampling; encoder features are injected via skip concatenation at each stage. A final 1×1 convolution produces three-class logits $[B,\ 3,\ 128,\ 128]$.

Completion formulation.

Three-class logits $\ell \in \mathbb{R}^3$ are produced per cell. For continuous metrics, the grayscale expectation maps free/unknown/obstacle logits to $v = \{0,\ 0.5,\ 1\}$, as given in (3.16). Discrete metrics use $\hat{c} = \arg\max_c \ell_c$.

$$\hat{g} = \sum_{c=1}^{3} \mathrm{softmax}(\ell)_c \cdot v_c \tag{3.16}$$

Training objective.

Class-weighted cross-entropy with label smoothing ($\varepsilon = 0.05$) is used, where $w_c = [1.0,\ 1.2,\ 1.0]$ are per-class weights for free/unknown/obstacle respectively, as given in (3.17).

$$\mathcal{L}_{\text{UNet}}^{\text{global}} = -\sum_i \sum_c w_c \cdot y_{ic}^{\text{smooth}} \log \mathrm{softmax}(\ell_{ic}) \tag{3.17}$$

Training uses ReduceLROnPlateau (factor 0.5), for up to 1,000 epochs.

3.5.6 GEOM: Rule-Based Baseline at Global Scale

The geometric baseline (Section 3.3) applied at 128×128 resolution. Training-free; runs on CPU; output quantised to $\{0, 0.5, 1\}$. The algorithm uses guide-line based door detection with minimum and maximum door width of 2–5 pixels, a maximum fill depth of 30 pixels, and wall clamping at map borders. Despite requiring no training, GEOM attains the highest SSIM among all models *under FULL evaluation*, demonstrating the power of geometric priors for structural preservation in rectilinear indoor environments. This advantage does not hold under the MASKED protocol—where performance drops to near zero—because 32×32 patches are too small to contain the door-frame and extended-wall cues that the GEOM algorithm requires; see Chapter 8 (Section 8.2) for detailed analysis.

3.6 Models for Frontier-Centered Local Completion

The frontier-centered setting is the most robotically demanding formulation. 32×32 patches are extracted from offline SLAM maps at positions corresponding to frontier cells; each patch contains an irregular unknown region whose shape and position varies across samples. The model must infer the unknown region while preserving all observed cells, where $M_{\text{patch}} \in \mathbb{R}^{1\times32\times32}$ encodes free (254), obstacle (0), and unknown (205) cells, as given in (3.18).

$$\hat{M}_{\text{miss}} = \mathcal{F}(M_{\text{patch}};\ \theta) \tag{3.18}$$

All models are trained and evaluated on this offline patch dataset (Chapter 6); the best-performing model is subsequently deployed in the live exploration pipeline, where its effect on exploration efficiency is measured at the system level (Chapter 7). All models share the same three-class input representation and are compared under the MASKED evaluation protocol (Section 3.2).

3.6.1 Residual U-Net Variants

Five U-Net variants are evaluated at the 32×32 frontier scale, all sharing the same encoder–decoder backbone with residual blocks, instance normalisation, and skip concatenation. The base model (`UNet33Res`) serves as the primary deployment candidate; the remaining variants systematically ablate architectural choices (attention gates, channel width, depth, and Fourier augmentation) to identify the factors that drive masked-region accuracy.

3.6.1.1 $\text{UNet}_{\text{base}}$ (`UNet33Res`)

The primary model is a compact residual U-Net for 32×32 frontier patches, illustrated in Figure 3.10. The input is a single-channel 32×32 patch processed through three downsampling stages and three upsampling stages with skip concatenation. The output is a three-class logit tensor $[B,\ 3,\ 32,\ 32]$.

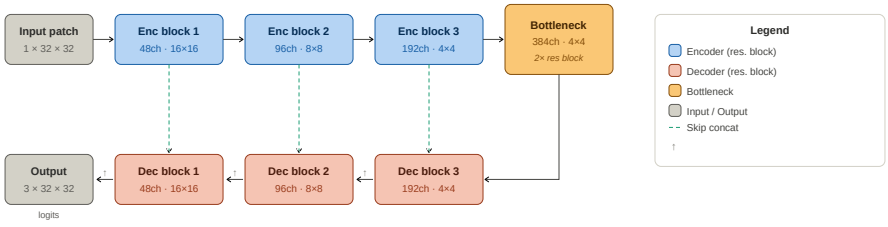


Figure 3.10 Residual U-Net ($\text{UNet}_{\text{base}}$, base$= 48$) for frontier-centered local completion.

Architecture.

The stem applies two 3×3 residual convolutions with instance normalisation and ReLU. Three downsampling stages (residual block + stride-2 convolution) reduce spatial resolution by $2\times$, $4\times$, and $8\times$, expanding channels from 48 to 384. The bottleneck applies two residual blocks at 4×4 resolution with 384 channels. Three upsampling stages restore spatial resolution via bilinear upsampling followed by skip concatenation and a residual convolution block. A final 1×1 convolution produces three-class logits $[B,\ 3,\ 32,\ 32]$.

Training objective.

Standard cross-entropy loss with fixed class weights $[1.0, 1.2, 1.0]$ and without Dice is used, as given in (3.19).

$$\mathcal{L}_{\text{UNet}} = \mathcal{L}_{\text{CE}} \tag{3.19}$$

Training uses ReduceLROnPlateau for up to 1,000 epochs, without dropout.

3.6.1.2 UNet_{att} (`UNet33ResAtt`)

The attention-augmented variant replaces standard decoder skip connections with attention-gated upsampling stages, as shown in Figure 3.11.

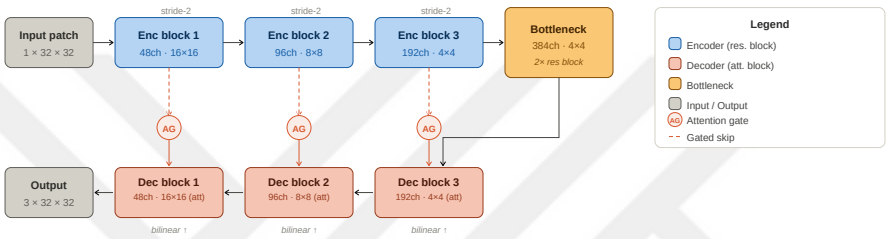


Figure 3.11 Attention U-Net (UNet_{att}) for frontier-centered local completion.

At each decoder level, an *attention gate* is computed, where $\mathbf{f}_i$ is the skip-connection feature, $\mathbf{g}$ is the gating signal from the upsampled decoder output, and σ denotes sigmoid, as given in (3.20).

$$\alpha_i = \sigma\big(\mathbf{W}_\alpha[\mathbf{f}_i + \mathbf{g}] + b_\alpha\big) \tag{3.20}$$

The reweighted skip feature $\alpha_i \odot \mathbf{f}_i$ is concatenated with the decoder output, aiming to suppress irrelevant structure far from the frontier. Architecture and training objective are otherwise identical to UNet_{new}, except that the base channel width remains 48.

3.6.1.3 UNet_{new} (`UNet33ResNew`)

This variant differs from $\text{UNet}_{\text{base}}$ in three respects: (1) the base channel width is increased from 48 to **64**, widening feature representations at every encoder and decoder stage; and (2) class weights are computed *dynamically* per batch rather than using the fixed per-class weights of $\text{UNet}_{\text{base}}$; and (3) random augmentations (horizontal and vertical flips, and $90°$ rotations) are applied during training. Both variants use the same loss structure, where $p_i = \text{softmax}(\ell_i)$ and $\gamma = 0.5$, as given

in (3.21) and (3.22).

$$\mathcal{L}_{\text{new}} = \mathcal{L}_{\text{CE}}^{\text{dynamic}} + \gamma \cdot \mathcal{L}_{\text{Dice}} \tag{3.21}$$

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_i p_i y_i}{\sum_i p_i + \sum_i y_i} \tag{3.22}$$

All other architectural choices (residual blocks, three down/up stages, skip connections, instance normalisation, dropout 0.1) are identical to $\text{UNet}_{\text{base}}$.

Under MASKED evaluation, UNet_{new} achieves a higher $\text{IoU}_{\text{macro}}$ than $\text{UNet}_{\text{base}}$ (0.336 vs.
0.316). However, when model selection is based on the five-metric normalised composite score (Pixel Accuracy, F1-macro, IoU-macro, inverted L1, and normalised throughput), $\text{UNet}_{\text{base}}$ ranks first (avg. 0.627), as it achieves a better balance across all criteria despite the lower individual IoU. The wider channel capacity of UNet_{new} does not translate to an overall deployment advantage on this dataset.

3.6.1.4 $\text{UNet}_{\text{deep}}$ (`UNet44Res`)

A four-down/four-up variant extends UNet_{new} with one extra downsampling stage, retaining the base channel width of **64**, extending the receptive field from ≈28 to ≈44 pixels. At 32×32 input resolution this produces a 2×2 bottleneck, which severely limits spatial information at the coarsest scale and prevents meaningful reconstruction. Despite the larger channel width, MASKED IoU drops to 0.261 (vs. 0.316 for $\text{UNet}_{\text{base}}$)—confirming that the additional downsampling stage is counterproductive at this resolution. Results are reported in Chapter 7.

3.6.2 LaMa-Style Inpainting Model (LamaUNet)

Motivated by large-mask inpainting [51], this variant replaces the bottleneck of UNet_{new} with two Fourier-augmented residual blocks (Figure 3.12).

FFCResidualBlock.

Each bottleneck block combines a local and a global frequency path:

- *Local path*: two 3×3 convolutions with residual skip.
- *Frequency path* (FourierUnit), where $\mathcal{F}$ denotes the 2D FFT, as given in

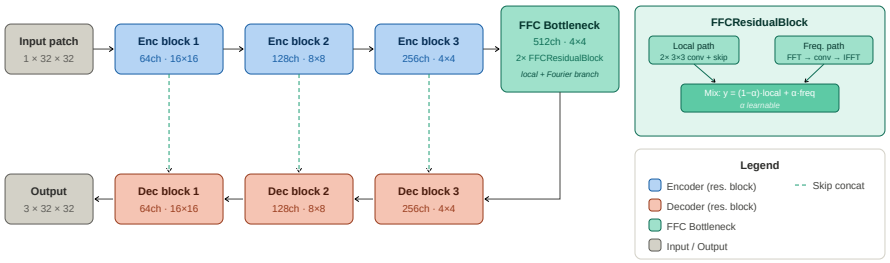


Figure 3.12 LaMa-style U-Net (LamaUNet, base= 64) for frontier-centered local completion.

(3.23):

$$\mathrm{FU}(\mathbf{x}) = \mathrm{IFFT}\big(\mathbf{W}_f \cdot [\mathrm{Re}(\mathcal{F}\{\mathbf{x}\}),\, \mathrm{Im}(\mathcal{F}\{\mathbf{x}\})]\big) \tag{3.23}$$

- Mixed output: $\mathbf{y} = (1-\alpha)\mathbf{y}_{\text{local}} + \alpha\mathbf{y}_{\text{freq}}$, with learnable α.

Training objective is identical in form to UNet_{new} (class-weighted CE + Dice, $\gamma = 0.5$, $\varepsilon = 0.05$, dynamic class weights) but with base= 64.

3.6.3 ImageGPT-Style Autoregressive Baseline ($\text{ImageGPT}_{\text{local}}$)

A lightweight ImageGPT-style causal transformer [7] adapted for frontier-centered patches, illustrated in Figure 3.13. The 32×32 patch is treated as a sequence of $T = 1024$ discrete tokens $\in \{0, 1, 2\}$ and trained in a fully supervised manner over all tokens simultaneously. Unlike the upper-to-lower variant (8 layers, 1 head, embedding dim 16), this model uses 3 layers, 2 attention heads, and embedding dimension 64.

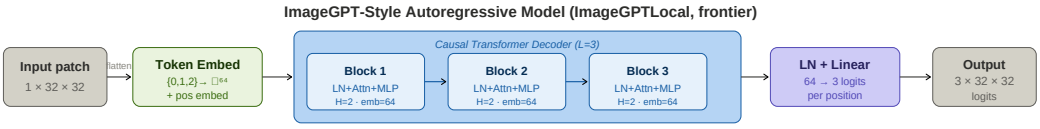


Figure 3.13 $\text{ImageGPT}_{\text{local}}$ causal transformer ($\text{ImageGPT}_{\text{local}}$) for frontier-centered completion.

Architecture.

The patch is flattened into 1024 tokens, each embedded from $\{0, 1, 2\}$ to $\mathbb{R}^{64}$ with a learnable positional embedding. Three causal transformer blocks (layer norm, 2-head self-attention, MLP dim 256, GELU, dropout 0.1) process the sequence; each token attends only to preceding tokens via a lower-triangular mask. A final linear projection produces three-class logits per position.

Training objective.

Standard cross-entropy is computed over all 1024 tokens, as given in (3.24).

$$\mathcal{L}_{\text{ImageGPT-local}} = -\sum_{i=1}^{1024} \log p(x_i \mid \mathbf{x}_{1:i-1};\, \theta) \tag{3.24}$$

The Adam optimiser ($\text{lr} = 10^{-3}$) with ReduceLROnPlateau (patience 4), for up to 1,000 epochs with batch size 25.

3.6.4 Row-Wise LSTM Baseline (LSTM_{row})

A bidirectional LSTM (`LSTMRowModel`) that processes each 32×32 patch as 32 independent row-wise 1D sequences, providing a baseline that captures horizontal context without any 2D spatial inductive bias. The LSTM cell mechanism [52] is illustrated in Figure 3.14; the row-wise architecture is shown in Figure 3.15.

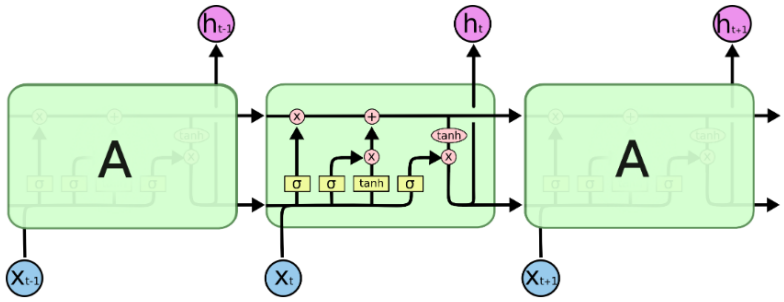


Figure 3.14 LSTM cell with forget, input, and output gates.

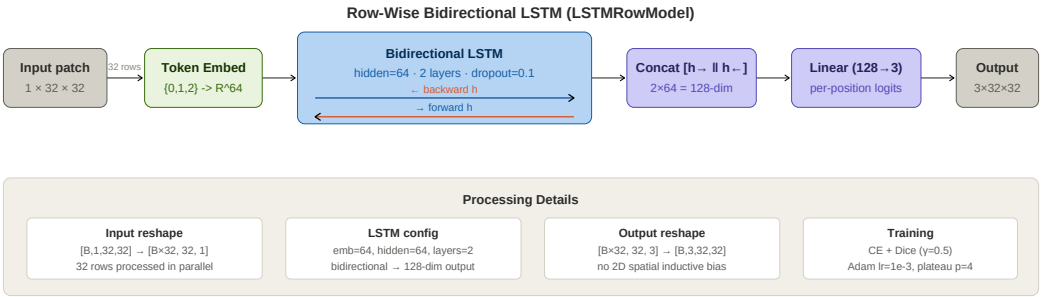


Figure 3.15 Row-wise bidirectional LSTM (LSTM_{row}) for frontier-centered completion.

Architecture.

The 32×32 input is reshaped to $[B \times 32, 32, 1]$, treating each row as an independent sequence of length 32. Tokens $\in \{0, 1, 2\}$ are embedded to $\mathbb{R}^{64}$. A 2-layer bidirectional LSTM with hidden size $h = 64$ per direction processes

each row; the concatenated forward and backward states ($2h = 128$ dimensions) are projected by a linear classifier to three logits per position. The output is reshaped to $[B, 3, 32, 32]$. Since rows are processed independently, cross-row spatial dependencies are not modelled.

Training objective.

Cross-entropy with Dice auxiliary loss is used, as given in (3.25).

$$\mathcal{L}_{\text{LSTM}} = \mathcal{L}_{\text{CE}}^{\text{dynamic}} + 0.5 \cdot \mathcal{L}_{\text{Dice}} \tag{3.25}$$

Training uses the Adam optimiser (lr $= 10^{-3}$) with ReduceLROnPlateau (patience 4), for up to 1,000 epochs with batch size 25.

3.6.5 Diffusion-Based RePaint Model (Diffusion$_{\text{local}}$)

A RePaint-inspired diffusion model [48] adapted for frontier-centered inpainting, illustrated in Figure 3.16. The architecture follows the same conditioned noise-prediction U-Net structure as the global variant (Section 3.5.3), scaled to 32×32 input with base channels 64.

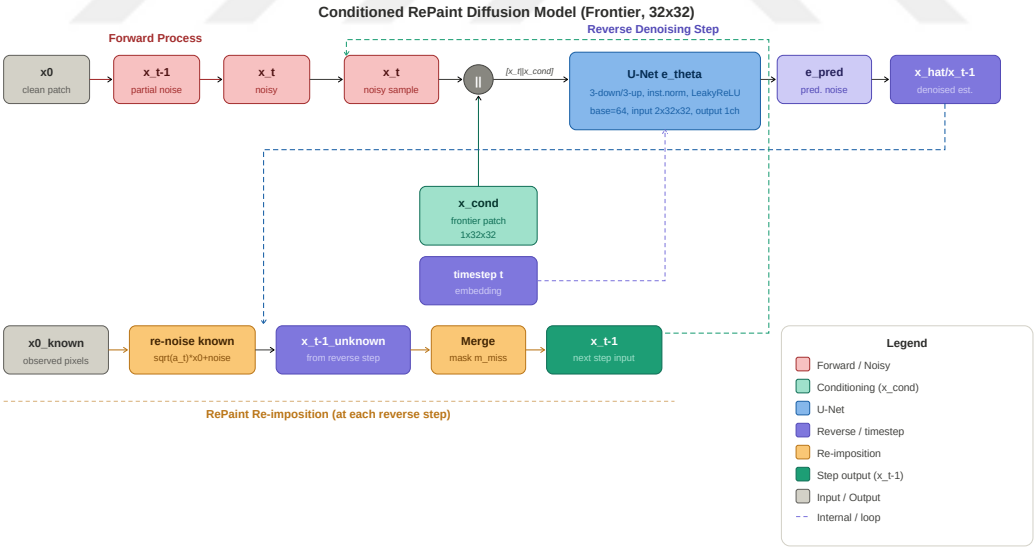


Figure 3.16 Conditioned RePaint diffusion model for frontier-centered completion (32×32).

Architecture.

The noise-prediction U-Net takes $[x_t, x_{\text{cond}}]$ as a 2-channel input with timestep embeddings injected at multiple scales (instance normalisation, LeakyReLU). The

forward process follows a linear β schedule ($T = 1000$, $\beta_{\text{start}} = 10^{-4}$, $\beta_{\text{end}} = 0.05$). At each reverse step, the RePaint re-imposition mechanism re-noises known pixels, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, as given in (3.26).

$$x_t^{\text{known}} = \sqrt{\bar{\alpha}_t}\, x_0^{\text{known}} + \sqrt{1 - \bar{\alpha}_t}\, \epsilon \tag{3.26}$$

Training objective.

The local RePaint model is trained with a noise prediction loss and an SSIM term, with $w_{\text{SSIM}} = 2.0$, as given in (3.27).

$$\mathcal{L}_{\text{RePaint}}^{\text{local}} = \mathbb{E}\big\|\epsilon - \epsilon_\theta(x_t, t, x_{\text{cond}})\big\|_1 + w_{\text{SSIM}}\big(1 - \text{SSIM}(\hat{x}_0, y)\big) \tag{3.27}$$

Unlike the global variant, no masked L1 term is applied ($w_{\text{L1}} = 0$); see Section 3.2 for the rationale. DDPM posterior sampling is used at inference. Training uses ReduceLROnPlateau for up to 1,000 epochs.

3.6.6 GAN Variants (Pix2Pix-Hinge and Seg-GAN)

Two GAN variants are evaluated at the 32×32 frontier scale, both adapting the global Pix2Pix-hinge architecture (Section 3.5, Figure 3.6) to the frontier setting. The key differences from the global variant are the input resolution (32×32 vs. 128×128), the resulting 1×1 bottleneck features, and—for Seg-GAN—a three-class logit output head replacing continuous grayscale. Both variants are illustrated in Figure 3.17.

Pix2Pix-Hinge.

The global Pix2Pix-hinge architecture is applied at 32×32 scale with the frontier patch as conditioning input, where $\lambda = 100$, as given in (3.28).

$$\mathcal{L}_G = -\mathbb{E}[D(x, G(x))] + \lambda \|G(x) - y\|_1 \tag{3.28}$$

Seg-GAN.

A segmentation-oriented variant that extends the generator to produce per-pixel three-class logits rather than continuous grayscale. The generator is supervised jointly, with $\lambda_{\text{adv}} = 0.05$, $\lambda_{\text{CE,mask}} = 2.0$, $\lambda_{\text{CE,full}} = 0.2$, as given in (3.29).

$$\mathcal{L}_G^{\text{Seg}} = \lambda_{\text{adv}}\, \mathcal{L}_{\text{adv}} + \lambda_{\text{CE,mask}}\, \mathcal{L}_{\text{CE}}^{\text{unknown}} + \lambda_{\text{CE,full}}\, \mathcal{L}_{\text{CE}}^{\text{full}} \tag{3.29}$$

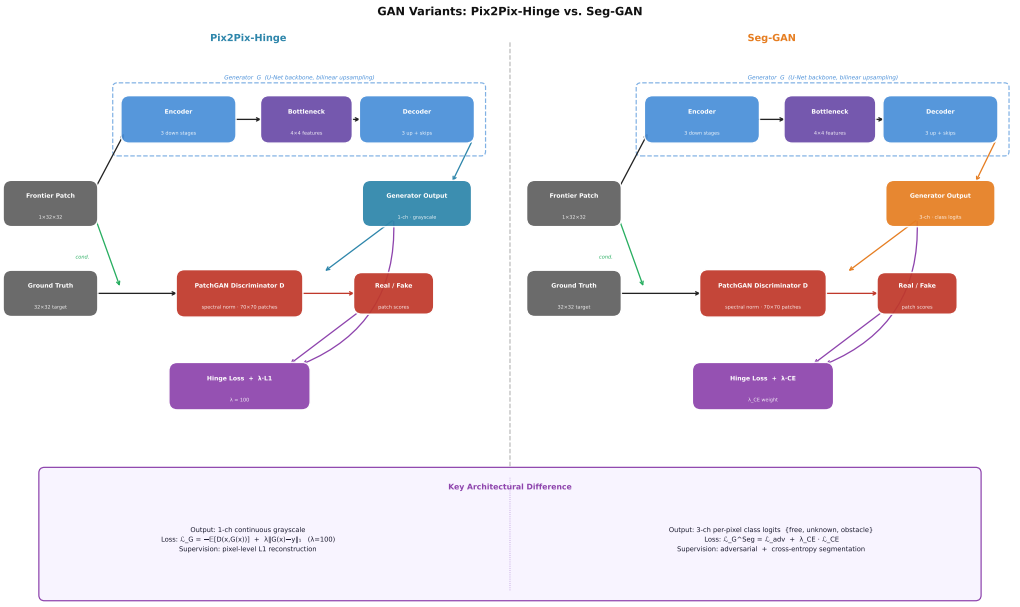


Figure 3.17 Pix2Pix-Hinge vs. Seg-GAN for frontier-centered completion.

The stronger weight on the unknown region aligns the generator objective with MASKED evaluation. Training uses the Adam optimiser ($\text{lr} = 2 \times 10^{-4}$) with 5 generator updates per discriminator step, for up to 1,000 epochs.

3.6.7 Conditional Variational Autoencoder (CVAE)

The CVAE for frontier-centered completion adopts the same pyramid conditioning strategy as the global variant (Section 3.5.1): the masked frontier patch (unknown cells set to 205) is average-pooled at three scales (16×16, 8×8, 4×4) and channel-concatenated at the corresponding decoder stages, with base channels 64 and latent dimension 32. The encoder is used during training to shape the latent posterior; at inference, $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled directly from the prior and passed to the decoder alongside the conditioning input, as given in (3.30).

$$\hat{M} = \mathcal{D}(z,\, M_{\text{patch}}^{\text{masked}};\, \theta_D) \tag{3.30}$$

The resulting architecture is illustrated in Figure 3.18.

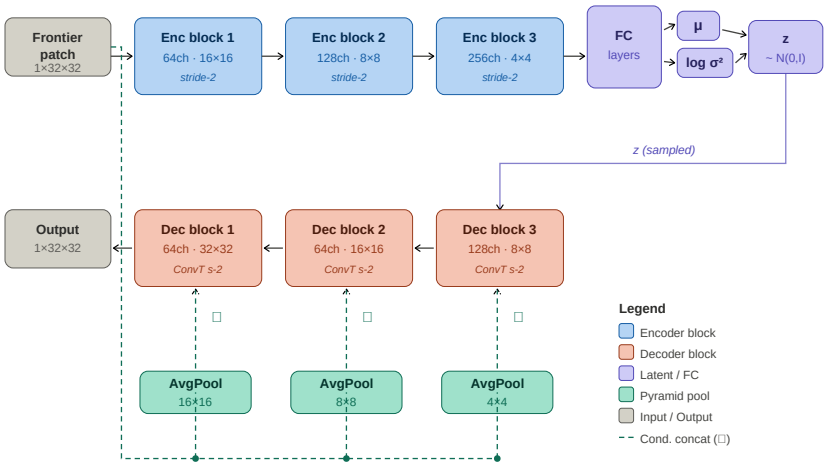


Figure 3.18 CVAE for frontier-centered completion (32×32) with pyramid conditioning.

Training objective.

The frontier CVAE is trained with a BCE-logits reconstruction loss and an annealed KL divergence term, as given in (3.31) and (3.32).

$$\mathcal{L}_{\text{CVAE}}^{\text{frontier}} = \mathcal{L}_{\text{BCE-logits}}(\hat{M}, M) + \beta_{\text{eff}} \cdot \mathcal{L}_{\text{KL}} \tag{3.31}$$

$$\beta_{\text{eff}} = 0.2 \cdot \text{anneal}(e;\, 0,\, 200) \tag{3.32}$$

Training uses ReduceLROnPlateau for up to 3,500 epochs. As discussed in Section 8.2, the CVAE's latent space does not generalise from fixed directional masks to irregular frontier shapes, resulting in low MASKED IoU.

3.6.8 Vision Transformer Encoder–Decoder (ViT$_{\text{frontier}}$)

A compact ViT encoder–decoder adapted for 32×32 frontier patches using a finer patch size than the global variant, illustrated in Figure 3.19.

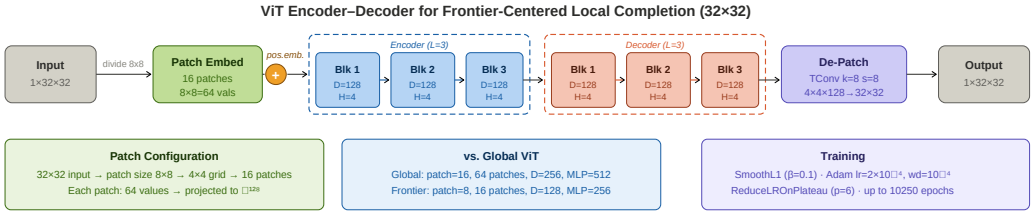


Figure 3.19 ViT encoder–decoder (ViT$_{\text{frontier}}$) for frontier-centered local completion (32×32).

Architecture.

The 32×32 input is divided into 16 non-overlapping 8×8 patches (vs. 64 patches at 16×16 in the global variant), each projected and position-encoded into $D = 128$-dimensional token embeddings. Three encoder and three decoder transformer blocks ($H = 4$ heads, MLP dim 256) process the sequence. The CLS token is dropped before a single transposed convolution (kernel and stride 8) restores $[B, 1, 32, 32]$ output logits.

Training objective.

Smooth-L1 loss ($\beta = 0.1$) is computed over the full predicted patch, as given in (3.33).

$$\mathcal{L}_{\text{ViT}}^{\text{frontier}} = \text{SmoothL1}\big(\hat{M}, M_{\text{target}};\, \beta = 0.1\big) \tag{3.33}$$

Training uses the Adam optimiser (weight decay 10^{-4}) with ReduceLROnPlateau, for up to 10,250 epochs.

3.7 Chapter Summary

This chapter described the map completion methodology underlying this thesis. Four formulations were addressed, each with distinct model families and mathematical objectives:

1. **Rule-based geometric baseline**: Training-free, effective on rectilinear layouts at global scale under FULL evaluation, but threshold-sensitive and unable to handle the irregular frontier unknown regions that fall below its minimum structural scale.

2. **Upper-to-lower local completion** (32×32): CVAE achieves strong accuracy with $\approx$ 0.03 ms inference; ImageGPT achieves comparable accuracy at substantially higher inference cost, precluding online use. Full results in Chapter 7.

3. **Global map completion** (128×128): Residual U-Net achieves the best aggregate accuracy; GEOM attains the highest SSIM under FULL evaluation; RePaint provides strong perceptual quality at throughput cost. Full results in Chapter 7.

4. **Frontier-centered local completion** (32×32): Multiple model families trained and evaluated offline under MASKED evaluation. Quantitative

results, dataset-size and patch-size ablations, and model selection rationale are in Chapter 7.

Key methodological note: MASKED evaluation is applied exclusively at test time. All training losses are computed over the full predicted patch. This ensures models learn to preserve observed structure while the evaluation protocol measures only genuine predictive accuracy over unknown cells. The sole exception is the global RePaint model, which includes a masked L1 term ($w_{\mathrm{L1}} = 3.0$) for training stability; the frontier RePaint variant does not use this term.

Based on the experimental analysis (Chapter 7), the residual $\mathrm{UNet}_{\mathrm{base}}$ is selected as the completion model for integration into the frontier-based exploration pipeline, deployed as a stateless batch predictor via a ROS service as described in Chapter 4.

4
MAP COMPLETION IN FRONTIER-BASED EXPLORATION

4.1 Overview

Autonomous mobile robots operating in partially known environments must continuously solve two coupled problems: *where to go next* (exploration) and *how to get there* (navigation). Frontier-based exploration addresses the first problem by identifying boundaries between known free space and unknown space—referred to as frontiers—and iteratively directing the robot toward them [53]. Classical frontier selection relies entirely on the current sensor-derived occupancy map. This limits its effectiveness whenever large unknown regions remain, because all frontiers appear equally uncertain from a geometric perspective alone.

This chapter describes how **learned map completion** is integrated into a frontier-based exploration framework to make selection more informed. The residual UNet model selected in Chapter 3 is deployed as a stateless batch predictor: for each candidate frontier, a 32×32 occupancy patch is extracted from the current SLAM map, completed by the UNet, and the result is used to compute the free-space gain r_f and a predicted-on-wall flag, both of which augment the frontier cost function. The design, implementation, and evaluation of a supplementary ground-truth oracle mode—used to observe navigation-level effects of ideal completion—is described separately in Chapter 5.

Three design principles bound the scope of the integration:

1. Map completion is used *only* at the exploration decision level. Path planning and navigation are performed on the original SLAM-generated occupancy map without modification.

2. The completion model is treated as a stateless patch predictor: it receives a single $P \times P$ occupancy patch and returns a completed version. No state is

maintained between planning cycles.

3. All behavioural parameters are controlled by launch configuration, so every configuration runs through an identical code path and can be compared under identical conditions.

The system is built as a direct extension of the `explore_lite` package [6], substantially modifying and extending its internals while preserving the original BFS-based frontier detection and ROS navigation interface.

4.2 System Architecture and Design Evolution

The current architecture is the result of an iterative design process motivated by performance and synchronisation problems encountered in an earlier implementation.

4.2.1 Earlier Design: Separate Publisher–Subscriber Node

The initial approach implemented map completion as a separate, independent ROS node written in Python. In this design, the completion module subscribed to the `map` topic and received frontier information—including distance and size—from the `explore_lite` node via a custom message. For each planning cycle, the completion node retrieved the published frontier list, located each frontier in the current map, extracted the corresponding patch, fed it to the completion model, computed revised costs, and published the best frontier as a navigation goal.

This design had three significant drawbacks. First, because the completion node operated independently, it duplicated several operations already performed inside `explore_lite` (coordinate lookups, distance computation, goal dispatch), introducing unnecessary computation. Second, the publisher–subscriber paradigm introduced synchronisation lag: the completion node operated on a frontier list that may have been computed one planning cycle earlier, producing stale decisions in fast-changing environments. Third, executing the full completion pipeline in Python—including model inference—introduced latency that exceeded the planning cycle budget on the evaluation hardware, since `explore_lite` is implemented in C++.

4.2.2 Current Design: Extended C++ Node with Batch Service

The current implementation addresses all three problems by extending the `explore_lite` node directly in C++. The map completion model is exposed as a dedicated ROS service (`MapCompletionBatch`) that receives a batch of frontier patches and returns completed patches, with no other responsibilities. The `explore_lite` node—now extended with completion, clustering, scoring, and progress monitoring logic—calls this service once per planning cycle and uses the results to augment its internal frontier cost computation.

This redesign achieves three improvements:

1. **No duplicated operations.** All frontier detection, distance computation, and goal dispatch occur inside a single node, eliminating the redundant parallel execution of the earlier design.
2. **Synchronous integration.** Completion results are computed and consumed within the same planning cycle, eliminating stale-frontier artifacts.
3. **Reduced latency.** The C++ host node executes all non-model logic at native speed; the Python completion service is invoked only once per cycle via a single batch call, amortising per-frontier inference cost.

In addition, a patch caching mechanism (Section 4.6.2) further reduces the number of model inference calls per cycle, providing an approximately order-of-magnitude reduction in completion latency when the frontier set is stable between consecutive cycles.

4.3 Full Pipeline Architecture

Each planning cycle executes a sequence of ordered stages inside the main search routine, with individual stages skipped when their corresponding launch parameter is disabled:

1. **BFS frontier detection.** A breadth-first search expands from the robot's nearest free cell and identifies all candidate frontier cells.
2. **Frontier clustering** (optional). Nearby frontiers are merged into representative clusters to reduce redundant completion calls and service latency.

3. **Batch map completion** (optional). All frontier patches are sent to the completion back-end in a single service call; results are cached.

4. **Completion score computation** (optional). Each frontier is scored for predicted free-space gain and checked for on-wall prediction.

5. **Euclidean distance computation.** Robot-to-frontier distances are calculated for all surviving frontiers.

6. **Cost computation and sorting.** Every frontier receives a scalar cost; the list is sorted in ascending order.

7. **Goal dispatch.** The first non-blacklisted frontier is sent to the ROS navigation stack as the next exploration target.

Several additional pipeline stages—gap closing and frontier filtering heuristics—are implemented in the codebase but were disabled in the reported exploration experiments. Their rationale and the circumstances under which they may be activated are discussed in Sections 4.6.3 and 4.7. The pipeline architecture and an example of frontier-centered patch extraction during exploration are illustrated in Figures 1.3 and 1.4 (Chapter 1). In particular, Figure 1.3 presents the overall system structure, including the interaction between the SLAM module, the exploration planner, and the map completion component, while Figure 1.4 demonstrates how local patches are extracted around frontier regions. These patches are then processed by the map completion module to guide exploration.

4.4 Frontier Extraction and Representation

4.4.1 BFS-Based Detection

At each planning cycle the occupancy grid is locked and a BFS expands from the cell nearest to the robot's current pose that carries the free-space label. For every free cell visited, its four neighbours are inspected: free neighbours are enqueued for further expansion, while cells at the boundary of known free space and unknown space trigger the construction of a new frontier.

A frontier cell is defined as an unknown cell adjacent to at least one free cell that has not yet been assigned to any frontier in the current cycle. Cells are marked to prevent double-counting.

A second BFS from the seed cell collects all connected frontier cells, accumulates their world coordinates to form the centroid, records the frontier size in cells, and

identifies the geometric midpoint. Frontiers whose metric size falls below the configured minimum (default $0.5\,\mathrm{m}$) are discarded. This detection logic follows the same principle as the original `explore_lite` BFS implementation; the extension introduces a snapped-centroid representation that ensures the patch centre always coincides with a genuine frontier cell, preventing the completion model from receiving patches centred on unknown space.

4.4.2 Frontier Point Representation: Centroid and Safest Point

Two distinct representative points are maintained per frontier, depending on the active mode.

Baseline mode.

The centroid is the arithmetic mean of all frontier cell world coordinates. Both the *patch-extraction point* and the *navigation goal* coincide with this centroid.

Map-completion mode.

The centroid is snapped to the actual frontier cell nearest to the arithmetic mean; this ensures the completion model always receives a patch whose centre lies on a genuine frontier cell. The *safest point*, by contrast, is set to the unsnapped arithmetic mean, which is typically slightly inside the free region and therefore a more stable navigation goal. The snapped centroid and safest point are defined as given in (4.1) and (4.2), where $\mathcal{F}$ is the set of frontier cell positions, $\mathbf{c}$ is the snapped centroid used for patch extraction, and $\mathbf{s}$ is the safest point used as the navigation goal.

$$\mathbf{c} = \arg\min_{\mathbf{p}_i \in \mathcal{F}} \left\| \mathbf{p}_i - \bar{\mathbf{p}} \right\|_2 \tag{4.1}$$

$$\mathbf{s} = \bar{\mathbf{p}} = \frac{1}{|\mathcal{F}|} \sum_{\mathbf{p}_i \in \mathcal{F}} \mathbf{p}_i \tag{4.2}$$

All downstream steps use $\mathbf{c}$ for patch extraction and $\mathbf{s}$ for goal dispatch.

4.5 Frontier Clustering

Without clustering, the batch completion service would receive one patch per raw frontier cell; in dense environments this number can be large enough to exceed the per-cycle latency budget imposed by the $10\,\mathrm{Hz}$ navigation loop. Clustering therefore serves two purposes: reducing redundant navigation targets and bounding the batch size dispatched to the completion service per planning cycle.

When many small frontier segments are detected in the same spatial region, directing the robot to each in turn is inefficient. The clustering stage merges frontiers whose centroids lie within a configured radius threshold (default $1.0\,\mathrm{m}$) into a single representative cluster. The representative is the frontier with the highest gain-to-distance ratio within the cluster, where s_f is the frontier size in meters and d_f is the Euclidean distance from the robot to frontier f, as given in (4.3).

$$f^* = \arg\max_{f \in C} \frac{s_f}{d_f + \epsilon} \tag{4.3}$$

Clustering is applied *before* map completion so that the completion service receives only the reduced set of representatives, reducing both service latency and cache memory usage. After clustering, costs are recomputed for the representatives.

4.6 Local Map Completion Integration

4.6.1 Operational Modes

The system supports two completion configurations:

Baseline mode.

No completion is performed. The frontier cost function uses only distance and size terms. This configuration reproduces the core behaviour of the original `explore_lite` baseline, enabling head-to-head comparison; optional stages such as frontier clustering are available but controlled by the same launch parameters as the MC configuration.

UNet model mode.

A pretrained residual UNet (Chapter 3, Section 3.6.1) is exposed as a ROS service. For each frontier, a $P \times P$ patch (default $P = 32$) is cropped from the current SLAM costmap centred on the snapped centroid $\mathbf{c}$. All patches are dispatched to the service in a *single batch call*, and the returned completed patches are stored in an in-memory cache. This mode constitutes the primary experimental configuration reported in Chapter 7.

A ground-truth oracle variant—in which frontier patches are drawn directly from a pre-aligned floor-plan image rather than from a neural model—is also implemented. This mode serves as an exploratory upper-bound reference: it was used in the `4bb74477` environment to characterize the expected benefit of ideal completion at the correct patch scale, and as the basis for a supplementary single-environment

navigation study. Due to time constraints, a systematic GT oracle comparison across all eight exploration environments was not conducted. The oracle exploration results are reported in Chapter 7 and the navigation study in Chapter 5.

4.6.2 Batch Service Architecture and Patch Caching

The batch completion architecture was introduced to replace the earlier sequential per-frontier design described in Section 4.2. In the batch design:

1. Before querying the model, each frontier's cache key—derived from its grid cell coordinates—is looked up. Patches younger than 30 seconds are reused without a new service call.
2. The remaining frontiers' raw $P \times P$ patches are serialised into a single batch request sent to the completion service.
3. The service returns completed patches in the same order; results are stored in the cache with a timestamp.

This design reduces the number of model inference calls per cycle from $|\mathcal{F}|$ to the number of cache misses, providing approximately an order-of-magnitude reduction in completion latency when the frontier set is stable between consecutive cycles.

4.6.3 Gap Closing

A morphological gap-closing operation is implemented that seals wall gaps narrower than a configured threshold (default $0.2\,\mathrm{m}$). This operation was disabled in the primary exploration experiments due to synchronisation risks and a close-once divergence problem described in Chapter 5. Future work on synchronisation-aware gap closing is discussed in Chapter 9.

4.6.4 Completion Score Computation

After batch completion, each frontier's completed patch is processed to produce two scores written into the internal frontier structure.

On-wall prediction flag.

A 5×5 kernel centred on the patch centre is inspected. If more than 8 of the 25 kernel pixels are classified as obstacle in the completed patch, the frontier is flagged

as being predicted on a wall. This flag indicates that the completion model believes the frontier lies *inside* a wall, making it likely unreachable.The wall penalty applied to on-wall frontiers is set to $c_{\text{wall}} = 100$, selected empirically to deprioritise them below most reachable alternatives under the default weight configuration, while remaining within the numerical range of the other cost terms.

Free-space gain score.

The predicted free-space gain r_f measures the fraction of unknown cells in the patch that the model predicts as free, normalized to $[0, 1]$ by the number of unknown cells, as given in (4.4).

$$r_f = \frac{|\{\mathbf{p} \in M_{\text{miss}} : \hat{M}(\mathbf{p}) = \text{free}\}|}{|M_{\text{miss}}|} \tag{4.4}$$

Completed patches are *not* written into the SLAM costmap: the navigation stack operates on the unmodified SLAM map throughout, ensuring that erroneous predictions cannot introduce failures in path planning.

4.7 Frontier Filtering

A set of lightweight heuristic filters is implemented to flag problematic frontier candidates before cost computation. In the GT oracle study (Chapter 5), these filters were applied as hard exclusions, permanently removing flagged frontiers. This revealed a key risk: in frontier-sparse environments, a single incorrect exclusion can halt exploration entirely. Accordingly, the production implementation assigns a large cost penalty to flagged frontiers rather than discarding them permanently, preserving them as fallback targets of last resort.

The following filters are implemented in the codebase:

Obstacle proximity filter.

Frontiers whose centroid lies within a threshold distance of the nearest obstacle are flagged. Such frontiers typically arise from SLAM noise and partial wall observations rather than genuine unexplored regions.

Trapped-corridor detection.

Frontiers enclosed in a narrow corridor with insufficient clearance for the robot footprint are identified by checking the free-space width along the approach path.

Thin-bridge detection.

Frontiers reachable only through a passage narrower than a configurable threshold are flagged as high-risk targets.

Small corner-pocket detection.

Frontiers corresponding to isolated small unknown pockets in corners are deprioritised.

Interior and exterior noise detection.

Spurious frontier cells generated by SLAM noise inside known free regions (interior noise) or beyond known walls (exterior noise) are identified and penalised.

All filters were disabled in the primary exploration experiments for two reasons. First, to isolate the effect of map completion alone, avoiding confounding variables from heuristics that have not yet been individually validated. Second, each filter introduces per-cycle cost; adding all filters simultaneously without prior ablation would make it impossible to attribute any performance change to a specific component. Their contribution in the oracle setting is discussed in Chapter 5; a dedicated ablation study is identified as future work (Chapter 9).

4.8 Frontier Scoring Mechanism

The cost assigned to each frontier governs the selection order. The formulation has three cases, where d is the Euclidean distance from the robot to the safest point s (meters), $s = |\mathcal{F}| \cdot r$ is the frontier size in meters (r = map resolution), w_d is the distance weight (default 3.0), w_s is the frontier-size weight (default 1.0), w_k is the free-space reward weight (default 40), and $c_{\text{wall}} = 100$ is a fixed penalty for frontiers predicted to lie inside a wall, as given in (4.5).

$$\text{cost}(f) = \begin{cases} w_d \cdot d - w_s \cdot s & \text{(BASE)} \\ w_d \cdot d - w_s \cdot s - w_k \cdot r_f & \text{(MC, reachable)} \\ w_d \cdot d - w_s \cdot s + c_{\text{wall}} & \text{(MC, on-wall)} \end{cases} \tag{4.5}$$

In the baseline case, the cost reduces to the classical distance-minus-size heuristic used in the original `explore_lite`. In the map-completion case, the free-space gain r_f in (4.4) augments the base cost with a predictive reward: frontiers whose neighbourhood is predicted to contain substantial new free space receive a lower

cost (i.e., are promoted in selection priority), while frontiers predicted to lie inside a wall receive a large penalty that places them below all feasible alternatives without permanently discarding them.

The wall penalty $c_{\text{wall}} = 100$ is calibrated to ensure that on-wall frontiers are visited only as a last resort when no other frontiers are available, which is the correct behaviour in environments where all accessible regions have been explored except a small set of SLAM-noise-induced or genuinely blocked boundaries.

After all costs are computed, the frontier list is sorted in ascending order of cost and the first non-blacklisted entry is dispatched as the navigation goal.

4.9 Target Selection and Execution

4.9.1 Goal Dispatch

Once the sorted frontier list is available, the exploration target is selected as follows:

1. The sorted list is scanned for the first frontier whose safest point $\mathbf{s}$ is not present in the blacklist.
2. If the selected target differs from the current active goal by more than $0.45\,\text{m}$, a new navigation goal is dispatched with $\mathbf{s}$ as the target pose.
3. If the frontier list is empty or all entries are blacklisted, exploration terminates.

4.9.2 Progress Monitoring and Blacklisting

A timeout mechanism detects stalled navigation. If the robot-to-target distance has not decreased by more than 8% (or by more than $0.30\,\text{m}$ absolute) within 20 seconds, the global progress timer is reset. If the duration since the last progress update exceeds the configured timeout (default $60\,\text{s}$), the current target is added to the blacklist and the next frontier is selected.

In the reported experiments, blacklisted frontiers and apparent stalls were handled via manual intervention rather than relying solely on the automatic blacklist, as described in Chapter 7. Extending the blacklist mechanism to detect and handle oscillatory behavior automatically is identified as future work (Chapter 9). Exploration is considered complete when no further frontiers are available.

4.9.3 Navigation Execution

Navigation is performed by the standard ROS `move_base` stack. The goal targets the safest point $\mathbf{s}$, which lies in or near free space. The completion model output is *never* written back to the costmap; path planning therefore operates on unmodified SLAM data throughout. This strict separation ensures that any observed performance difference between configurations is attributable solely to the frontier selection policy.

The navigation stack uses A* as the global planner and DWA as the local planner. TEB may be substituted as an alternative local planner without architectural changes. These components remain unchanged across all experimental configurations.

4.10 Configuration Parameters

Table 4.1 summarises the key launch parameters that control the pipeline behaviour. Parameters are grouped into mode flags, cost weights, and geometric thresholds.

Table 4.1 Key configuration parameters and their roles

Parameter	Default	Role
Mode flags		
`use_completed_map`	0	Enable completion-based scoring
`with_map_completion`	0	When completion enabled: 1 = UNet model; 0 = GT oracle (active only when completion is enabled)
`use_closing`	0	Apply gap closing to completed map
`use_closed_live_map`	0	Maintain gap-closed live map
`use_clustering`	0	Enable frontier clustering
`use_filtering`	0	Enable frontier filtering heuristics
Cost weights		
`potential_scale`	3.0	Distance weight w_d
`gain_scale`	1.0	Frontier-size weight w_s
`completion_known_scale`	40	Free-space reward weight w_k
Geometric thresholds		
`rect_size`	32	Patch side length P (cells)
`door_threshold_m`	0.2	Gap-closing threshold (m)
`min_frontier_size`	0.5	Minimum frontier length (m)
`progress_timeout`	60	Stall detection timeout (s)
`cluster_distance_threshold`	1.0	Clustering radius (m)

Setting the completion flag to disabled reproduces the core behavior of the original `explore_lite` baseline, enabling direct head-to-head comparison. All

parameters not listed in Table 4.1 are geometric and filter thresholds for the optional heuristic components and are documented in the codebase.

4.11 Algorithm

Algorithm 3 summarises the complete planning cycle for the primary experimental configuration (UNet model, no gap closing, no filtering).

Algorithm 3: Map Completion–Enhanced Frontier-Based Exploration

Input: SLAM occupancy map M, robot pose $\mathbf{p}$, pretrained completion model (optional)
Output: Navigation goal $\mathbf{g}$

$\mathcal{F} \leftarrow$ **BFS-FrontierDetection**(M, $\mathbf{p}$);
Discard frontiers with size $< s_{\min}$;
if *clustering enabled* **then**
  $\mathcal{F} \leftarrow$ **ClusterFrontiers**($\mathcal{F}$, δ_c);
end
if *completion enabled* **then**
  BatchCompletePatches($\mathcal{F}$) ; `// UNet model`
  foreach $f \in \mathcal{F}$ **do**
    Compute r_f and on-wall flag for f (Eq. 4.4);
  end
end
foreach $f \in \mathcal{F}$ **do**
  $d_f \leftarrow \|\mathbf{s}_f - \mathbf{p}\|_2$;
  $\mathrm{cost}(f) \leftarrow$ Eq. (4.5);
end
Sort $\mathcal{F}$ by cost (ascending);
foreach $f \in \mathcal{F}$ **do**
  if $\mathbf{s}_f \notin$ blacklist **then**
    return $\mathbf{s}_f$;
  end
end
return stop;

4.12 Summary

This chapter described the integration of learned map completion into a frontier-based exploration system. The pipeline extends `explore_lite` with optional stages for frontier clustering, batch patch completion, and completion-aware frontier scoring. The design evolved from an earlier publisher–subscriber architecture to a tightly integrated C++ extension of the explore_lite node, with a dedicated Python completion service handling model inference via a batch ROS service call.

The frontier cost function is augmented with a free-space gain reward r_f derived from the completion model's prediction, and a fixed penalty for frontiers predicted to lie inside a wall. Navigation and path planning remain entirely on the unmodified

SLAM map, ensuring compatibility with existing ROS infrastructure and enabling clean attribution of any performance gains to the frontier selection policy. Gap closing and frontier filtering heuristics are implemented but disabled in the reported experiments: each introduces per-cycle cost and carries a risk of irreversible exploration decisions in frontier-sparse environments. Future integration directions are discussed in Chapter 9.

A ground-truth oracle mode is also supported, in which patches are drawn from a pre-aligned floor-plan image rather than from the neural model. The dedicated navigation study using this oracle—including the effects of wall-closing and frontier-filtering under ideal conditions—is described in Chapter 5. The effectiveness of the UNet-based exploration system relative to the classical baseline is evaluated in Chapter 7.

5
GT ORACLE NAVIGATION STUDY

5.1 Motivation

The primary exploration system described in Chapter 4 uses the UNet completion model exclusively to augment frontier scoring, deliberately leaving the navigation map and the path planner unchanged. This decision to limit completion to the frontier scoring stage was partly motivated by an observation made during early exploration experiments: a significant portion of exploration time was spent not on navigating between frontiers but on repeated, unsuccessful path-planning attempts toward frontiers that could not be reached along the path selected by the global planner. Inspecting the SLAM maps revealed that these failures shared a common cause—narrow wall gaps left open by the SLAM algorithm, through which the global path planner attempted to route the robot.

This observation raised a natural question: if these wall gaps could be reliably closed, and if frontiers adjacent to them could be deprioritised, how much would navigation efficiency improve? Because testing this question with a learned model would conflate model accuracy with integration effects, a *ground-truth oracle* approach was adopted: frontier patches are drawn directly from a pre-aligned floor-plan image, ensuring perfect local completion independent of any model error. This allowed the effect of the integration strategy—wall closing and frontier filtering—to be studied in isolation.

This line of investigation produced a separate conference paper [8]. The results are reported here as a supplementary observation rather than as a target against which the UNet-based exploration system is evaluated: the two studies measure fundamentally different aspects of the pipeline, and the measured improvement is partly attributable to artifacts specific to the experimental setup, as discussed in Chapter 7 and Chapter 8.

5.2 Oracle Mode Design

5.2.1 Ground-Truth Map Loading and Alignment

The oracle mode operates by substituting the deployed completion model with a pre-loaded ground-truth floor plan. At system startup, the full floor-plan image for the current environment is read once and stored in memory. At each planning cycle, for each candidate frontier, the corresponding $P \times P$ patch is extracted from the stored floor-plan image. The floor-plan image is aligned with the SLAM coordinate frame using the map origin and resolution parameters, so that each cell in the floor plan corresponds to the same world position as the corresponding cell in the SLAM occupancy grid.

The oracle study differs fundamentally from the primary UNet-based system in its integration approach. In the primary system, completion influences frontier selection through the r_f score and the on-wall penalty in the cost function, while the navigation stack operates on the unmodified SLAM map. In the oracle study, the cost function uses only distance and size terms. Completion influences navigation through two mechanisms: frontier filtering, which uses the completed map to identify and permanently exclude invalid frontier candidates before goal selection; and a published `/completed_map` topic to which the navigation stack subscribes, allowing the path planner to operate on the oracle-completed representation rather than the raw SLAM map.

5.2.2 Wall Closing in the Oracle Setting

In the oracle navigation study, the gap-closing operation was activated. After each planning cycle, a morphological closing operation seals wall gaps narrower than the configured threshold in the completed map. In the oracle setting, the completed map is derived from the floor-plan image, so the closing operation acts on nearly perfect structural information, reliably sealing the wall gaps that would otherwise invite incorrect path-planning routes.

As described in Chapter 4 (Section 4.6.3), the gap-closing operation was disabled for the primary exploration experiments due to synchronization and algorithm limitations that arise when applying closing to the live SLAM map. In the oracle setting, these limitations are less severe because the floor-plan image is static and fully pre-loaded, but synchronization between the closed map and the SLAM-based navigation stack remained a source of occasional inconsistency. An additional risk in the non-oracle case is that an inaccurate completion model may introduce erroneous structure into the navigation map, causing the path planner to route

through predicted free regions that are in fact occupied—a failure mode that does not arise in the oracle setting but motivates the conservative design of the primary UNet-based system.

5.2.3 Frontier Filtering in the Oracle Setting

In the primary exploration experiments, frontier filtering was disabled because permanently excluding a frontier on the basis of an imperfect heuristic risks halting exploration entirely in frontier-sparse environments. In the oracle study, the obstacle proximity filter was enabled: frontiers whose centroid lies too close to an obstacle are permanently excluded. In this setting, the filter's effectiveness is enhanced by the oracle completion itself—completed patches reveal wall structures not yet observed by SLAM, enabling earlier and more reliable detection of frontiers that would otherwise appear reachable on the raw SLAM map alone.

5.3 Summary

This chapter described the supplementary GT oracle navigation study and its design rationale. The experimental setup and quantitative results are reported in Chapter 7 (Section 7.6); interpretation of the findings in the context of the broader system is discussed in Chapter 8.

6
DATASETS

This chapter introduces the dataset generation pipelines designed and implemented for the purposes of this research. To effectively support learning-based robotic tasks such as exploration and map completion, two complementary pipelines were developed, each producing datasets that serve distinct roles in the training and evaluation processes.

The first dataset was generated through a static and computationally lightweight conversion process. This method transforms raw 2D floorplan images into complete occupancy grids via a scripted PNG-to-PGM transformation. It enables rapid and large-scale data preparation, particularly advantageous for early-stage development, ablation studies, and controlled evaluation of generative models under artificial masking conditions.

The second dataset is derived from a more sophisticated simulation-based pipeline that leverages a full 3D environment in Gazebo and autonomous robot navigation using ROS. A mobile robot explores the environment, incrementally building occupancy maps through SLAM, thus capturing realistic artifacts such as sensor noise, partial observability, and map drift.

This simulation-driven pipeline produces two distinct dataset types:

- **Global Map Dataset:** Full occupancy maps captured at different time steps during exploration. Input-output pairs are created using maps at time t and $t{+}3$, suitable for global scene extrapolation and temporal forecasting.
- **Patch-Based Dataset:** Localised input-output patch pairs centred around frontier regions, extracted from the same sequential maps as the Global Map Dataset. These samples are extracted at multiple resolutions (32×32 and 64×64) and are designed for training models that learn to complete partial views of the environment.

Both dataset generation pipelines utilise 2D architectural floorplans from the HouseExpo dataset as their foundational input. These floorplans are processed and extended—either through scripted conversion or simulation-based exploration—to produce meaningful training data tailored to various tasks.

Together, the static and Gazebo-based datasets form a comprehensive and flexible framework for scalable model training, realistic evaluation, and both global and local map completion in autonomous robotics.

The dataset generation pipelines described in this chapter were developed iteratively, in parallel with the evolution of the completion formulations outlined in Section 1.4. Each pipeline addresses the specific data requirements of the formulation it supports: Dataset 2 was designed to provide large-scale directional patch data without the overhead of full simulation; Dataset 1 provides realistic sequential SLAM maps for global and frontier-centered evaluation; and Dataset 3 was constructed to ensure that the frontier-centered patch distribution closely matches the patches encountered during live robot exploration.

6.1 Source Dataset: HouseExpo

The HouseExpo dataset provides a large-scale collection of over 35,000 2D floor plans with 252,550 rooms in total, originally extracted from the SUNCG dataset [3]. Floor plans are distributed both as JSON files (line segments in metric coordinates) and as pre-rendered binary PNG images. In this work, the pre-rendered PNG images were used directly as input to the occupancy map generation pipeline. The binary representation encodes key architectural features such as walls, doors, and hallways in a format directly compatible with image-based processing, making the dataset particularly suitable for robotic simulation and environment modeling.

Each image in the dataset encodes spatial information by distinguishing between free and occupied areas using pixel intensities. This minimalistic yet informative format supports efficient parsing and conversion into occupancy grid representations, which are widely used in robotic mapping and navigation.

Although HouseExpo [3] is distributed together with PseudoSLAM, a lightweight Python-based simulation platform with an OpenAI Gym-compatible interface, this simulator was not used in the present work. PseudoSLAM is designed for training deep reinforcement learning policies via discrete action steps and does not provide a ROS interface; it exposes no `nav_msgs/OccupancyGrid` topics, no `tf` coordinate frames, and no `move_base` navigation stack. The frontier-based

exploration system developed in this thesis relies on all of these components, making direct integration with PseudoSLAM infeasible without a full ROS bridge layer. Instead, the HouseExpo floor plans were used solely as static map sources for offline dataset construction and for Gazebo-based exploration experiments, where a real SLAM stack (GMapping) and the `move_base` planner operated on the actual ROS infrastructure [4]. Adapting PseudoSLAM to support ROS-compatible SLAM output and continuous navigation would be a viable avenue for accelerating future data collection, and is noted as future work (Chapter 9).

6.2 Static Map Conversion for Generative Training

6.2.1 Motivation

This pipeline directly converts 2D architectural floor plans into labelled occupancy grid maps without simulating robot motion or sensor behaviour, enabling efficient and reproducible training data generation. This approach is conceptually similar to lightweight SLAM simulators such as PseudoSLAM [3], which also bypass full sensor simulation by operating directly on ground-truth maps. This method is well-suited for training deep learning models, offering rapid prototyping and controlled experimentation.

6.2.2 Pipeline Overview

The static map generation pipeline is composed of two primary stages, each serving a distinct role in preparing data for generative model training:

1. **Map Selection and Resizing:** A subset of $2{,}500$ HouseExpo maps was selected and uniformly resized to 320×320 pixels, providing a practical balance between architectural detail and computational feasibility.

2. **STL/SDF Model Creation:** The `image2gazebo` package is used to convert binary PNG floor plan images into 3D mesh models in STL format, along with corresponding SDF files that define the scene structure for the Gazebo simulator. This step is *optional*: if omitted, predefined static values are used as origin coordinates instead of SDF-derived values, as outlined in Algorithm 4.

3. **Occupancy Map Generation:** A custom Python script processes each floorplan image by classifying pixel intensities into semantic categories such as walls, free space, and unknown regions. These classifications are then encoded into grayscale values to produce a PGM (Portable GrayMap)

occupancy map. The resulting map format is directly compatible with SLAM-based frameworks and robotic simulation environments, providing a standardised input for generative model training.

6.2.3 Occupancy Map Generation and Pixel Classification

Each PNG file in the HouseExpo dataset is processed by classifying pixel values into predefined occupancy classes, enabling the transformation of architectural floorplans into semantically meaningful maps. The classification scheme is as follows:

- **Walls (black)**: Pixel value 0
- **Free space (white)**: Pixel value 254
- **Unknown regions (gray)**: Pixel value 205

These pixel values are chosen to mirror conventional representations used in robotic occupancy grid maps, where black indicates obstacles, white denotes navigable space, and gray signifies unexplored or uncertain areas.

To better simulate the output typically produced by SLAM systems, wall regions can optionally be subjected to morphological dilation. This operation broadens the wall boundaries, introducing realistic noise and thickness variation, which helps bridge the gap between idealised floorplans and practical robotic mapping results.

The final labelled image is saved in PGM format, which is natively supported by Gazebo's map server and other ROS-compatible mapping tools, ensuring seamless integration into simulation and navigation frameworks.

6.2.4 Algorithm: Static Map Conversion

Algorithm 4: Static Map Conversion with Origin and Resolution Handling

Input: `map.pdf` from HouseExpo, optionally `model.sdf`

Output: Occupancy map `map_code.pgm`

1. Load `map.pdf` from the HouseExpo dataset [3];

2. Set origin coordinates $(\texttt{origin_x}, \texttt{origin_y})$:

Option A: Extract origin from the associated `model.sdf`;

Option B: Assign default values (e.g., -10.0, -10.0) for static environments;

3. Convert each pixel (i_1, j_1) in the PNG image to world coordinates:

$$x_1 = j_1 \cdot r + \texttt{origin_x}, \quad y_1 = -i_1 \cdot r + \texttt{origin_y}$$

4. (Optional) Transform world coordinates to the robot frame using yaw and robot pose;

5. Define the PGM image size:

Option A: Calculate dimensions based on PNG resolution and scaling factor r;

Option B: Use a fixed size (e.g., 192×192) for consistency;

6. Convert world coordinates (x_2, y_2) back to PGM pixel positions (i_2, j_2);

7. Assign occupancy values: `wall` $\rightarrow 0$, `free space` $\rightarrow 254$, `unknown` $\rightarrow 205$;

8. (Optional) Apply morphological dilation to wall pixels for realistic thickness;

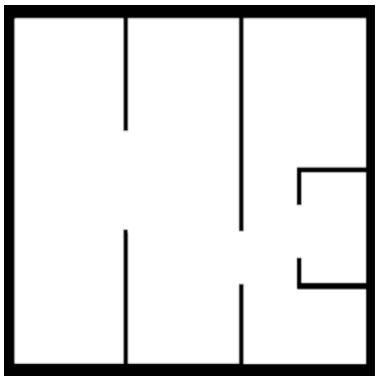
9. Export the resulting labelled image as `map_code.pgm`;

Figures 6.1 and 6.2 illustrate two examples from the dataset. In each case, the left image shows the original HouseExpo floorplan in PNG format, and the right image presents the corresponding PGM occupancy map generated according to Algorithm 4.

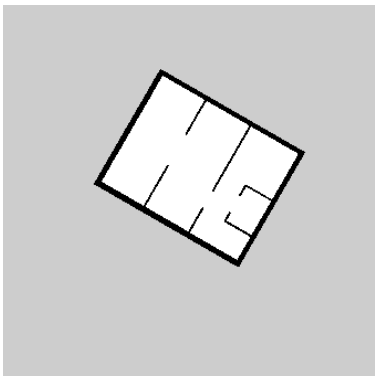
6.2.5 Applications and Benefits

The statically generated occupancy maps provide a versatile and efficient foundation for a range of tasks beyond the upper-to-lower completion formulation used in this thesis. Potential applications include:

- **Global map completion:** Full occupancy maps can be used to train models that predict entire unobserved regions from partial SLAM snapshots, without requiring simulation-based exploration data.

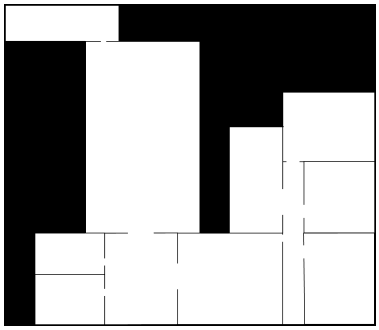


Original HouseExpo PNG image.

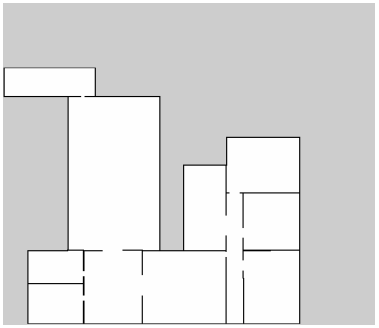


Converted PGM occupancy map.

Figure 6.1 Example using fixed values for PGM dimensions and origin coordinates in the static map conversion pipeline (Algorithm 4).



Original HouseExpo PNG image.



Converted PGM occupancy map.

Figure 6.2 Example using STL-based origin coordinates and dynamically computed PGM dimensions in the static map conversion pipeline (Algorithm 4).

- **Controlled ablation studies:** Artificial masks of various shapes, sizes, and positions can be applied to create precisely defined input-output conditions, enabling targeted evaluation of model behavior under different partial observability scenarios.
- **Frontier-centered patch training:** Frontier-like masks can be applied to sub-regions of the maps to generate synthetic frontier patch datasets, complementing simulation-based collection.
- **Efficiency:** Unlike simulation-based pipelines that require 3D navigation and sensor modelling, the static pipeline enables rapid dataset generation, significantly reducing computational overhead and accelerating the training cycle.

The complete source code and full dataset used in this pipeline are publicly available [5].

6.2.6 Static Dataset Output

Each output instance in the static dataset is a complete occupancy grid map stored in grayscale PGM format, with pixel values 0 (wall), 254 (free space), and 205 (unknown). An optional color visualization of the labeled map can also be generated for inspection purposes.

All generated maps conform to ROS and Gazebo standards. Dataset splits for the upper-to-lower local completion task use $2{,}500$ maps for training and 250 for sampling (validation/test), yielding $97{,}050$ training patches and $23{,}375$ sampling patches at 32×32 resolution (Section 6.4.1).

6.3 Gazebo-Based Simulation Pipeline

6.3.1 Motivation

While the static map conversion method offers speed and simplicity—making it highly suitable for rapid experimentation and initial model development—it does not reflect the complexities of real-world robotic perception. In practice, occupancy maps are affected by sensor noise, localisation drift, occlusions, and dynamic constraints, which are not captured by purely static pre-processing techniques.

To bridge this realism gap, a second dataset generation pipeline was developed using the Gazebo simulator in conjunction with ROS. This simulation-based pipeline

enables a mobile robot to autonomously explore a virtual environment derived from architectural floor plans, capturing occupancy maps that more closely resemble those produced during real-world exploration.

By simulating actual robot behaviour—including mapping, motion planning, and sensor feedback—this approach introduces natural imperfections and uncertainties into the resulting maps. These realistic artifacts are crucial for training and evaluating models intended for deployment in physical environments, where idealised data is rarely available.

6.3.2 Pipeline Overview

The Gazebo-based simulation pipeline consists of several sequential stages, starting with environment generation and autonomous exploration, and branching into two dataset types:

- **Global Sequential Map Dataset:** Full occupancy maps collected at different time steps, used for global map completion and geometric rule-based approaches.
- **Frontier Patch Dataset:** Localized input-output patch pairs extracted around frontiers from the same sequential maps, used for training data-driven generative models.

The pipeline stages are as follows:

- **3D Model Construction from 2D Maps:** Each architectural floorplan from HouseExpo is converted into a 3D environment for the Gazebo simulator using the `image2gazebo` package, which generates a 3D mesh (STL) and a scene description file (SDF).
- **Autonomous Exploration via TurtleBot3:** A TurtleBot3 robot navigates each environment autonomously, using Voronoi graph-based planning to visit meaningful exploration targets such as corridor centres and junctions. Navigation goals are determined using the `tuw_voronoi` ROS package.
- **Occupancy Map Logging with a Custom ROS Node:** As the robot moves, a custom ROS node triggers the `map_saver` utility to periodically record occupancy maps—particularly when the robot reaches Voronoi crossings. These maps contain realistic noise, partial information, and artifacts due to real-time SLAM.

The output of these first three stages forms the **global sequential dataset**, which contains full 2D occupancy maps captured at different time steps.

6.3.3 Custom Voronoi-Based Map Saving Node

A custom ROS node was developed to guide the robot to structurally meaningful locations and save occupancy maps upon arrival. The node integrates the `tuw_voronoi_graph` package to construct a Voronoi-based graph over the occupancy grid and identify crossing points—junctions and topologically meaningful decision locations. The custom node implements the following:

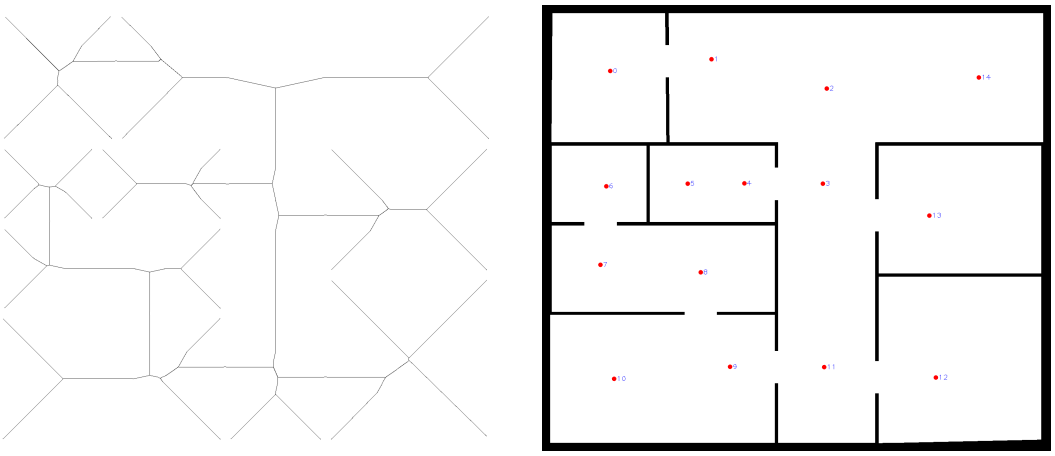
- Navigation goal dispatch to each identified crossing point
- Map saving upon arrival at each crossing point
- Logging of metadata such as robot pose and environment identifier

The Voronoi graph construction relies on the following components from the `tuw_voronoi_graph` package:

- **VoronoiGraph:** Generates the skeleton of free space using the medial axis transform.
- **Segment:** Encodes linear paths between nodes in the graph.
- **Crossing:** Detects intersection points on the graph.
- **SegmentExpander:** Dynamically expands the graph towards frontier regions to guide exploration.

The extracted Voronoi graph structure and the identified key locations, including crossings and segments, are illustrated in Figure 6.3.

In earlier stages of dataset construction, occupancy maps and local patches were extracted around *all* Voronoi graph points. However, it was observed that many of these points lie very close to each other along corridors, resulting in highly similar input–output samples and artificially inflated model performance metrics. To reduce this redundancy and ensure greater structural diversity, only Voronoi *crossing points*—corresponding to junctions and topologically meaningful decision locations—were retained in later stages. All datasets used in the final experiments are therefore generated using Voronoi crossings exclusively.



(a) Voronoi graph with path segments.

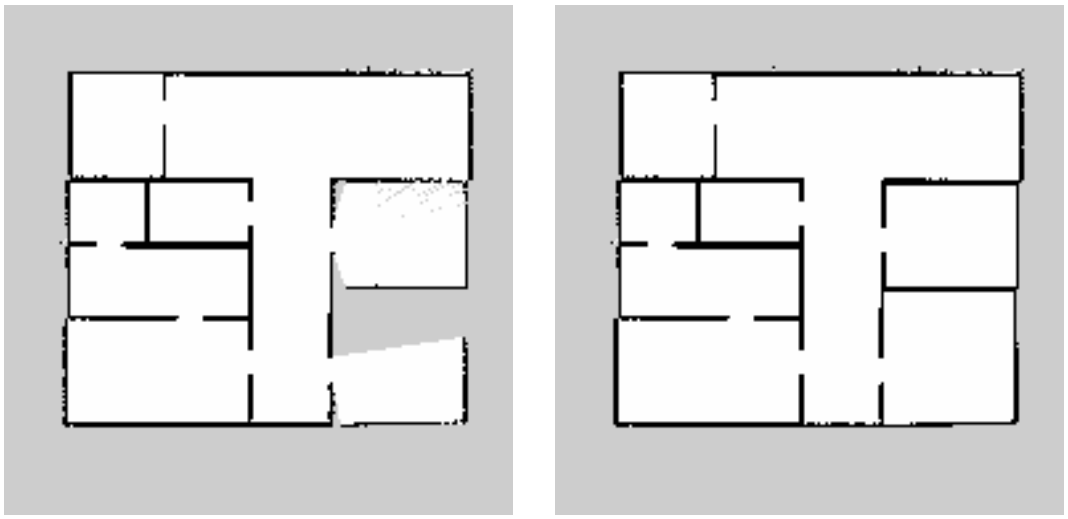
(b) Crossings marked on the map.

Figure 6.3 Voronoi graph structure and crossing points used for map saving triggers.

6.3.4 Input-Output Map Pairs for Global Completion

From the sequentially saved occupancy maps, input-output pairs are constructed by choosing the map at time t as the input and the map at time $t+3$ as the ground-truth output. This temporal gap allows the output to reflect an extended exploration state compared to the input, consistent with the dataset generation script [4].

An example of this input-output pairing is shown in Figure 6.4, where the partial observation at time step t is used as input, and the corresponding completed map at time step $t + 3$ serves as the ground-truth output.



Partial input image.

Completed output image.

Figure 6.4 Example of an input-output map pair generated from sequential exploration data, where the input corresponds to time t and the ground-truth output corresponds to time $t + 3$.

Figure 6.5 shows representative samples from Dataset 1.

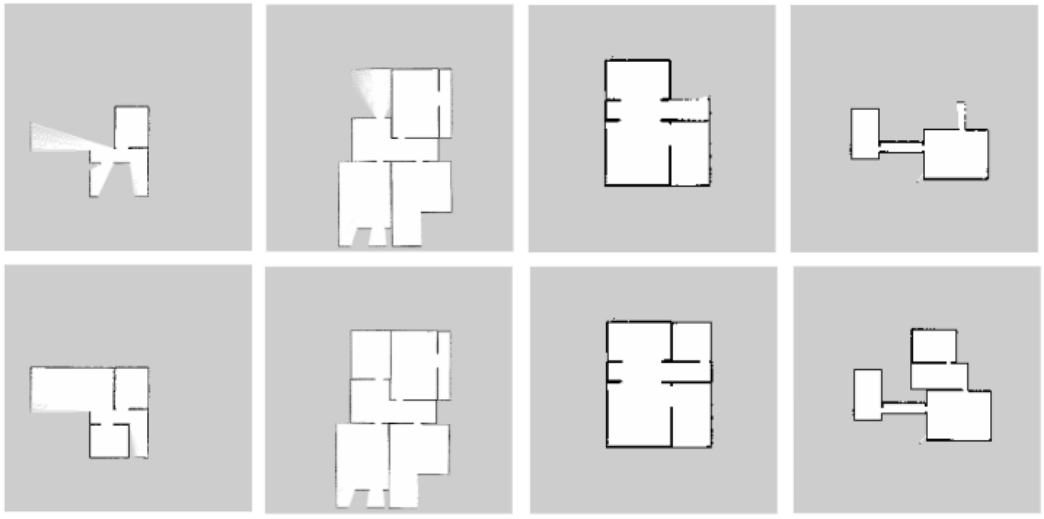


Figure 6.5 Representative occupancy maps from Dataset 1, collected via Gazebo-based simulation across diverse HouseExpo floor plans. Upper row: partial maps at time t. Lower row: corresponding more complete maps at time $t + 3$.

6.3.5 Post-Processing: Frontier-Based Patch Extraction

In addition to the global dataset, a **localised patch-based dataset** is extracted in a post-processing step designed to support training of generative models on partial local observations.

This process involves:

1. **File Matching:** Match input-output maps saved at time t and $t + 3$.
2. **Frontier Detection:** Identify frontier regions—boundaries between explored and unexplored space—in the input map using a free-cell adjacency criterion (a cell is a frontier if it is free and has at least one unknown neighbour).
3. **Patch Extraction:** For each frontier centroid, extract corresponding square patches from the input and output maps. Patch sizes of 32×32 and 64×64 are used in experiments; patches are centred on the frontier cell and discarded if they fall outside the map boundary.
4. **Metadata Logging:** Save patch pairs along with metadata such as coordinates, environment ID, and patch index in structured CSV files for traceability. Only the PNG image files are used in the reported experiments; the metadata files serve as auxiliary records for reproducibility.

Examples of extracted input-output patch pairs centred around frontier regions are illustrated in Figure 6.6.

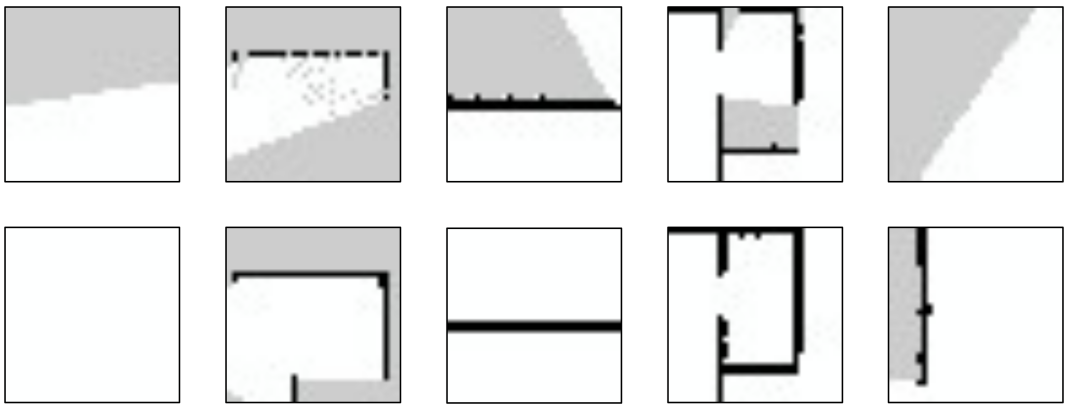


Figure 6.6 Extracted 32×32 input (top) and output (bottom) patches centred at frontiers. Note that the bounding boxes around the patches are for visualization purposes and do not represent wall structures. The leftmost column corresponds to the map in Figure 6.4.

6.3.6 Algorithm: Full Simulation-Based Dataset Generation

Algorithm 5: Gazebo-Based Dataset Generation Pipeline

Input: 2D map from HouseExpo

Output: (1) Full map sequences, (2) 32×32 patch pairs around frontier regions

foreach *map in HouseExpo folder* **do**
- Rename as `map.pdf`;
- Run `create_world_new.sh` to generate STL and SDF files;
- Launch Gazebo and simulate robot exploration;
- Save sequential occupancy maps using SLAM and `map_saver` at Voronoi crossing points;
- **foreach** *input-output map pair (time t, $t+3$)* **do**
  - Save full map input-output pair for global map completion;
  - Detect frontier regions in input map;
  - Find connected components and compute centroids;
  - **foreach** *centroid* **do**
    - Extract 32×32 patch pair centred at the frontier;
    - Save patches and log metadata to CSV;
  - **end**
- **end**

end

Figure 6.7 illustrates the frontier-centered patch extraction process: for a selected

frontier in the input map at time t, a 32×32 patch is extracted from both the input and the corresponding output map at time $t + 3$, forming an input-output pair for training.

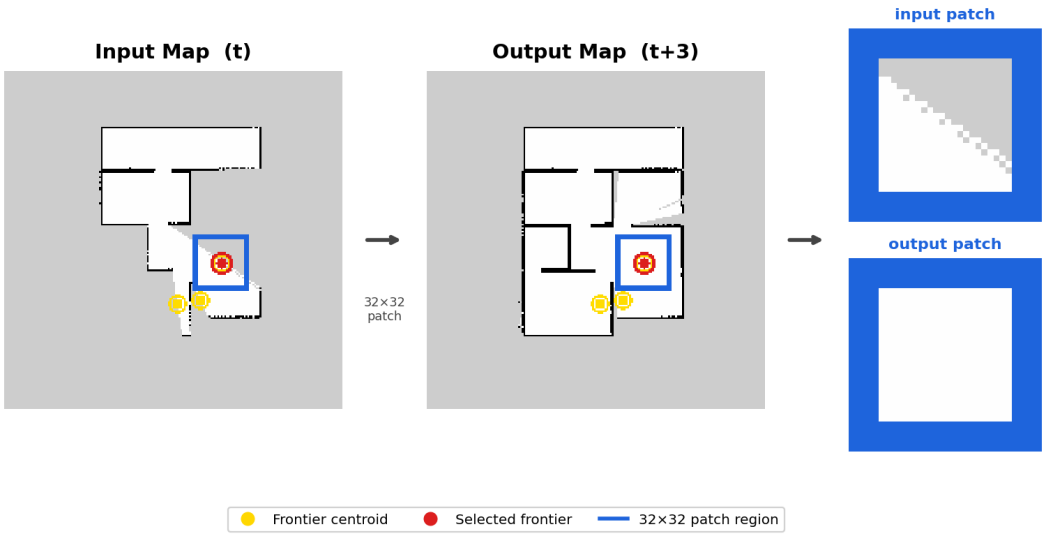


Figure 6.7 Frontier-centered patch extraction from sequential SLAM maps. A 32×32 patch centred on a selected frontier is extracted from both the input map at time t and the output map at time $t + 3$, forming a training pair. Yellow circles: frontier centroids; red circle: selected frontier; blue rectangle: 32×32 patch region.

6.3.7 Gazebo-Based Dataset Output

The final output of this pipeline consists of two interrelated datasets:

- **Global Map Completion Dataset:** Sequential occupancy map pairs at time t and $t + 3$, used for training generative and geometric models for global completion.
- **Frontier Patch Dataset:** Local input-output patch pairs around frontiers extracted from the same maps. Patch sizes of 32×32 and 64×64 are used in experiments, with structured metadata files for traceability.

Each sample is saved as a grayscale image pair: the input representing a partially explored region, and the output providing the more complete version. The complete pipeline, source code, and datasets are available supporting reproducibility and further research [4].

6.4 Dataset Splits and Quality Control

Ensuring valid train/validation/test splits is critical for obtaining reliable performance estimates. Two distinct failure modes were identified and addressed during dataset construction.

Data leakage via environment mixing.

In early dataset versions, patches from the same environment were distributed across all splits. This introduced leakage: the model could encounter structural patterns from the same environment in both training and test sets, making the test samples easier to predict and leading to inflated performance metrics. To prevent this, **all patches from a given environment are assigned exclusively to one split** (train, validation, or test). No environment appears in more than one split.

Near-duplicate frames.

Even after switching to Voronoi crossing points, some consecutive frames remain structurally very similar due to small robot displacements between crossings. Automated filtering using IoU and SSIM thresholds was applied as a first pass; however, pairs that were too similar to be reliably separated by these thresholds were removed through manual inspection. This step was essential to prevent artificially high similarity scores from being rewarded during training.

6.4.1 Static Dataset Split

The static pipeline uses $2{,}500$ environments for training and 250 for sampling (validation/test), yielding $97{,}050$ training patches and $23{,}375$ sampling patches at 32×32 resolution.

6.4.2 Global Map Completion Dataset Split

After quality filtering and environment-level split assignment, the global 128×128 dataset contains:

- **Train:** 3,200 map pairs
- **Validation:** 361 map pairs
- **Test:** 426 map pairs

Entire scene sequences remain in a single split to preserve temporal coherence. Near-duplicate frames (identified by visual inspection after SSIM/IoU filtering) were pruned prior to split assignment.

6.4.3 Frontier Patch Dataset Split

Two frontier patch dataset sizes are used in ablation experiments at 32×32 resolution:

- **Small Dataset (SDS):** 9,418 training patches (471 environments), 999 validation patches (60 environments), and 160 test patches (12 environments). The 160 test patches are a random subset of a larger 1,160-sample test pool (64 environments); this subset is held fixed for fair SDS/BDS comparison.
- **Large Dataset (BDS):** 10,418 training patches (524 environments); validation and test sets are identical to SDS.

A 64×64 variant was also generated to assess the effect of patch size on completion quality. Due to the larger patch footprint, the available patch count from the same source pool is reduced; generating a separate BDS at this resolution would have resulted in too few training samples to draw meaningful conclusions. A single dataset of 8,750 training patches (468 environments), 948 validation patches (60 environments), and 160 test patches was therefore used for the 64×64 experiments. This variant was used as an approximate indicator of patch size effect rather than a controlled comparison, since training and test set sizes differ from the 32×32 baseline.

6.5 Comprehensive Dataset Comparison

To highlight the complementary roles of the datasets developed in this study, Table 6.1 provides a side-by-side comparison. It contrasts the static dataset with the two output variants of the Gazebo-based pipeline: full sequential maps and localised frontier patches.

As shown in Table 6.1, the static dataset is best suited for controlled experiments, while Gazebo-based datasets capture realistic exploration conditions at both global and local scales. The environment-level split strategy, combined with near-duplicate pruning, ensures that reported performance metrics reflect genuine generalisation to unseen environments rather than memorisation of structural patterns seen during training.

Table 6.1 Comparison of static, full-map, and patch-based datasets

Aspect	Static Dataset	Gazebo Full Maps	Gazebo Patches
Source	PNG floorplans from HouseExpo	SLAM maps at t and $t{+}3$	Cropped patches around frontiers
Purpose	Controlled generative training with masks	Global map completion, forecasting	Local completion, extrapolation
Automation	Scripted, offline	Real-time robot navigation	Post-processing of logs
Simulation	None (image-based)	Requires Gazebo + ROS	Same as full maps
Map Dynamics	Static, ideal maps	Noisy, evolving maps	Local frontier-based dynamics
Masking	Artificial (e.g., bottom-half)	Natural from SLAM process	Frontier-based partiality
Patch Size	Whole map (320×320)	Full maps (128×128)	32×32, 64×64
Split	2,500 / 250 (train/test)	3,200 / 361 / 426 (train/val/test)	32×32: SDS 9,418/999/160; BDS 10,418/999/160; 64×64: 8,750/948/160
Split Strategy	Map-level	Environment-level, no leakage	Environment-level, no leakage
Use Case	Upper-lower completion, ablations	Global completion benchmarking	Frontier-centered local completion
Code	[5]	[4]	[4]

Table 6.2 summarizes the exact sample counts across all pipelines and splits. The static dataset yields the largest training set, while the frontier patch datasets are bounded by the number of extracted frontiers.

Table 6.2 Dataset statistics for all pipelines

Dataset	Train	Val	Test	Note
Static (32×32)	97,050	23,375		2,500 / 250 envs
Global (128×128)	3,200	361	426	env-level split
SDS Frontier (32×32)	9,418	999	160*	471 / 60 / 12 envs
BDS Frontier (32×32)	10,418	999	160*	524 / 60 / 12 envs
Frontier (64×64)	8,750	948	160†	468 / 60 / 12 envs

* Fixed 160-patch subset of 1,160-sample pool; identical across SDS and BDS.
† Single dataset; no BDS variant at this resolution due to insufficient patch count.

6.6 Summary

This chapter introduced two complementary dataset generation pipelines to support learning-based map completion in robotics. The first is a fast, static PNG-to-PGM conversion process that transforms 2D floorplans into clean and complete occupancy maps for use in controlled, idealised experiments.

The second pipeline is based on realistic, simulation-driven exploration using Gazebo and ROS, where a mobile robot autonomously navigates virtual environments. This approach yields two distinct dataset variants: (1) full occupancy maps saved at sequential time steps (t and $t{+}3$) for global completion tasks, and (2) localised input-output patch pairs extracted around frontier regions from the same maps, for fine-grained local map completion. Dataset quality is ensured through environment-level train/val/test splitting, Voronoi crossing-point selection, and near-duplicate pruning.

The resulting datasets—a $97{,}050$-patch static set (upper-to-lower task), a $3{,}200/361/426$ global map set, SDS/BDS frontier patch sets at 32×32 (approximately 9,400–10,400 training patches each), and a single frontier patch set at 64×64 (8,750 training patches)—provide a robust and scalable foundation for the model training and evaluation reported in Chapter 3.

Reproducibility

The dataset generation pipeline is publicly available in `dataset_generator.zip` at `https://github.com/f0120050/local_map_completion_and_integration_to_exploration`.
The generated frontier-centered patch datasets are available via Google Drive: the primary 32×32 dataset (SDS/BDS) at `https://drive.google.com/file/d/1rV2ugK4UFYJP9R7uqv4WzI-5I_keTrD5`, the 64×64 ablation dataset at `https://drive.google.com/file/d/18io9o68RZAzHbRWMLi9sOX0H9SZZkPqB`, and the full patch pool at `https://drive.google.com/file/d/1kuCWd1IrBpl13LxG6Ez3jB7UlpmzxQD3`.

7
EXPERIMENTS

This chapter evaluates map completion across four complementary dimensions and validates the end-to-end exploration system. The progression moves from isolated model comparison to integrated system validation:

1. **Section 7.1:** Rule-based geometric baseline establishing a training-free reference.
2. **Section 7.2:** Directional completion comparing CVAE and ImageGPT.
3. **Section 7.3:** Global map completion benchmarking six model families.
4. **Section 7.4:** Frontier-centered local completion under MASKED evaluation (primary model selection criterion).
5. **Section 7.5:** Final model selection for online exploration.
6. **Section 7.6:** Supplementary single-environment GT oracle navigation study.
7. **Section 7.7:** System-level exploration experimental setup.
8. **Section 7.8:** Quantitative exploration performance across eight environments.
9. **Section 7.9:** Synthesis of findings and deployment recommendations.

Implementation Details

All learning-based models are implemented in PyTorch. The Adam optimiser is used throughout with a reduce-on-plateau learning rate scheduler (`ReduceLROnPlateau`). Where Smooth-L1 loss is specified, `SmoothL1Loss` is used with $\beta = 0.1$. Unless otherwise noted, all model training was conducted on an Intel Core i9-13900HX CPU with 64 GB RAM and an NVIDIA RTX 4070 GPU.

Evaluation Protocol

Unless otherwise stated, model comparison follows a two-tier criterion. For cross-family comparison (Section 7.4), the primary selection score is the macro-average of $\mathrm{IoU}_{\mathrm{macro}}$ and $\mathrm{F1}_{\mathrm{macro}}$, supported by L1/RMSE, SSIM, and throughput. For within-family UNet variant selection (Section 7.5), the primary score is the unweighted average of five normalised metrics: Pixel Accuracy, F1-macro, IoU-macro, L1 (inverted), and normalised throughput. In both tiers, only models that meet the deployment throughput threshold (≥ 20 img/s) are eligible for selection; models below this threshold are reported for completeness but excluded from the selection ranking. The threshold is set to twice the $\approx$10 Hz planning frequency of the ROS navigation stack. In environments with many active frontiers, processing the full frontier set within a single planning cycle is necessary to avoid inference latency accumulating and disrupting the navigation loop; a safety factor of two over the planning frequency yields the 20 img/s threshold. For frontier-centered local completion, two evaluation protocols are maintained: **FULL** (all pixels) and **MASKED** (unknown cells only). MASKED evaluation is the primary criterion for this setting, as it measures genuine prediction quality over the regions the model must actually infer. FULL evaluation is also reported to allow comparison with models and studies that report full-patch metrics. For global completion, only FULL evaluation is reported, as this stage serves as an offline model family comparison rather than direct deployment; the MASKED criterion becomes primary again in the frontier-centered setting that follows.

7.1 Rule-Based Baseline: Geometric Map Completion

7.1.1 Experimental Setup

A total of 7,486 images from the HouseExpo dataset, produced using the static conversion pipeline described in Section 6.2, were used to test the proposed geometric map completion approach. Before evaluation, maps that were already complete were filtered out via manual inspection, resulting in 7,302 images for IoU calculation.

All experiments were conducted using Python and the OpenCV 4.5.5 library on a laptop with an Intel Core i7 processor and 16 GB RAM. Each map was first checked for orientation and corrected if rotated using bounding box-based alignment to ensure consistent horizontal and vertical wall detection.

For door discovery, predefined thresholds were applied to wall and door pixel counts along scanlines, with sensitivity tested using vertical thresholds of 125 and

100. Edge detection using the Canny method [42] was applied, followed by guide line extraction using the Hough transform [43] and clustering to identify structural directions. IoU evaluation was performed within the region of interest (RoI) defined by the bounding box of each partial map. Binary completion results were compared against ground truth full maps cropped to the same RoI.

7.1.2 Quantitative Results

We use Intersection over Union (IoU) [54] as the primary metric. Both predictions and reference maps are converted to binary format. The RoI is defined by the bounding box of each partial input map. Table 7.1 presents the average IoU scores across 7,302 test images from 123 distinct environments.

Table 7.1 IoU results for the rule-based geometric completion algorithms across 7,302 HouseExpo test maps

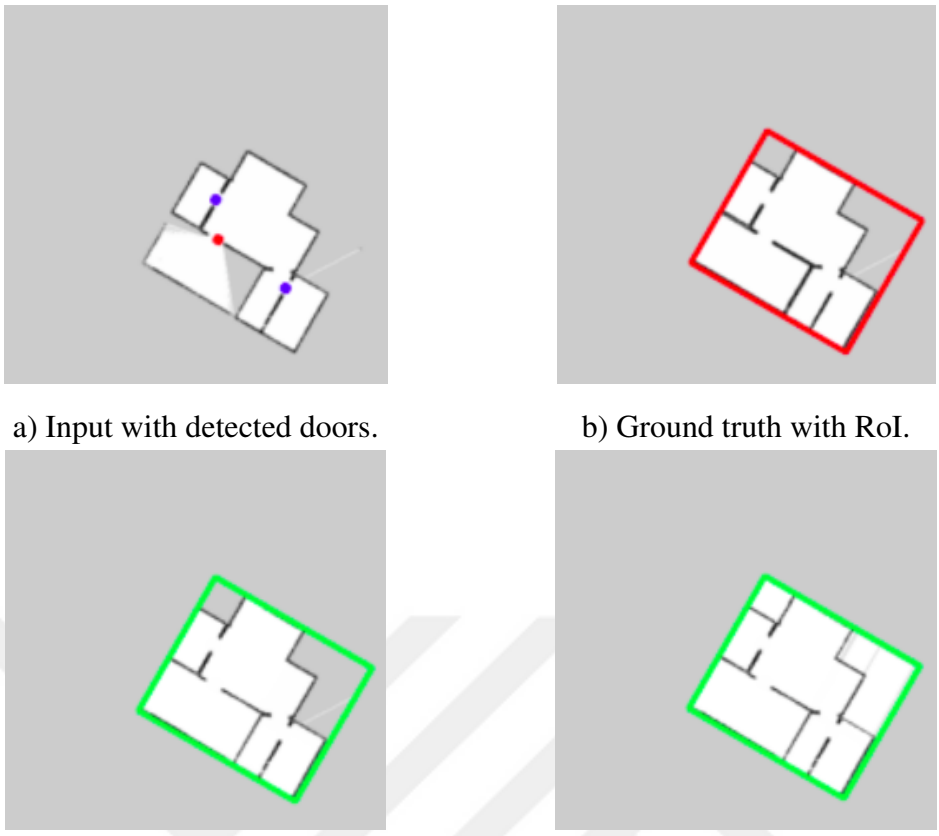
	Algorithm 1	**Algorithm 2**
Mean IoU	0.8790	0.8784
Maximum	0.9999	0.9993
Minimum	0.4330	0.4330
Standard Deviation	0.1023	0.1020

Algorithm 1 consistently yields more accurate results with a higher true positive rate, but produces more cautious completions bounded by nearby guide lines. Algorithm 2 achieves broader coverage at the cost of more false positives. Despite the high IoU values, these scores are computed only within the RoI; since most areas outside are already known, the results may not fully reflect performance in genuinely unknown regions. The learning-based methods introduced in subsequent sections address this limitation.

7.1.3 Qualitative Results

Figure 7.1 presents a representative example highlighting the behavioural differences between the two algorithms. The input map contains detected door locations, while the ground truth defines the correct region of interest. Algorithm 1 produces a structured completion that closely follows the room layout and avoids extending into uncertain areas. In contrast, Algorithm 2 expands the completion more aggressively, which may lead to overestimation of free space.

Figure 7.2 illustrates a complementary case in which Algorithm 2 performs better. The input map contains limited structural gaps, and the ground truth indicates a more compact completion region. In this scenario, Algorithm 1 remains overly

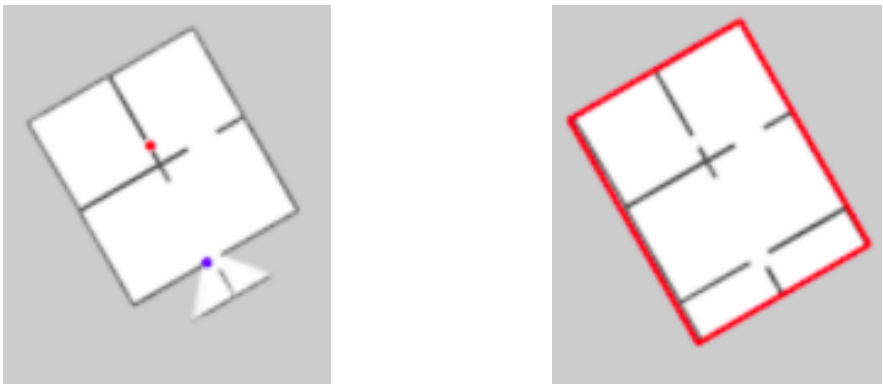


a) Input with detected doors. b) Ground truth with RoI.

c) Algorithm 1 output. d) Algorithm 2 output.

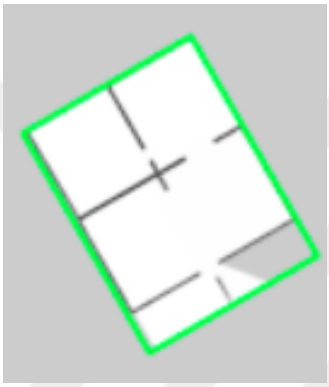
Figure 7.1 Comparison on a partially observed environment: (a) detected door locations, (b) ground truth region of interest, (c) completion produced by Algorithm 1, and (d) completion produced by Algorithm 2.

conservative and fails to fully recover the expected structure, whereas Algorithm 2 successfully fills the region due to its more permissive expansion strategy.

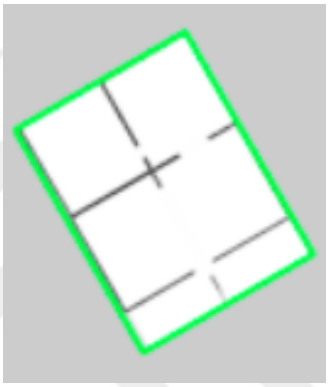


a) Input with detected doors.

b) Ground truth RoI.



c) Algorithm 1 output.



d) Algorithm 2 output.

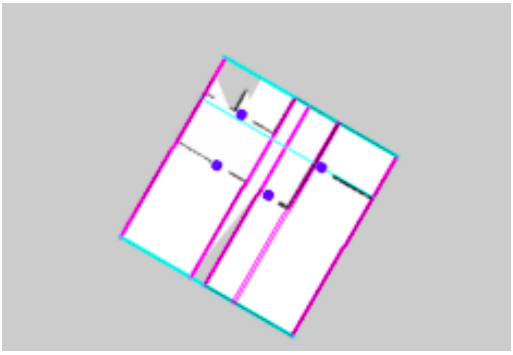
Figure 7.2 Comparison on a scenario with limited empty space inside the bounding box: Algorithm 2 provides a more complete reconstruction than Algorithm 1.

Impact of Threshold

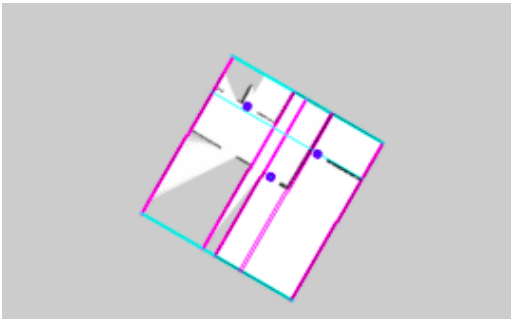
The proposed algorithms are sensitive to threshold selection. Figure 7.3 demonstrates the impact of changing the minimum vertical door detection threshold. A threshold of 125 correctly identifies the door and enables successful room completion. Reducing the threshold to 100 causes incorrect door detection in this case, preventing the room from being completed.

Limitations

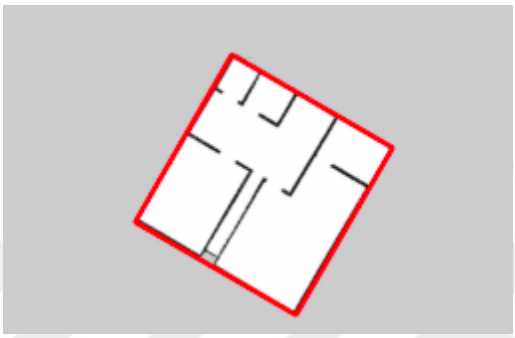
Both algorithms share a fundamental limitation: they are designed for axis-aligned environments and have no mechanism to handle diagonal or curved wall geometries. When non-perpendicular walls are present, the guide-line extraction fails to follow the actual geometry, causing both algorithms to produce incorrect or absent completions. Figure 7.4 illustrates this case.



a) Threshold 125: door detected, room completed.

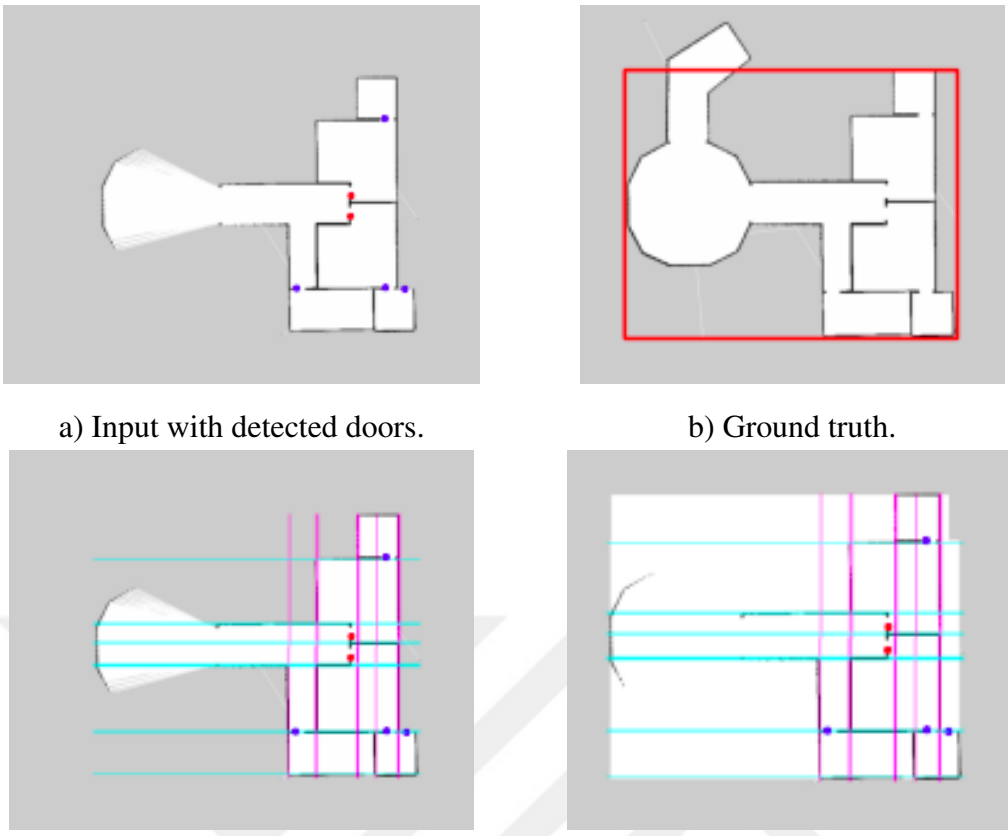


b) Threshold 100: door missed, room not completed.



c) Ground truth.

Figure 7.3 Impact of threshold selection on completion performance. A higher threshold (125) detects the door more precisely and enables correct room completion, while a lower threshold (100) misses the door in this case and leaves the room undetected.



a) Input with detected doors.

b) Ground truth.

c) Algorithm 1 with guidelines.

d) Algorithm 2 output.

Figure 7.4 Limitation: non-axis-aligned walls cannot be handled by either algorithm. The guide-line extraction fails to follow the diagonal geometry, resulting in incorrect or absent completions.

7.2 Upper-to-Lower Directional Local Completion

7.2.1 Experimental Setup

This section evaluates the *directional completion* formulation introduced in Chapter 3: the upper half of each 32×32 patch is treated as the observed condition and the model predicts the lower half. The static dataset (Section 6.2) is used: 2,500 maps for training and 250 for sampling, yielding 97,050 training patches and 23,375 test patches.

All models were trained for 250 epochs on a laptop with an Intel Core i9-13900HX CPU, 64 GB RAM, and an NVIDIA RTX 4070 GPU (batch size 25).

Two models are compared: (i) **ImageGPT** [7] (autoregressive transformer over discrete pixel tokens), and (ii) **CVAE** [44, 45] (fast decoder-based conditional prediction). Since only the lower half is unknown by design, all metrics are computed over the generated lower region.

Training duration: ImageGPT required ≈ 3.5 min per epoch; CVAE required ≈ 0.4 min. Inference: ImageGPT ≈ 0.1 min per patch; CVAE ≈ 0.00003 min (direct decoding, encoder bypassed). For CVAE, the β parameter was tuned over $\{0, 0.25, 0.5, 0.75, 1.0\}$; $\beta = 0.5$ gave the best performance and was used for all reported results.

7.2.2 Quantitative Results

Table 7.2 reports IoU, SSIM, F1-Score, Accuracy, Precision, Recall, and FID for both models, broken down by orientation angle [1].

Table 7.2 Upper-to-lower completion results for ImageGPT and CVAE across four orientation angles ($0°, 30°, 60°, 90°$) and their average (best average values in bold)

Model	Angle	IoU↑	SSIM↑	F1↑	Acc↑	Prec↑	Rec↑	FID↓
ImageGPT	0°	0.904	0.608	0.458	0.582	0.555	0.458	66.473
	30°	0.873	0.532	0.495	0.662	0.587	0.534	127.938
	60°	0.875	0.503	0.485	0.640	0.543	0.534	122.718
	90°	0.865	0.540	0.455	0.593	0.533	0.471	67.550
	Avg	0.879	0.546	0.473	0.619	0.555	0.499	96.670
CVAE	0°	0.906	0.543	0.774	0.860	0.838	0.759	30.456
	30°	0.937	0.612	0.837	0.912	0.860	0.826	27.993
	60°	0.947	0.635	0.841	0.923	0.864	0.831	24.768
	90°	0.941	0.707	0.857	0.917	0.877	0.857	32.460
	Avg	**0.933**	**0.624**	**0.827**	**0.903**	**0.860**	**0.818**	**28.919**

CVAE outperforms ImageGPT on all metrics. The FID gap (28.9 vs. 96.7) indicates that CVAE produces completions whose statistical distribution is closer to the ground truth. The inference time advantage is decisive for online use (≈ 1.8 ms vs. ≈ 6 s per patch), representing a $> 3000\times$ advantage. CVAE is therefore selected as the best model for the directional completion task.

However, this formulation assumes a fixed directional prior (upper $\rightarrow$ lower) and is not directly equivalent to frontier-centered local completion, where the unknown region may occupy any part of the patch. Although the formulation could be extended to other orientations by rotating patches prior to inference, doing so would introduce additional preprocessing overhead and would still not fully address the irregularly shaped unknown regions characteristic of frontier-centered patches, where the boundary between known and unknown cells follows no fixed axis. For this reason, the frontier-centered setting is evaluated separately in Section 7.4.

7.2.3 Qualitative Results

Figures 7.5 and 7.6 present successful predictions across four orientations ($0°, 30°, 60°, 90°$). In each column, the left patch is the ground truth, the centre shows the input (upper half only), and the right displays the completed patch. Figure 7.7 shows failure cases for both models.

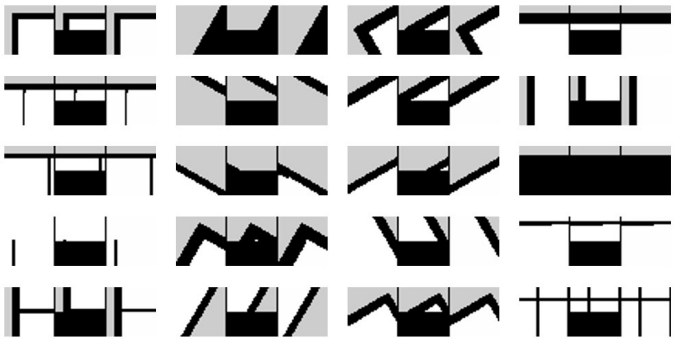


Figure 7.5 Successful upper-to-lower completion results from ImageGPT across four orientation angles.

7.3 Global Map Completion

This section evaluates *global* completion: predicting a more complete 128×128 occupancy map from a partially observed input at time t, with the target being the map at time $t + 3$. Unlike the directional formulation, the unknown region corresponds to exploration-induced unobserved areas distributed across the map.

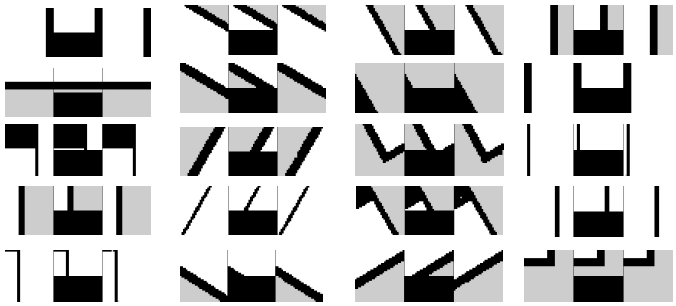
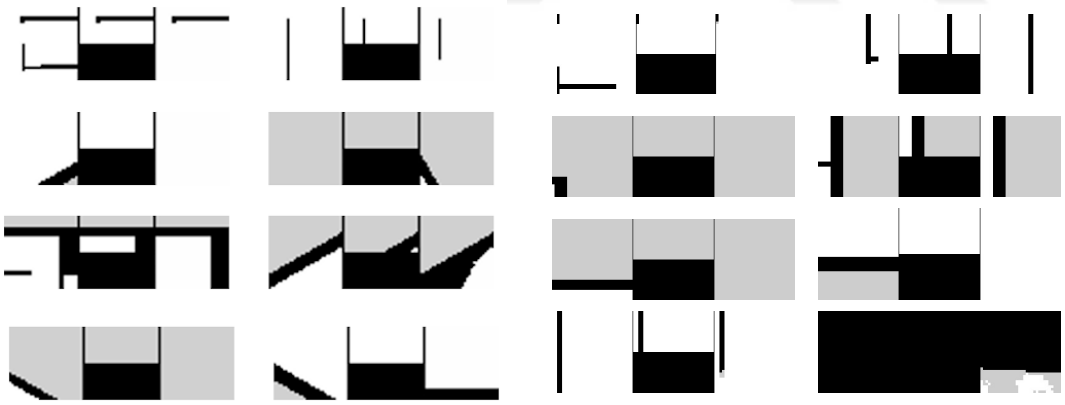


Figure 7.6 Successful upper-to-lower completion results from CVAE across four orientation angles.



Failure cases from ImageGPT.

Failure cases from CVAE.

Figure 7.7 Failure cases from ImageGPT (left) and CVAE (right). First columns: lack of guiding structure in input. Second columns: shape inconsistencies or unrealistic continuations.

7.3.1 Experimental Setup

Six model families are compared [2]: CVAE, CGAN (pix2pix-hinge) [46], RePaint diffusion [48], ViT encoder–decoder [49], residual UNet [50], and the training-free GEOM baseline (Chapter 3). The Gazebo/ROS sequential dataset (Section 6.4.2) is used: 3,200/361/426 train/val/test maps. All models receive single-channel 128×128 partial maps as input. Adam optimiser is used with batch size 32, a reduce-on-plateau learning rate scheduler [55] (patience 6), and seed 42. LaMa is evaluated as a standalone model in the frontier-centered setting (Section 7.4); it is not included in the global comparison as its inpainting-oriented architecture targets a different prediction regime than the sequential map prediction task here.

7.3.2 Quantitative Results

Table **??** reports test-set results under FULL evaluation (all pixels). Among the feedforward models (CVAE, CGAN, ViT, UNet), all exceed 20 img/s throughput. RePaint runs at approximately 220 ms per map (50 DDIM steps), and the training-free GEOM runs on CPU at approximately 52 ms per map.

captionGlobal map completion FULL evaluation results for 128×128 (best values per column in bold)

Model	**L1↓**	**MSE↓**	**RMSE↓**	**SSIM↑**	**UQI↑**	**ERGAS↓**	**IoU↑**	**F1↑**	**Acc↑**
CVAE	0.0672	0.0324	0.1795	0.7486	0.7856	25.00	0.702	0.815	0.882
CGAN	0.0649	0.0367	0.1911	0.7519	0.7738	26.64	0.716	0.828	0.882
RePaint	0.0718	0.0312	0.1760	0.8179	0.8120	24.50	0.772	0.869	0.901
ViT (16)	0.0812	0.0344	0.1850	0.7168	0.7672	25.76	0.657	0.775	0.873
UNet (3+3)	**0.0453**	**0.0262**	**0.1610**	0.8279	**0.8366**	**22.43**	**0.796**	**0.884**	**0.916**
GEOM (CPU)	0.0529	0.0287	0.1686	**0.8484**	0.8153	23.50	0.779	0.874	0.898

UNet (3+3) achieves the best aggregate accuracy across all pixel-wise and discrete metrics. GEOM attains the highest SSIM (0.8484) despite requiring no training, demonstrating the power of explicit structural priors under rectilinear layouts. RePaint provides strong perceptual quality (SSIM 0.8179) but at a 4–5$\times$ throughput penalty compared to feedforward models. ViT underperforms UNet by a 14% IoU gap, attributed to coarse 16×16 tokenisation that produces discontinuous predictions for thin walls at this scale rather than smooth boundary recovery. CGAN achieves slightly better IoU than CVAE (0.716 vs. 0.702) but falls behind UNet and RePaint; its adversarial training adds useful sharpness at the cost of some structural stability and occasional hallucinated geometry. CVAE obtains the lowest IoU among generative models at this scale, likely because its latent-space sampling introduces spatial smoothing that is better tolerated in the directional completion setting than in the full 128×128 prediction task.

The global results confirm the model ranking established in Chapter 3 and motivate the reuse of the same families for frontier-centered local completion.

7.3.3 Qualitative Results

Figure 7.8 shows representative qualitative results across all six model families.

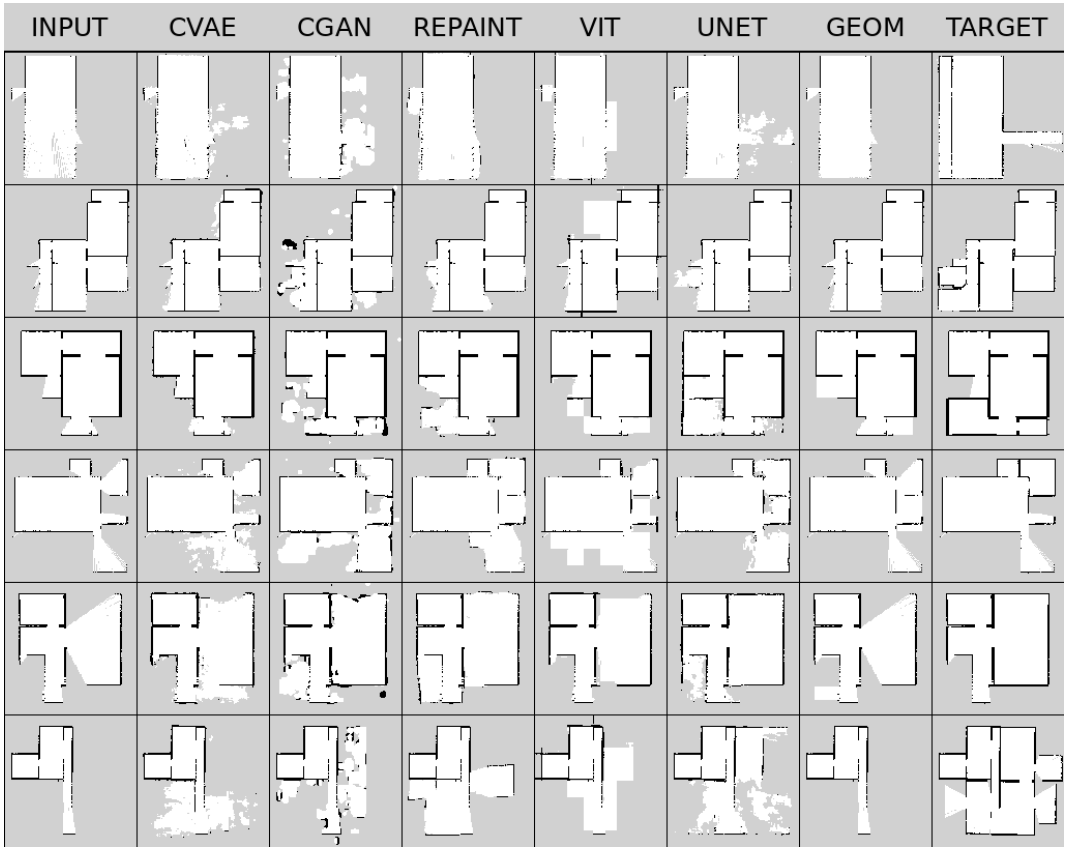


Figure 7.8 Qualitative global completion examples. Columns left to right: input, CVAE, CGAN, RePaint, ViT, UNet, GEOM, target.

UNet produces the sharpest wall boundaries with the most accurate structural recovery. GEOM competes well under regular rectilinear layouts where its door-detection heuristics are reliable. RePaint occasionally over- or under-completes uncertain regions due to the stochastic nature of its denoising process. CGAN produces generally coherent outputs but introduces hallucinated structures in ambiguous areas. ViT generates globally plausible completions but struggles to maintain thin wall continuity, resulting in fragmented or blocky wall representations due to its coarse patch tokenisation. CVAE produces smooth but spatially blurred completions that underestimate fine structural details.

7.4 Frontier-Centered Patch-Based Local Map Completion

This section evaluates the frontier-centered local completion formulation: given a 32×32 patch extracted from a SLAM map and centred on a frontier, the model predicts the unknown region. Although this formulation reflects the patch structure used during online exploration, the evaluation here is conducted offline on held-out test patches. System-level online evaluation is reported in Section 7.8.

7.4.1 Experimental Setup

The dataset configuration, evaluation protocol, and model set used in this section are described below.

Dataset and Evaluation Protocol

The frontier patch dataset (Section 6.4.3) is used under two regimes: **SDS** (small training set) and **BDS** (larger training set). Both regimes share the same held-out test set, enabling direct comparison of model rankings under different amounts of training data. The BDS provides a larger training set drawn from additional environments; the goal was to determine whether a small data augmentation would change model rankings. Since no ranking change was observed—UNet remained top under both regimes—further dataset expansion was not pursued given the time and resource constraints of the collection process; UNet variants were subsequently compared on SDS. Patches from the same environment are assigned exclusively to one split (environment-level, leakage-free).

The dataset splits and patch counts used in all experiments are summarised in Table 6.2 (Chapter 6).

Two evaluation scopes are reported:

- **FULL**: metrics over the entire patch. Reported for compatibility with prior work.
- **MASKED**: metrics only over cells that were unknown in the input. **This is the primary criterion.**

Primary selection score: macro-average of $\mathrm{IoU}_{\mathrm{macro}}$ and $\mathrm{F1}_{\mathrm{macro}}$ on the masked region (cross-family tier), subject to a deployment throughput threshold of $\geq 20\,\mathrm{img/s}$.

It is important to note the distinction between offline and online evaluation in this chapter. All frontier-centered models are trained and evaluated offline on the static patch dataset described in Chapter 6: model selection is based entirely on held-out test set MASKED metrics. The selected model (residual U-Net) is subsequently deployed in the live exploration pipeline, where its effect on exploration efficiency is measured online across eight environments (Section 7.8). These are two independent evaluation stages; the offline metrics inform model selection, while the online experiments measure system-level impact on exploration efficiency rather than online patch completion quality.

Model Set

All models evaluated for global completion are also evaluated here: UNet variants, CVAE [44, 45], ViT [49], Pix2Pix-hinge CGAN [46], RePaint [48]. Additional local models include: LaMa [51] (architecturally related to the encoder–decoder family, but included in the cross-family comparison rather than the UNet variant table owing to its widespread independent recognition as a large-mask inpainting model; its throughput of 12.67 img/s places it below the deployment threshold), Seg-GAN hinge, ImageGPT-style local baseline [7], and row-wise LSTM. Multiple U-Net variants are also evaluated: attention-gated (UNet_{att}), 4-layer deeper (UNet_{4L}), enhanced configuration with Dice loss, dynamic class weighting, and augmentation (UNet_{new}).

7.4.2 Quantitative Results

7.4.2.1 SDS MASKED Evaluation

Table 7.3 reports MASKED evaluation on SDS. Models below 20 img/s are excluded from deployment selection; among the remaining models, selection is based on the macro-average of $\mathrm{F1}_{\mathrm{macro}}$ and $\mathrm{IoU}_{\mathrm{macro}}$.

$\texttt{UNet}_{\texttt{base}}$ achieves the highest average of $\mathrm{F1}_{\mathrm{macro}}$ and $\mathrm{IoU}_{\mathrm{macro}}$ (0.364) among deployment-eligible models, marginally ahead of Pix2Pix-hinge (0.363). Note that LaMa achieves a higher average (0.373) but is excluded from deployment consideration due to its throughput of 12.67 img/s. Models such as RePaint and ViT are known to achieve competitive results in the literature when trained on large datasets; however, under the small dataset setting, their data-hungry nature manifests as reduced MASKED performance relative to the residual U-Net. ViT's attention mechanism requires substantial data to learn meaningful spatial priors for thin-wall localisation; furthermore, its coarse patch tokenisation causes thin wall structures to fall across token boundaries, resulting in discontinuous and

Table 7.3 Frontier-centered local completion on SDS, MASKED evaluation, where † = excluded (throughput < 20 img/s)

Model	F1-mac↑	IoU-mac↑	Avg↑	Notes
$\text{UNet}_{\text{base}}$	**0.412**	**0.316**	**0.364**	Best overall
Pix2Pix-hinge	0.409	0.316	0.363	Strong GAN
Seg-GAN-hinge	0.371	0.276	0.324	
ViT	0.319	0.226	0.273	Needs more data
CVAE	0.291	0.201	0.246	Fails locally
LSTM_{row}	0.094	0.052	0.073	Not suitable
GEOM	0.000	0.000	0.000	Rule-based
$\text{LaMa}^{\dagger}$	0.428	0.318	0.373	12.67 img/s
RePaint	0.111	0.062	0.087	Low MASKED score
$\text{ImageGPT}^{\dagger}{}_{\text{local}}$	0.225	0.137	0.181	$\approx$0.05 img/s

fragmented wall predictions rather than smooth geometric recovery. RePaint's iterative denoising introduces high variance under limited training signal, making it unsuitable despite passing the throughput threshold. Furthermore, RePaint's sampler re-imposes known pixels at every denoising step, making its FULL metric partially inflated by trivially correct observed cells—a blending effect absent in the UNet and GAN variants. LaMa's large-mask inpainting architecture provides strong global Fourier context, but this global reasoning is less suited to the localised 32×32 patch regime where fine-grained wall structure dominates; it also incurs a throughput penalty from its FFT operations. GAN variants (Pix2Pix-hinge, Seg-GAN) are competitive but exhibit training instability and occasional hallucinated geometry at this scale. CVAE fails under irregular-mask conditions: its latent encoding, shaped during training to handle structured directional masks, does not generalise to the frontier-centered setting where the unknown region takes arbitrary shapes. LSTM and ImageGPT cannot exploit 2D spatial structure and are unsuitable for this task.

7.4.2.2 SDS FULL Evaluation

Table 7.4 presents FULL metrics alongside throughput. Models below 20 img/s (LaMa: 12.67, $\text{ImageGPT}_{\text{local}}$: $\approx$0.05) are excluded from deployment selection.

7.4.2.3 Effect of Dataset Size (SDS → BDS)

Tables 7.5 and 7.6 show the effect of the larger dataset. UNet performance is nearly unchanged between SDS and BDS, confirming that frontier-centered local completion is *geometry-dominant* rather than data-hungry. Models with strong 2D convolutional inductive biases are relatively insensitive to moderate increases in

Table 7.4 Frontier-centered local completion on SDS, FULL evaluation with throughput

Model	IoU-mac↑	F1-mac↑	L1↓	RMSE↓	SSIM↑	img/s↑
$\text{UNet}_{\text{base}}$	**0.643**	**0.776**	**0.090**	0.232	0.645	42.25
LaMa	0.633	0.767	0.097	0.246	**0.661**	12.67
Seg-GAN-hinge	0.592	0.734	0.105	0.237	0.604	42.21
Pix2Pix-hinge	0.586	0.729	0.107	0.247	0.621	**43.59**
ViT	0.551	0.695	0.119	**0.231**	0.600	41.72
GEOM	0.566	0.718	0.122	0.257	0.644	41.67
CVAE	0.511	0.662	0.133	0.255	0.559	43.06
LSTM_{row}	0.481	0.620	0.153	0.316	0.543	39.38
RePaint	0.484	0.628	0.194	0.327	0.504	35.43
$\text{ImageGPT}_{\text{local}}$	0.534	0.680	0.139	0.301	0.555	≈0.05

training data, though substantially larger datasets may still yield marginal gains. For models such as LaMa that are known to scale with dataset size in natural image domains, future work with substantially larger frontier patch datasets may reveal whether data-hungry models can eventually close the gap with UNet. RePaint's iterative denoising yields near-zero MASKED scores under both dataset sizes (SDS: 0.087, BDS: 0.051), confirming that it is unsuitable for this task regardless of training data volume.

Table 7.5 Frontier-centered local completion on BDS, MASKED evaluation

Model	Pixel Acc↑	F1-mac↑	IoU-mac↑	L1↓
$\text{UNet}_{\text{base}}$	**0.619**	**0.409**	**0.309**	**0.206**
Pix2Pix-hinge	0.613	0.384	0.289	0.223
ViT	0.570	0.323	0.242	0.236
CVAE	0.404	0.267	0.180	0.317
RePaint	0.102	0.094	0.051	0.591

GEOM is included in the FULL evaluation for completeness but is excluded from the MASKED comparison: GEOM produces near-zero MASKED scores in the frontier-centered setting (IoU $= 0.000135$).

7.4.2.4 Effect of Patch Size (32×32 vs 64×64)

Table 7.7 compares UNet on 32×32 versus 64×64 patches across two dataset sizes.

The 64×64 variant shows a moderate IoU drop (≈0.05–0.09 points) under the current dataset size. This is attributed to two compounding factors: the larger patch captures a wider spatial context and extracts richer features, but simultaneously

Table 7.6 Frontier-centered local completion on BDS, FULL evaluation

Model	Pixel Acc↑	F1-mac↑	IoU-mac↑	L1↓	RMSE↓
UNet$_{\text{base}}$	**0.8455**	**0.7834**	**0.6524**	**0.0860**	**0.2933**
Pix2Pix-hinge (ep700)	0.8145	0.7240	0.5829	0.1051	0.3242
ViT	0.7994	0.7005	0.5558	0.1185	0.3443
GEOM	0.7665	0.7175	0.5663	0.1222	0.3496
CVAE	0.7596	0.6575	0.5071	0.1359	0.3686
RePaint	0.7210	0.6072	0.4642	0.2016	0.4490

Table 7.7 Effect of patch size on UNet performance (MASKED evaluation)

Patch	DS	Pixel Acc↑	F1↑	IoU↑	L1↓	RMSE↓
32×32	SDS	≈0.62	≈0.41	≈0.31	≈0.19	≈0.33
32×32	BDS	≈0.62	≈0.41	≈0.31	≈0.21	≈0.34
64×64	SDS	0.452	0.324	0.225	0.287	0.396
64×64	BDS	0.558	0.373	0.274	0.237	0.366

exposes a substantially larger unknown region that the model must predict. The 64×64 evaluation also uses a test set spanning only 4 environments, since the environments were not independently sampled for each resolution—patches at different sizes were drawn from the same pool of exploration runs, and a controlled resolution-matched comparison would require independent dataset construction. The observed IoU drop therefore reflects both the data-volume constraint and reduced evaluation diversity; a controlled comparison with independently sampled, resolution-matched splits would be needed to isolate the effect of patch size alone.

Regarding the relationship between offline patch-size results and exploration performance: the GT oracle study (Section 7.8.1.2) demonstrates that in the `4bb74477` environment, using a 64×64 oracle during exploration recovers a substantial advantage over the 32×32 oracle. This shows that the topological information necessary to guide frontier selection in that environment lies beyond the 32×32 window, and that increasing the prediction window size—if accompanied by sufficient training data—could yield system-level gains. This is a separate question from whether the UNet's offline accuracy at 64×64 would exceed its accuracy at 32×32; the oracle study establishes the potential exploration value of wider context, while the offline comparison characterises the current model's prediction quality at each scale.

7.4.3 Qualitative Results

The visual performance of the evaluated models across the frontier-centered local completion task is illustrated in Figures 7.9, 7.10, and 7.11. These qualitative results highlight the structural reasoning capabilities and characteristic failure modes of each model when faced with irregular unknown regions.

As shown in Figure 7.9, the **residual U-Net** demonstrates a highly consistent ability to preserve thin wall structures and maintain rectilinear geometry, producing completions that closely align with the ground-truth targets.

The **ViT** model produces globally plausible structures, but its coarse patch tokenisation causes thin walls to fall across token boundaries, resulting in discontinuous and fragmented wall predictions rather than smooth geometric recovery. The **Pix2Pix** model generates coherent and relatively sharp outputs, but may struggle with precise geometric alignment, especially in structurally ambiguous regions, occasionally producing hallucinated structures.

The **RePaint** diffusion model introduces noticeable noise artifacts in uncertain regions, often resulting in fragmented or speckled structures. The **CVAE** exhibits unstable behavior under irregular masks: in some cases it produces plausible completions, while in others it fails to capture the underlying structural pattern or generates hallucinated geometry that does not correspond to the true layout.

The training-free **GEOM** baseline relies on door detection and structural inference. While effective in larger contexts where door openings are clearly observable, it struggles in small frontier-centered patches where such cues are often absent or cropped out. As a result, it frequently fails to complete the structure in these localised settings.

The limitations of non-spatial and sequential modelling approaches are further illustrated in Figure 7.10. The **LSTM** baseline fails to capture 2D spatial dependencies, resulting in fragmented and discontinuous structures that lack physical plausibility. Similarly, the **ImageGPT** model struggles with spatial coherence, producing noisy and geometrically inconsistent predictions despite capturing some local patterns.

The **LaMa** model leverages global frequency-based context and produces visually coherent completions; however, it may fail to reconstruct thin wall structures at the 32×32 scale and, in some cases, generates hallucinated regions or incomplete geometric patterns where its global context provides insufficient local guidance.

Finally, a comparison of **U-Net variants** is provided in Figure 7.11. While all variants—$\text{UNet}_{\text{base}}$, UNet_{new}, UNet_{att}, and UNet_{4L}—produce structurally consistent completions, the differences between them remain minor. Architectural modifications such as attention gates (UNet_{att}) or increased depth (UNet_{4L}) do not yield a consistent qualitative improvement over $\text{UNet}_{\text{base}}$, reinforcing the observation that the frontier-centered completion task is primarily geometry-dominant and well captured by convolutional models.

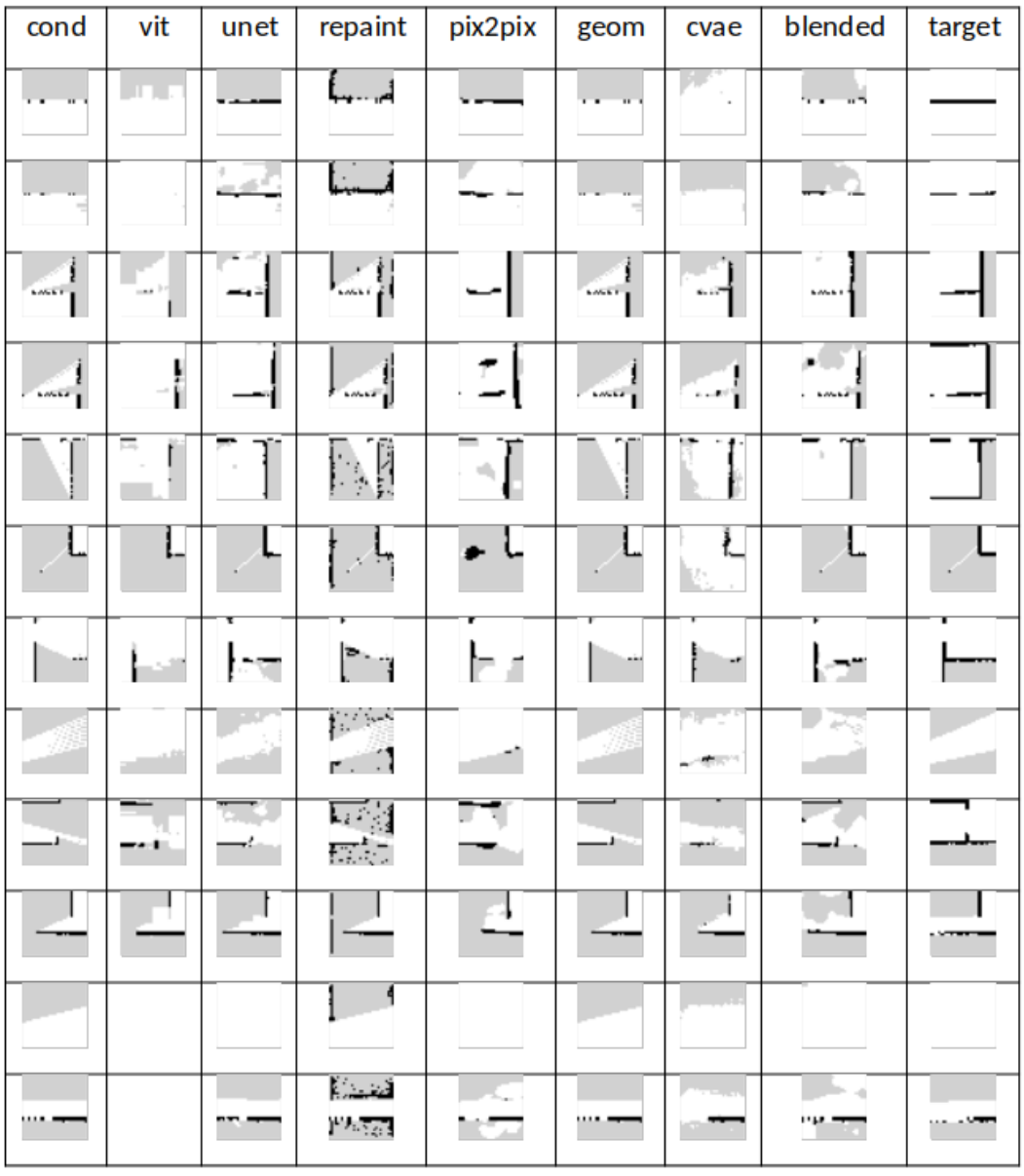


Figure 7.9 Qualitative comparison of primary model families on the SDS test set. Columns left to right: input condition (cond), ViT, UNet, RePaint, Pix2Pix, GEOM, CVAE, blended result, and ground-truth target.

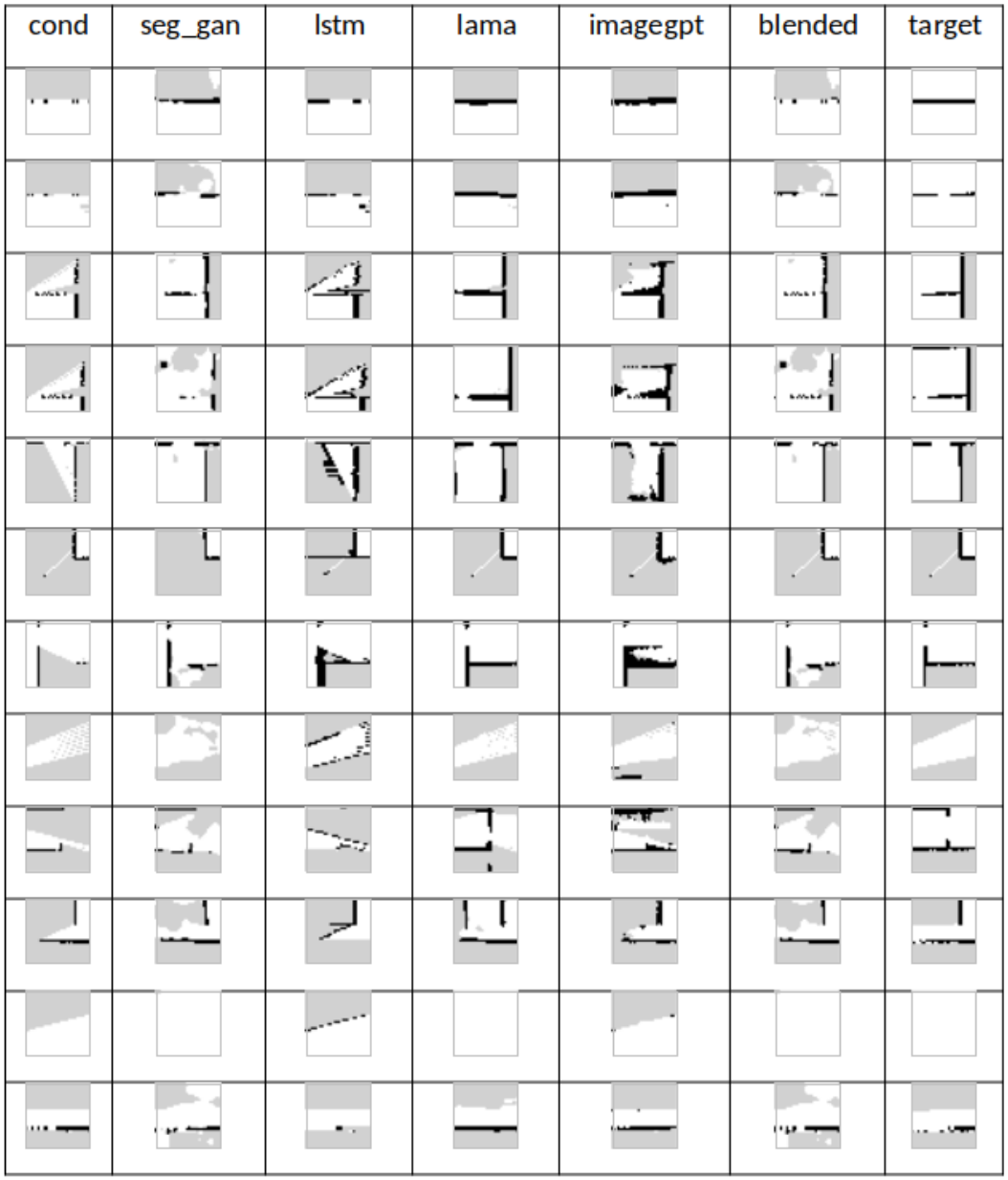


Figure 7.10 Qualitative results for GAN-based and sequential model variants. Columns: input, Seg-GAN, LSTM, LaMa, ImageGPT, blended, and target.

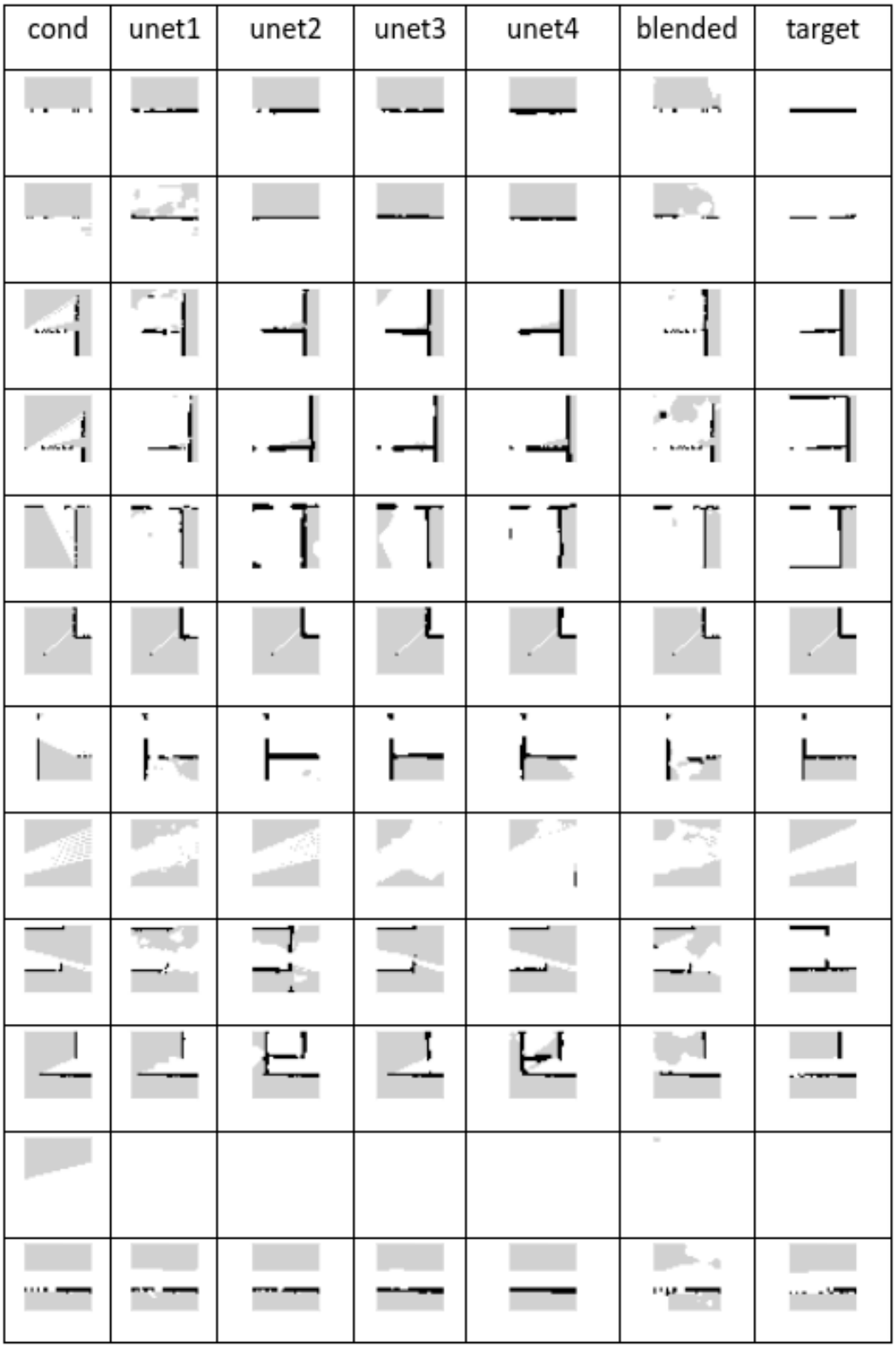


Figure 7.11 Qualitative comparison across U-Net architecture variants on the SDS test set. Columns left to right: input condition (cond), $\text{UNet}_{\text{base}}$, UNet_{new}, UNet_{att}, UNet_{4L}, blended result, and ground-truth target. All configurations produce structurally consistent maps, with marginal differences in obstacle recall.

7.5 Final Model Selection for Online Exploration

Under the constraints of masked-region prediction quality and online inference feasibility, UNet-family models provide the strongest trade-off across all evaluation settings and are selected as the primary local completion module. Model selection proceeds in two tiers. First, among all model families, those achieving at least 20 img/s are ranked by the macro-average of MASKED $\mathrm{IoU}_{\mathrm{macro}}$ and $\mathrm{F1}_{\mathrm{macro}}$; this eliminates LaMa (12.67 img/s) and ImageGPT$_{\mathrm{local}}$ ($\approx$0.05 img/s) from deployment consideration on throughput grounds; RePaint (35.43 img/s) passes the throughput threshold but is excluded from deployment due to its iterative denoising cost and high variance under limited training data; the remaining eligible models identify the UNet family as the top-performing group (avg. 0.364). Second, among UNet variants (Table 7.8), the deployment candidate is chosen by the unweighted five-metric normalised average (Pixel Accuracy, F1-macro, IoU-macro, L1 inverted, normalised throughput); UNet$_{\mathrm{base}}$ achieves the highest composite score (0.627) and is selected. Models below 20 img/s are excluded from deployment consideration (†).

Table 7.8 UNet variant comparison on SDS, MASKED evaluation (best values in bold)

Model	Pixel Acc↑	F1-mac↑	IoU-mac↑	L1↓	img/s↑	Avg↑
UNet$_{\mathrm{base}}$	**0.649**	0.412	0.316	**0.192**	42.25	**0.627**
UNet$_{\mathrm{new}}$	0.571	**0.444**	**0.336**	0.233	39.94	0.603
UNet$_{\mathrm{att}}$	0.587	0.441	0.333	0.228	39.33	0.603
UNet$_{\mathrm{4L}}$	0.562	0.441	0.334	0.237	**43.63**	0.616
LaMa†	0.541	0.428	0.318	0.250	12.67	0.464

UNet$_{\mathrm{base}}$ achieves the highest normalised five-metric composite score (0.627), driven by its superior Pixel Accuracy (0.649) and L1 (0.192), despite slightly lower IoU than deeper variants. Although UNet$_{\mathrm{new}}$, UNet$_{\mathrm{att}}$, and UNet$_{\mathrm{4L}}$ achieve marginally higher IoU and F1 values individually, the composite score—which accounts for throughput and penalises higher L1—favours the base configuration. LaMa, while architecturally related to the UNet family, is treated as a standalone model in the cross-family comparison and excluded here due to its throughput. The practical performance of the selected $\texttt{UNet}_{\texttt{base}}$ is validated in Section 7.8.

7.6 GT Oracle Navigation Study: Supplementary Single-Environment Analysis

The model-comparison experiments above establish which completion model best predicts unknown regions. This section reports a supplementary GT oracle navigation study [8] that investigates how much navigation improvement is achievable under ideal local completion conditions, by injecting ground-truth patches directly into the navigation map.

7.6.1 Motivation and Oracle Framework

During exploration development, the navigation stack was observed to repeatedly attempt routing through spurious wall gaps in the SLAM map, and to pursue frontiers generated beyond walls by SLAM noise. It was hypothesised that if these failure modes were addressed—for instance, by using completed maps that fill the gaps—the navigation stack would operate more efficiently, and the effect of map completion on navigation could be measured more cleanly, independently of these stochastic artifacts.

Rather than proposing a new completion model, this study treats map completion as a modular capability and uses an aligned ground-truth occupancy map as an oracle. When a frontier is selected, a 32×32 patch centred on the frontier is extracted from the GT map and fused into the navigation map, which is published as a separate completed-map topic consumed by the navigation stack via topic remapping. The exploration node continues to operate on the raw SLAM map.

This design enables model-agnostic evaluation and provides an observation of the potential navigation improvement achievable under ideal local completion conditions in a single environment.

It is important to contextualise this study within the broader thesis: the GT oracle navigation experiment is not a target contribution of this work but rather a complementary analysis conducted to establish a theoretical reference point for navigation improvement under perfect local completion. The primary performance gains observed in this study are attributable not solely to map completion itself, but largely to two specific failure modes in the baseline navigation stack: repeated path planning attempts through spurious wall gaps in the SLAM map, and the pursuit of unreachable exterior frontiers. Improving the `explore_lite` blacklist mechanism—for example, by detecting forward–backward oscillation using heading variance or displacement tracking in addition to timeout—and tuning global planner parameters may to some degree reduce these failure modes

independently of any completion model. However, since the global planner operates on the occupancy grid cell-by-cell, parameter adjustments such as costmap resolution and inflation radius may not fully eliminate the routing problem; gap closing at the map level may also be needed. Additionally, the gap-closing algorithm itself requires further development to handle dynamic SLAM updates reliably before it can be integrated into the live pipeline.

7.6.2 Scenarios

Five navigation scenarios are evaluated:

- **Baseline:** Raw SLAM map, Euclidean frontier costs, no completion or filtering.
- **GWC:** Global wall closing applied to the navigation map.
- **LMC+Filt:** Transient local completion around selected frontiers with invalid frontier filtering.
- **GWC+Filt:** Global wall closing with frontier filtering; GT patches are not used in this case.
- **LMC+GWC+Filt:** Full system combining transient local completion, global wall closing, and filtering.

All completion uses a *transient* strategy: local GT patches are applied only while the associated frontier remains active and discarded when the frontier is deselected. This prevents accumulated GT–SLAM misalignment from degrading navigation over time.

7.6.3 Experimental Setup

Experiments are conducted in a complex HouseExpo environment (Figure 7.12) containing narrow doorways, spurious wall gaps from SLAM noise, and exterior frontiers, making navigation particularly sensitive to partial observability. Four independent SLAM runs (ENV 1–ENV 4) are retained from multiple runs based on best alignment with the offline GT map; results are averaged across runs. The TurtleBot3 platform is used in Gazebo with `gmapping` SLAM (sensor range 15 m; motion noise $srr = 0.1$, $srt = 0.2$, $str = 0.1$, $stt = 0.2$; 20 particles; resolution 0.1 m).

Two navigation-centric metrics are used (lower is better):

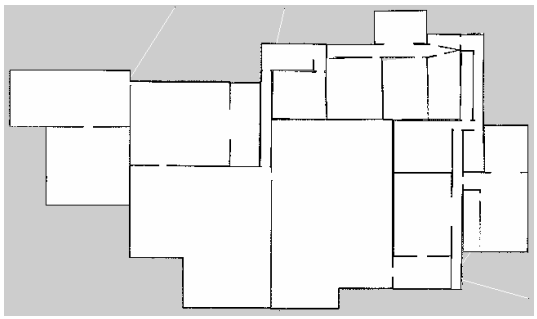


Figure 7.12 The HouseExpo environment used in the GT oracle navigation study. The map contains narrow doorways, parallel corridors with spurious wall gaps, and exterior spurious frontiers from SLAM noise.

- **Navigation duration:** total time (seconds) within a fixed time horizon.
- **makePlan cycles (MP):** number of global planner (A*) invocations inside `move_base`.

A further limitation of this study is that all four runs are conducted within a single HouseExpo environment. While four independent SLAM runs provide some variance across localisation trajectories and the environment presents various navigation patterns (narrow doorways, parallel corridors, and exterior spurious frontiers that together stress different subsystems), the findings cannot be generalised to the full range of indoor layouts evaluated in the main exploration experiments (Section 7.8).

7.6.4 Quantitative Results

Table 7.9 summarises navigation performance across all scenarios and SLAM runs. MP denotes makePlan cycles (lower is better); duration is reported in seconds. Across all four environments, the **LMC+GWC+Filt** configuration consistently achieves the lowest makePlan cycle counts and shortest durations, confirming that the combination of local map completion, gap-closing, and frontier filtering yields the most efficient exploration behaviour. The baseline, by contrast, incurs substantially higher planning overhead in every environment, with ENV 1 and ENV 2 showing the largest absolute differences. Individual scenario results reveal that the benefit of map completion is most pronounced in structurally complex environments, where predictive frontier scoring reduces unnecessary replanning.

Table 7.10 reports the average improvement over the baseline for each scenario. The combined **LMC+GWC+Filt** configuration achieves the largest gain (+48.2% in both MP and duration), while **GWC+Filt** ranks second (+41%). Notably, even

Table 7.9 Navigation metrics comparison across five scenarios and four SLAM runs (best results per run in bold)

Run	Metric	Baseline	GWC	LMC+Filt	GWC+Filt	LMC+GWC+Filt
ENV 1	MP	948	564	509	443	**440**
	Duration	3164	1878	1701	1481	**1471**
ENV 2	MP	962	553	564	482	**427**
	Duration	3210	1847	1884	1611	**1429**
ENV 3	MP	602	490	464	560	**431**
	Duration	2010	1634	1551	1871	**1441**
ENV 4	MP	818	543	599	471	**426**
	Duration	2730	1811	2001	1574	**1421**
Average	MP	832.5	537.5	534.0	489.0	**431.0**
	Duration	2779	1792	1784	1634	**1440**

the simplest augmentation—**GWC** alone—reduces both MP cycles and duration by over 35%, suggesting that gap closing provides a strong baseline benefit regardless of filtering. The incremental gains from adding frontier filtering (**GWC+Filt** vs. **GWC**) and local map completion (**LMC+GWC+Filt** vs. **GWC+Filt**) confirm that each component contributes independently to the overall improvement.

Table 7.10 Average performance improvement over baseline for each scenario

Scenario	MP Improvement	Duration Improvement
GWC	$+35.4\%$	$+35.5\%$
LMC+Filt	$+35.9\%$	$+35.8\%$
GWC+Filt	$+41.3\%$	$+41.2\%$
LMC+GWC+Filt	$\mathbf{+48.2\%}$	$\mathbf{+48.2\%}$

7.6.5 Qualitative Results

Figures 7.13 and 7.14 illustrate the two primary failure modes addressed by the oracle system.

7.6.6 Why This Approach Was Not Adopted for the Exploration Pipeline

Two risk factors led to the decision not to inject completed patches into the navigation costmap in the main exploration system. First, if a frontier is erroneously predicted as lying on a wall and subsequently removed, the robot may permanently miss an entire unexplored sub-region with no recovery mechanism—a risk that persists regardless of model accuracy. Second, a wall prediction in

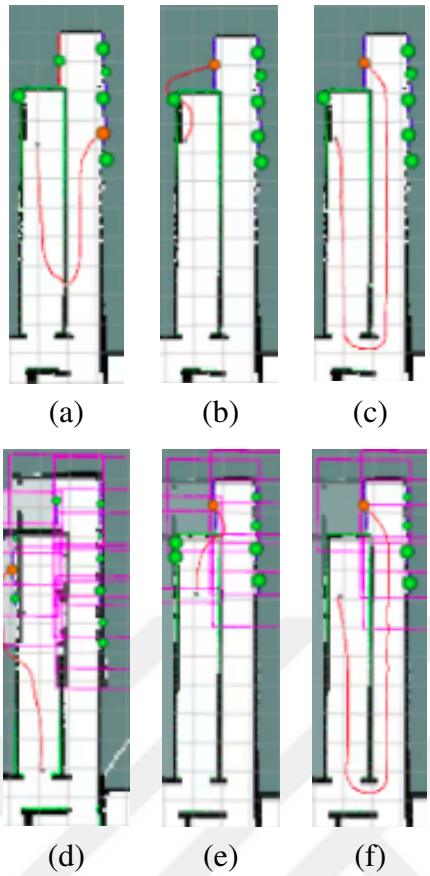


Figure 7.13 Parallel corridors problem. (a–b) Baseline zigzag attempts through wall gaps; (c) eventual resolution. (d) LMC+Filt partial fix; (e) LMC+Filt with alignment shift; (f) Full system (LMC+GWC+Filt) resolved. The pink rectangles show the patches around frontiers. If the completion is performed they are filled with the corresponding patch from GT map. Green spheres are frontiers and the orange sphere is the target frontier.

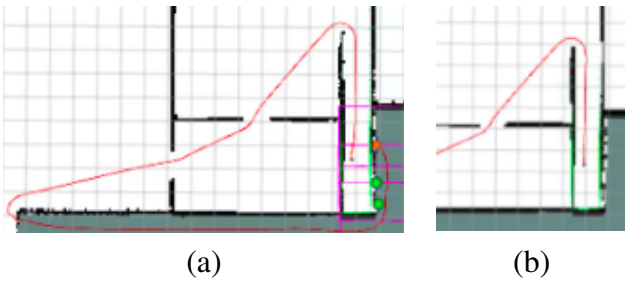


Figure 7.14 Exterior frontier problem. (a) SLAM noise creates spurious target beyond wall which causes the robot to attempt to reach an unreachable region; (b) Filtering eliminates the invalid frontiers.

a genuinely open area could prevent the global planner from finding any path to a frontier, causing navigation to stall. An intermediate approach—using only the closed live map (a gap-closed version of the current SLAM map) for navigation without completed patches—was also considered and implemented, but was abandoned due to synchronisation issues and the need for further algorithm development: the closing operation could complete on a partially updated SLAM map, producing inconsistent costmap representations that caused incorrect path planning; additionally, the gap-closing algorithm itself requires improvement to detect and re-open gaps subsequently filled by the SLAM algorithm. The overhead of maintaining and publishing an additional costmap topic was also a practical concern. These directions remain open for future work (Chapter 9).

7.7 System-Level Exploration Evaluation: Setup

The frontier-centered UNet model selected in Section 7.5 is deployed within the extended `explore_lite` framework described in Chapter 4. This section and Section 7.8 report the full quantitative and representative qualitative evaluation of exploration efficiency across eight HouseExpo simulation environments, comparing two primary configurations: the baseline frontier-based exploration (BASE) and the UNet map-completion system (MC). A ground-truth oracle (GT) is evaluated only for the `4bb74477` environment—the single case where BASE outperforms MC in the majority of milestones—to isolate the effect of prediction window size (Section 7.8.1.2).

7.7.1 Experimental Setup

7.7.1.1 Environments

Eight environments are drawn from the HouseExpo dataset [3]. Each environment is run three times per configuration; all reported values are 3-run averages of elapsed time in seconds to reach each coverage milestone. Coverage is measured as the fraction of ground-truth free-space pixels observed in the SLAM costmap.

The environments are classified into three difficulty tiers based on narrow-corridor ratio, wall density, openness ratio, and free area. The narrow-corridor ratio measures the fraction of free space disappearing after a 5-pixel morphological erosion (structuring element radius = 5 px = 0.5 m); the openness ratio is its complement. Table 7.11 summarises the key structural metrics; Figure 7.15 shows the complete map of each environment.

Table 7.11 Structural properties of the eight evaluation environments; difficulty tiers assigned by a four-rule heuristic

ID	Tier	Free (m^2)	Narrow (%)	Wall (%)	Open (%)
`10161ca6`	Easy	1781	14.7	3.6	85.3
`46155736`	Easy	406	28.8	7.1	71.2
`3a4b51bf`	Medium	1438	20.6	5.4	79.4
`4bb74477`	Medium	1352	25.0	5.8	75.0
`1260131f`	Medium	1077	20.7	5.0	79.3
`958a8b57`	Hard	1152	38.5	7.7	61.5
`d0733a59`	Hard	675	42.5	10.0	57.5
`dd3b9ef6`	Hard	1819	28.8	6.3	71.2

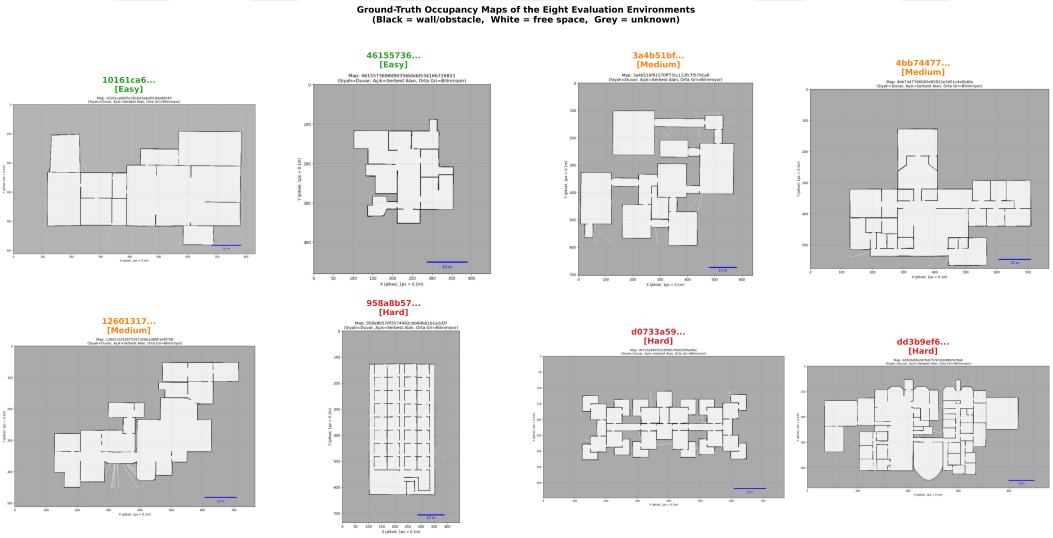


Figure 7.15 Ground-truth occupancy maps of the eight evaluation environments. Black = wall; white = free space; grey = unknown. Environments ordered by difficulty tier (Easy, Medium, Hard).

7.7.1.2 Runtime Environment and Execution Procedure

All system-level exploration experiments were executed within a Docker-based ROS environment. Containerisation was adopted due to compatibility issues between the host operating system and the required ROS/Gazebo package dependencies. This approach ensured a stable and reproducible runtime environment across all experiments, eliminating dependency conflicts and simplifying system deployment.

The experiments were conducted using a TurtleBot3 simulation in Gazebo, with SLAM, navigation, and exploration modules running concurrently inside a single container. GPU support was enabled for the map completion module, while the workspace directory was mounted between the host and the container to allow persistent access to logs and outputs. All ROS nodes were executed within the same container using multiple terminal sessions to enable parallel operation.

The exploration pipeline follows a staged ROS launch sequence. The simulation, SLAM, navigation, and exploration modules are initialised for all configurations (BASE and MC), while the learned map completion module is activated only in the MC configuration.

The execution sequence is defined as follows:

1. `roslaunch turtlebot3_gazebo houseexpo.launch`
2. `roslaunch turtlebot3_slam turtlebot3_slam.launch`
3. `roslaunch turtlebot3_navigation move_base.launch`
4. `roslaunch explore_lite explore.launch`

For the MC configuration, the following additional node is launched:

- `roslaunch map_completion batch_map_completion_unet.launch`

In the BASE configuration, no learned completion model is used. The BASE configuration operates solely on the raw SLAM map.

All nodes remain active concurrently during each run, forming a real-time exploration loop in which frontier detection, navigation, and (optional) map completion operate continuously throughout the experiment.

7.7.1.3 Configurations and Evaluation Protocol

Three exploration configurations are evaluated:

- **BASE** (`use_completed_map=0`): baseline frontier-based exploration; cost follows the BASE case of Equation 4.5.
- **MC** (`use_completed_map=1`, `with_map_completion=1`): cost follows the MC cases of Equation 4.5, incorporating the predicted free-space gain and wall penalty.
- **GT** (`use_completed_map=1`, `with_map_completion=0`): ground-truth oracle; patches cropped directly from the known floor plan; cost computation otherwise follows the same MC formula. Evaluated only for `4bb74477`.

For the `4bb74477` environment, an additional GT variant using 64×64 patches (GT64) is evaluated alongside GT32 (32×32) to isolate the effect of prediction window size (Section 7.8.1.2).

Frontier clustering (`use_clustering=1`) is enabled for all configurations to ensure a fair comparison: it reduces the number of patches queried from the completion service in the MC condition and is matched in BASE to eliminate it as a confounding variable.

The primary evaluation metric is elapsed time (seconds) to reach each coverage milestone: 70%, 75%, 80%, 85%, 90%, 95%, and maximum achieved coverage (Max). A total of 56 milestone comparisons are available (8 environments $\times$ 7 milestones), each derived from 3-run averages. Among these, the **95% milestone is designated as the primary evaluation boundary**: it represents the practical completion point of autonomous frontier-based exploration, as the remaining frontier cells beyond this threshold expectedly correspond to geometrically constrained corner pockets that the system deliberately defers and that may be more suitably addressed by offline or human-assisted methods. Results across all milestones are reported for completeness, but system-level conclusions are drawn primarily from the 95% milestone.

7.7.1.4 Manual Stall Intervention

During exploration runs, occasional stall events were observed in which the robot became stuck despite the `explore_lite` blacklist mechanism—the blacklist

sometimes failed to recover from persistent navigation failures within the allotted timeout. To prevent such incidental navigation failures from confounding the comparison of exploration strategies, stall events were manually detected and resolved by advancing the robot to the nearest free cell and resuming exploration. The time elapsed during each manual intervention is included in the reported elapsed times; stall periods themselves are excluded from the count, and only the active intervention duration is added. This approach isolates the evaluation of frontier selection quality from the stochastic behaviour of the global planner.

7.7.2 Effect of Clustering on Exploration Efficiency

Frontier clustering groups spatially adjacent frontier cells into representative targets before the cost function is evaluated. The clustering algorithm uses a fixed radius parameter of $r_{\text{cluster}} = 0.5\,\text{m}$; frontiers within this radius of a cluster centroid are merged into a single representative point. This reduces the number of UNet inference calls per exploration cycle in the MC configuration and limits the combinatorial growth of the frontier set in dense environments.

Before comparing BASE and MC, the isolated effect of frontier clustering is characterised in the BASE configuration across representative environments from each difficulty tier (Easy, Medium, and Hard).

In the two easy environments the effect is mixed. In `10161ca6`, clustering wins 14 of 20 milestone comparisons against no-clustering BASE, with the largest advantage in the 80–90% band ($\Delta \approx -15$ to $-30\,\text{s}$). In `46155736` the relationship is reversed: BASE without clustering outperforms BASE with clustering at every milestone from 55% onwards (10.5 s penalty at 98%, 12.1% slower). This environment has the highest openness ratio (71.2%) and largest initial unknown fraction (79.6%), suggesting that in small environments with many diffuse frontiers clustering prematurely discards useful targets.

In medium and hard environments the clustering effect is neutral to mildly positive, with differences typically below 5% at any individual milestone. Table 7.12 summarizes the milestone win counts for BASE with and without clustering across the three representative environments, where win count is the number of milestones where clustering is faster (out of 20 per environment) and Δ_{avg} is the mean time difference (negative = clustering faster).

For the MC configuration, the benefit of clustering is more pronounced because each cluster representative requires one UNet inference call. Reducing frontier count via clustering directly lowers per-cycle computation. Clustering is therefore retained as

Table 7.12 Effect of frontier clustering on BASE exploration efficiency

Environment	Tier	Wins (C vs. no-C)	Δ_{avg}
`10161ca6`	Easy	14/20	-18 s
`46155736`	Easy	0/20	$+8$ s
`4bb74477`	Medium	10/20	≈ 0 s
`dd3b9ef6`	Hard	11/20	-5 s

a fixed system component for all MC experiments.

7.8 System-Level Exploration Evaluation: Results

7.8.1 Quantitative Results

7.8.1.1 Aggregate Performance Across Environments

Table 7.13 summarises the 8-environment average elapsed time for each milestone. Figure 7.16 presents the same data with statistical significance markers derived from the Wilcoxon signed-rank test [56], a non-parametric paired test that assesses whether the distribution of differences between MC and BASE elapsed times is centred around zero. It requires no assumption of normality, making it appropriate for the small paired samples here ($n = 24$ paired observations: 8 environments $\times$ 3 runs). The reported p-value is the probability of observing a test statistic at least as extreme as the one obtained under the null hypothesis that the two configurations are equivalent; values below a chosen threshold (here $\alpha = 0.05$) are interpreted as statistically significant evidence that MC and BASE differ at that milestone [57].

Table 7.13 Cross-environment average elapsed time (seconds) to reach each coverage milestone; Δ = MC$-$BASE (negative = MC faster)

Milestone	MC (s)	BASE (s)	Δ (s)	Adv. (%)	Sig.
70%	779.1	856.0	-76.9	$+9.0$	$p < 0.01$
75%	873.8	969.0	-95.2	$+9.8$	$p < 0.01$
80%	989.8	1103.1	-113.3	$+10.3$	$p < 0.01$
85%	1129.0	1184.6	-55.6	$+4.7$	$p < 0.05$
90%	1214.8	1249.6	-34.8	$+2.8$	n.s.
95%	1286.9	1436.4	-149.6	$+10.4$	$p < 0.001$
Max	1570.4	1565.0	$+5.4$	-0.3	n.s.

Across 56 milestone comparisons, MC achieves a lower elapsed time in **45 out of 56 cases (80.4%)**, with an average speedup of **7.2%** (median 5.9%) across all milestones. The 95% milestone is the most consistent and practically relevant boundary: MC wins in **all eight environments** ($p < 0.001$, average advantage $+10.4\%$). This threshold marks the point at which remaining frontier cells

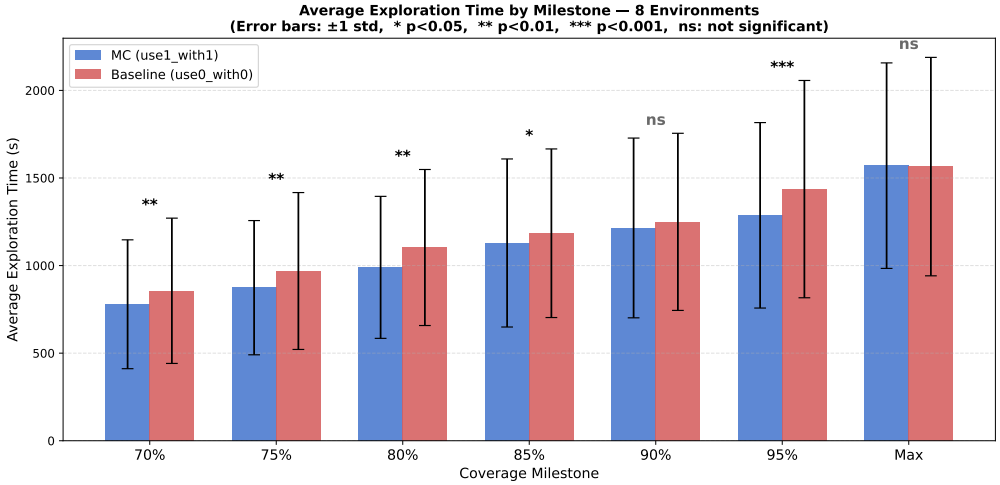


Figure 7.16 Average elapsed time to reach each coverage milestone across eight environments (error bars: ± 1 SD; Wilcoxon signed-rank significance markers, $n = 24$). MC (blue) is consistently faster than BASE (red) at all milestones except Max.

expectedly correspond to geometrically constrained corner pockets that the system deliberately defers—due to the high wall-prediction penalty near obstacles—and that may be more suitably addressed by offline global completion or human-assisted verification. At Max coverage the two methods converge ($p \geq 0.05$), confirming that MC improves *exploration efficiency* rather than *final coverage extent*. The full advantage heatmap (Figure 7.17) makes the milestone-by-environment structure visible. The `4bb7` row is the only one predominantly red (BASE faster); the 95% column is uniformly green. Notably, even in the `4bb74477` outlier, MC achieves a marginal advantage at the 95% milestone ($+1.8\%$), indicating that the topological bottleneck primarily affects early-phase exploration rather than late-stage clean-up (Section 7.8.1.2).

Figure 7.18 shows the corresponding statistical significance map; the 95% column is the only one where all eight environments reach $p < 0.001$.

7.8.1.2 Per-Environment Analysis

Figures 7.19 and 7.20 show the milestone-by-milestone time and distance profiles per environment.

All per-environment results reported below use `clustering` for BASE and MC configurations. The isolated effect of clustering is characterised separately in Section 7.7.2.

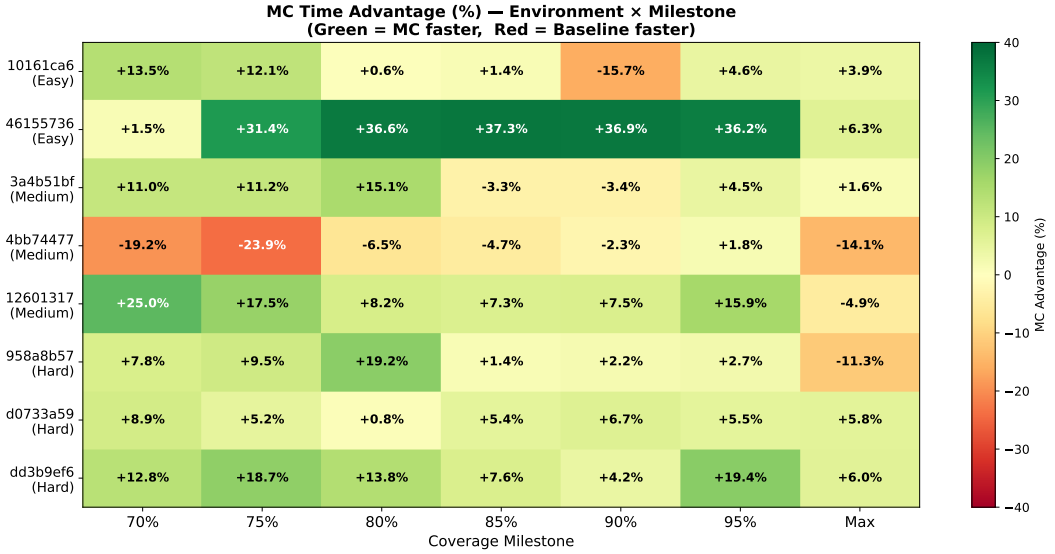


Figure 7.17 MC time advantage (%) relative to BASE for each environment–milestone combination. Green = MC faster; red = BASE faster. Each cell: $(t_{\text{BASE}} - t_{\text{MC}})/t_{\text{BASE}} \times 100$. The 95% column is the only one where all eight environments are green; `4bb74477` is the only predominantly red row.

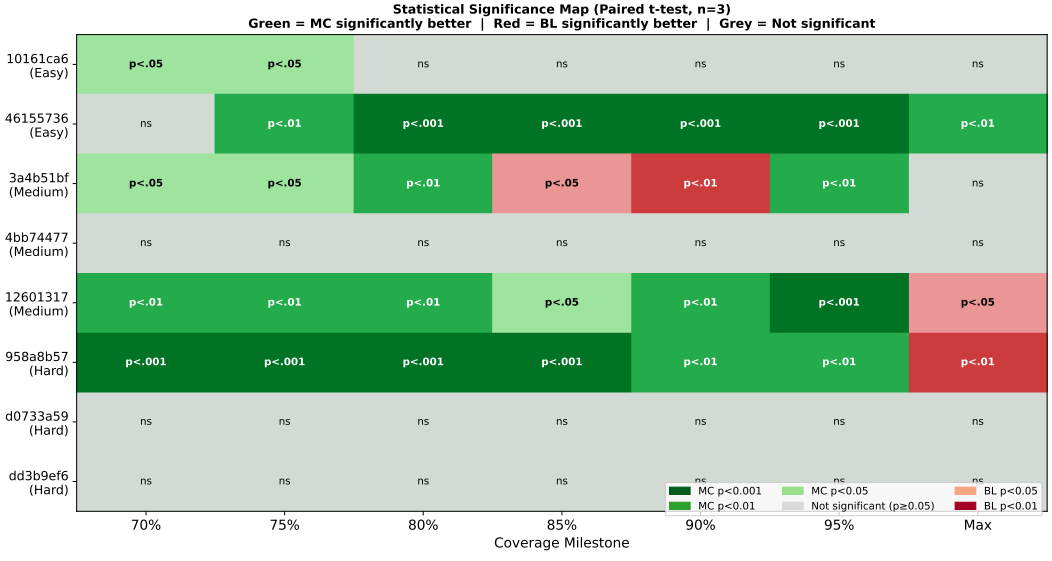


Figure 7.18 Statistical significance map (paired t-test, $n = 3$ per cell). Dark green: $p < 0.001$ (MC); mid green: $p < 0.01$; light green: $p < 0.05$; grey: n.s.; pink/red: BL significant.

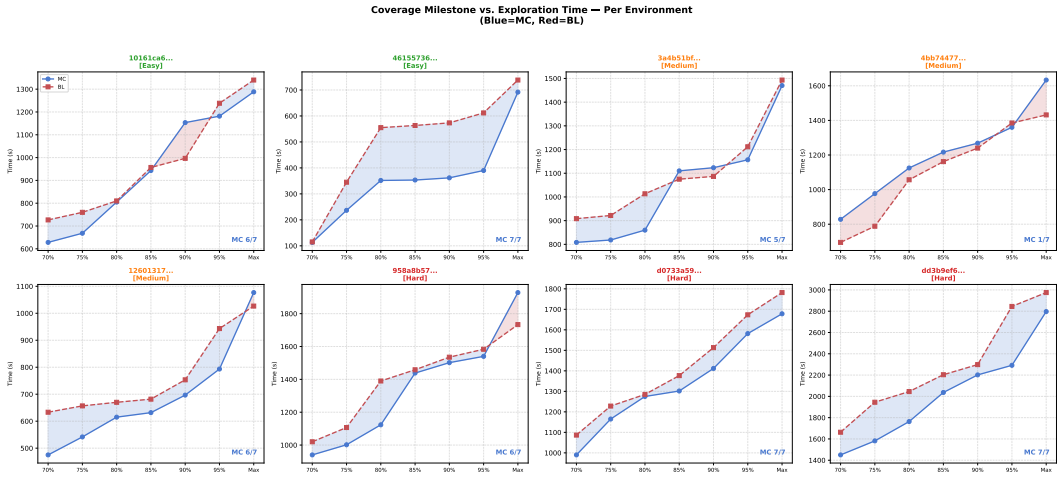


Figure 7.19 Elapsed time to reach each coverage milestone per environment. Blue solid: MC; red dashed: BASE. Shaded area marks where one method leads. Corner label "MC k/7" counts MC milestone wins.

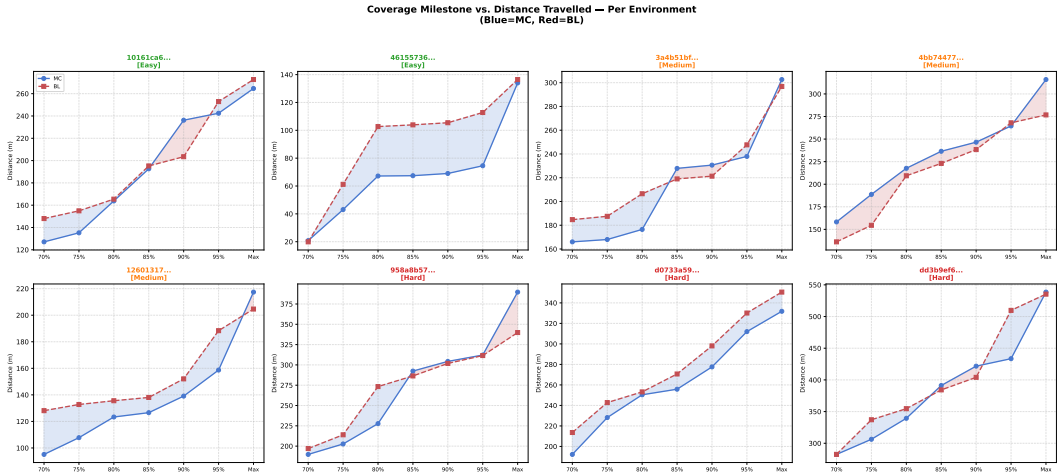


Figure 7.20 Total distance travelled to reach each milestone per environment. Layout identical to Figure 7.19.

`10161ca6` (Easy, MC 6/7).

Large open-plan environment (1,781 m^2, 14.7% narrow corridors). Consistent MC advantage at 70–85% ($\Delta \approx -91$ to -98 s). An anomaly occurs at 90% where MC is 156.7 s slower on average (present in two of three runs), suggesting the 32×32 patch is insufficient to resolve a small number of hard corner frontiers near complete coverage. MC recovers at 95% (-56.7 s).

`46155736` (Easy, MC 7/7).

Despite being the smallest environment (407 m^2), MC achieves its strongest relative advantage: 7/7 milestones, average $+27.1\%$. The high unknown fraction (79.6%) means the completion model is predicting into a large blank canvas whose residential topology aligns well with its spatial priors. The 75–95% band shows the most dramatic gains ($\Delta \approx -108$ to -222 s, roughly 36% speedup).

`3a4b51bf` (Medium, MC 5/7).

Large multi-room environment (1,438 m^2). MC leads at 70–80% (-100 to -153 s) but falls behind BASE at 85–90% ($+35$ to $+37$ s, $p = 0.03$ and $p = 0.008$). At 95% MC recovers (-55 s). The mid-range reversal is consistent across all three runs, pointing to a systematic effect at high but not yet complete coverage.

`4bb74477` (Medium, MC 1/7).

Principal outlier: BASE outperforms MC in 6 of 7 milestones. The largest penalty occurs at 75% ($+188.3$ s avg.; Run 2 reaches $+345$ s). The single milestone where MC wins is 95% (-25 s, not significant). The structural analysis in Section 7.8.1.2 examines why this environment behaves differently from all others.

`1260131f` (Medium, MC 6/7).

Strong MC environment: 3-run average advantage -158.3 s at 70% ($+25\%$) and -150 s at 95% ($+15.9\%$).

`958a8b57` (Hard, MC 6/7).

Most regular grid-like floor plan (38.5% narrow corridors). MC achieves its clearest advantage at 80% (-266.7 s, $+19.2\%$). At 85–95%, MC is faster by time but BASE travels shorter distance—a time–distance decoupling discussed in Section 7.8.1.4.

`d0733a59` (Hard, MC 7/7).

Most extreme structural profile (42.5% narrow corridors, 10.0% wall density). MC wins all 7 milestones, with the strongest advantage at 95% (-91.7 s, $+5.5\%$). In large, complex environments the cost of selecting the wrong distant frontier is high; the completion model's structural prediction steers the robot toward the correct region on the first attempt.

`dd3b9ef6` (Hard, MC 7/7).

Largest environment (1,820 m^2, 96 m span). MC wins all 7 milestones with an exceptional -553.3 s advantage at 95% ($+19.4\%$). Run 3 contributes a -875 s advantage at 95%. High run-to-run variance is expected for large environments but the aggregate is unambiguously MC-positive.

The `4bb74477` Outlier: Analysis and Oracle Study

The `4bb74477` environment is the only case where MC underperforms BASE in the majority of milestones (1/7). However, as discussed in Section 7.8.1.1, the 95% milestone is designated as the primary evaluation boundary, and even in this outlier environment MC achieves a marginal advantage at that threshold ($+1.8\%$). The underperformance is therefore concentrated in the early exploration phase (70–80%), where the topological bottleneck lies beyond the 32×32 prediction window—not in the late-stage clean-up that constitutes the practical completion of autonomous exploration.

Structural context.

The floor plan (Figure 7.15) shows a central north–south corridor with multiple lateral room clusters. The HouseExpo topology for this map is characterised by asymmetric connectivity: the left and right halves are linked by a single narrow passage at the top. A robot starting at the bottom centre must navigate through this topological bottleneck to access the left cluster. The structural metrics (25.0% narrow corridors, 5.8% wall density, 75.0% openness) place it in the middle of the distribution; the bottleneck is a topological rather than a metric difficulty. Other environments with similar aggregate structural metrics (e.g. `3a4b51bf` and `1260131f`) do not exhibit the same underperformance, suggesting that those environments have more distributed connectivity that does not create a single topological bottleneck within the 32×32 prediction window. The aggregate statistics used here do not capture this distinction; identifying which topological

properties predict MC benefit most reliably would require dedicated graph-based environment analysis.

GT oracle comparison.

Table 7.14 reports 3-run average elapsed times for BASE, GT32, MC, and GT64.

Table 7.14 3-run average elapsed time (seconds) for `4bb74477` (best value per milestone in bold)

Milestone	BASE	GT32	MC	GT64
70%	695.0	702.5	828.3	**674.2**
75%	788.8	815.8	978.5	**697.5**
80%	1057.1	1055.6	1126.8	**939.0**
85%	1163.5	1125.6	1218.5	**1006.5**
90%	1241.8	1282.3	1270.2	**1118.2**
95%	1385.0	1319.0	1363.2	**1254.4**
98%	1420.1	1455.6	1396.6	**1322.8**

The GT32 oracle closely tracks BASE: even perfect local completion at patch scale 32 (3.2 m) does not help in this environment, indicating that the useful structural information lies at a scale larger than 3.2 m. The GT64 oracle (64×64, 6.4 m) outperforms all configurations in the 70–95% band by a substantial margin (e.g. -11.6% vs. BASE at 75%, -13.5% at 85%). This demonstrates that the bottleneck topology *is* predictable from the floor plan, but only when the prediction window is large enough to encompass the topological transition.

Interpretation.

The `4bb74477` result is not a failure of the completion concept but of the patch scale matched to environment topology. Increasing the operational patch size from 32 to 64 recovers the advantage, as shown by GT64. This suggests that the overall system performance reported in Section 7.8.1.1 is likely to improve further as the operational patch size grows: environments where topological features approach or exceed the current 32×32 prediction window—currently neutral or slightly negative—would be expected to transition to positive MC advantage, raising the aggregate performance ceiling.

7.8.1.3 Cross-Environment Analysis

Milestone band structure.

MC's advantage is non-uniform across coverage bands. The 95% milestone is the most consistent ($8/8$ environments, $p < 0.001$), reflecting that late-stage

exploration requires predicting where the last unknown pockets are—precisely the task for which patch completion is designed. The 70–80% band is also strong ($7/8$, $p < 0.01$), corresponding to the phase where large connected unknown regions begin to be separated by walls. The 85–90% band is the most contested ($6/8$ and $5/8$ respectively), coinciding with a phase where the remaining unknown area is small enough that baseline distance heuristics can navigate it reasonably well without prediction assistance. This does not contradict the strong 95% result: at 95% the robot must specifically locate the last geometrically isolated pockets—a task for which patch-based spatial prediction provides a more decisive advantage than pure distance scoring. At Max both methods converge.

Difficulty and MC advantage.

Figure 7.21 plots each environment's MC time advantage against its difficulty tier. MC is effective across all three tiers; the benefit depends more on topological structure and unknown fraction than on the difficulty tier alone. The `46155736` environment achieves the highest MC advantage ($+27.1\%$) despite its Easy classification, owing to its large unknown fraction (79.6%) which provides an abundant prediction canvas for the completion model. The `4bb74477` outlier is the only point below the zero line, consistent with the topological mismatch explanation discussed in Section 7.8.1.2.

The slight performance dip in the noted exception is attributed to environments with high structural symmetry but extremely narrow passages. In such cases, a false-positive wall prediction in a narrow choke-point can disproportionately increase the path length by forcing the robot to take a significantly longer alternate route to a frontier that was actually accessible.

Structural profiles.

Radar charts (also known as spider charts or Kiviat diagrams) are a multivariate visualisation technique in which each axis represents one variable, all axes share a common origin, and values are connected to form a polygon [58]. The area and shape of the polygon reflect the overall profile of the entity being compared. The radar profiles in Figure 7.22 display the four structural metrics used in the difficulty tier heuristic for each environment on a normalised scale, enabling a quick visual comparison of how environments differ across all dimensions simultaneously. The `4bb74477` panel shows a polygon comparable to other Medium-tier environments, confirming that the outlier behaviour is not captured by aggregate structural metrics. The underperformance of MC in this environment is attributed to topological

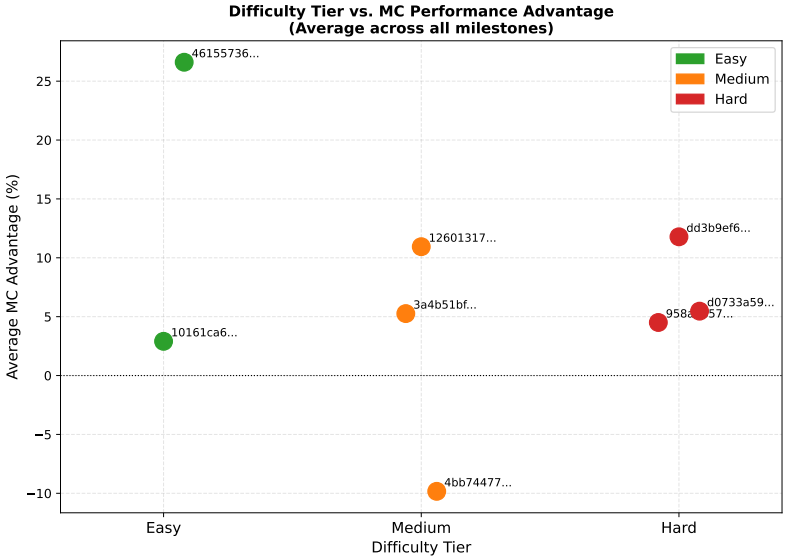


Figure 7.21 Scatter plot: average MC time advantage over BASE (y-axis) per environment, grouped by difficulty tier (x-axis). Colours: green = Easy, orange = Medium, red = Hard. Points within the same tier are separated by a small horizontal jitter for readability. `4bb74477` is the only point below the zero line.

properties of the floor plan that are not captured by aggregate metrics such as narrow-corridor ratio and openness, and whose spatial scale appears to exceed the 32×32 prediction window. This hypothesis is supported by the GT oracle comparison: a 32×32 oracle provides no advantage over BASE, while a 64×64 oracle recovers a substantial gain. Identifying such properties would require graph-based topological analysis and is identified as a direction for future work (Chapter 9).

7.8.1.4 Time–Distance Decoupling

In several environments and milestone bands, MC achieves shorter elapsed time but longer distance than BASE (notably `958a8b57` at 85–95% and `dd3b9ef6` at 85–90%). This decoupling is physically consistent: MC moves more continuously because the completion-based cost function steers the robot toward high-gain frontiers even at moderate distance, reducing time spent stationary or replanning. BASE often selects the nearest frontier regardless of predicted gain, resulting in many short-distance incremental moves. Elapsed time is the primary indicator of exploration efficiency; path length is reported for completeness but should not be used as the sole efficiency criterion when the two metrics diverge.

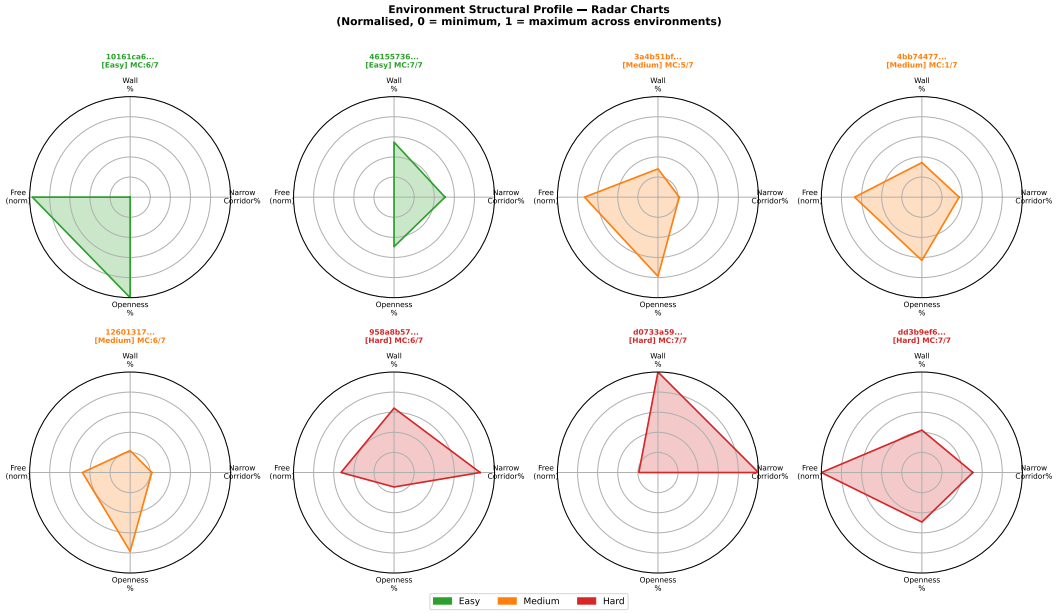


Figure 7.22 Structural profile radar charts for all eight environments (normalised per axis). Each panel reports the MC milestone win count. The `4bb74477` panel (MC 1/7) shows an unremarkable polygon, confirming that the outlier behaviour is not captured by aggregate structural metrics.

7.8.2 Qualitative Results

Figure 7.23 illustrates representative examples of frontier-centered patch extraction and the corresponding online completion predictions. For each frontier, a local region is extracted around its centroid and passed to the completion model. The predicted structure within this region is then used to adjust the frontier cost, without directly modifying the underlying SLAM map.

These visualisations provide intuition for the observed performance differences reported in the preceding sections.

7.9 Chapter Summary and Key Findings

7.9.1 Model Selection Hierarchy

1. **Rule-based baseline:** Geometric completion achieves 0.879 IoU on 7,302 HouseExpo maps without training, establishing a strong structural-cue reference. Note that this metric is computed over the full region of interest rather than the masked unknown region, and the method is highly sensitive to predefined thresholds for door detection and guide-line extraction. It is limited to axis-aligned environments and does not produce the online patch format required by the exploration pipeline.

2. **Upper-to-lower completion:** CVAE (avg. IoU 0.933, FID 28.9) outperforms

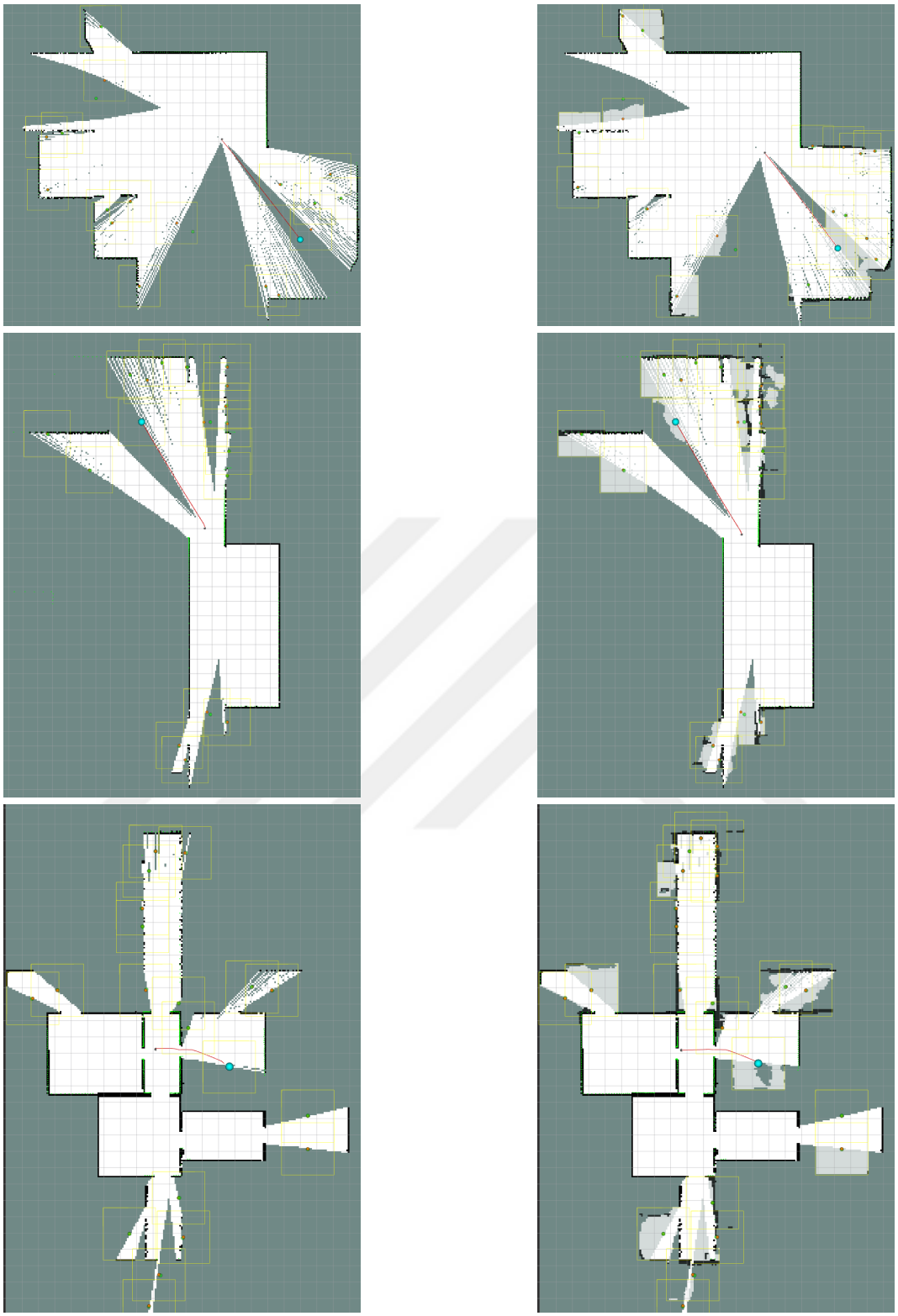


Figure 7.23 Completion-guided frontier evaluation examples. Left column: patch selection; right column: prediction. Green: selected targets; orange: centroids; cyan: current goal.

ImageGPT (avg. IoU 0.879, FID 96.7) on every metric with a $> 3000\times$ inference advantage. Selected for directional completion; not used in the online exploration pipeline.

3. **Global completion:** UNet achieves the best aggregate accuracy under FULL evaluation (IoU 0.796, F1 0.884, Acc 0.916); GEOM attains the highest SSIM (0.848). These results are reported over the entire 128×128 map rather than masked unknown cells, as global completion is used for offline model family comparison rather than direct deployment. Both findings inform the frontier-centered model choice in the subsequent setting.

4. **Frontier-centered local completion:** Among deployment-eligible models ($\geq$ 20 img/s), UNet dominates consistently across both dataset sizes (SDS MASKED avg(IoU, F1) 0.364 vs. next-best Pix2Pix-hinge 0.363). LaMa (12.67 img/s) and $\text{ImageGPT}_{\text{local}}$ ($\approx$0.05 img/s) are excluded from deployment consideration due to insufficient throughput; RePaint, despite passing the throughput threshold, is unsuitable due to its iterative denoising cost and high variance under limited training data. Pix2Pix-hinge and Seg-GAN show competitive results but exhibit training instability and occasional hallucination at this scale. ViT requires substantially more data to learn spatial priors for thin-wall localisation and produces discontinuous wall predictions due to its coarse patch tokenisation. CVAE fails under irregular-mask conditions (SDS MASKED IoU 0.201). Among UNet variants, $\texttt{UNet}_{\texttt{base}}$ achieves the highest five-metric composite score (0.627) and is selected for the exploration pipeline.

7.9.2 System-Level Performance Summary

The integrated exploration system across eight HouseExpo environments:

- MC outperforms BASE in **45 of 56 milestone comparisons (80.4%)**, with an average speedup of **7.2%** (median 5.9%) across all milestones.
- The **95% coverage milestone** is the most consistent and practically relevant evaluation boundary: MC wins in all eight environments ($+10.4\%$, $p < 0.001$, $n = 24$). Beyond this point, remaining frontier cells are geometrically isolated corner pockets deliberately deferred by the system's cost function, and may be more suitably addressed by offline or human-assisted methods.
- At **Max coverage** the two methods converge (avg. difference $< 1\%$, not significant): MC improves *exploration efficiency*, not final coverage extent.

- The strongest MC environment is `46155736` (7/7, avg. $+27.1\%$); the only BL-favourable environment is `4bb74477` (1/7, avg. -10.5%), explained by a topological bottleneck larger than the 32×32 prediction window. Importantly, even in this outlier environment MC achieves an advantage at the primary 95% milestone ($+1.8\%$, Section 7.8.1.2), confirming that the underperformance is concentrated in the early exploration phase rather than at the practically relevant coverage threshold.

- GT64 on `4bb74477` outperforms both BASE and GT32 in the 70–95% band (up to -28.7% vs. MC at 75%), demonstrating that the bottleneck is predictable with a wider spatial context.

- The supplementary single-environment GT oracle study recorded a **+48%** navigation improvement under ideal local completion; this result is influenced partly by wall-gap path planning artifacts and is reported as a contextual by-product observation rather than a formal benchmark target.

7.9.3 Structural and Methodological Insights

1. **Patch scale and topology must be matched.** The `4bb74477` outlier shows that a prediction window smaller than the relevant structural feature provides no benefit and can actively mislead frontier selection. GT64 recovers the advantage.

2. **Clustering is environment-dependent but operationally necessary.** In small environments clustering can reduce performance by prematurely merging frontier information. In medium and hard environments it is neutral to mildly beneficial and is essential for keeping MC's per-cycle inference cost bounded.

3. **The 85–90% coverage band is the most variable.** This band corresponds to the transition between the main exploration phase and the final clean-up phase. Both `10161ca6` and `3a4b51bf` show statistically significant reversals at this band. Since the total area of the environment is not known in advance during online exploration, it is not possible to detect this transition point reliably at runtime. However, if the approximate floor area were provided as a prior—for example, from building plans or prior surveys—a coverage-adaptive scoring strategy could be activated when estimated progress exceeds a threshold, potentially recovering the MC advantage in this band. This is identified as a direction for future work (Chapter 9).

4. **MC is effective across both easy and hard environments.** The positive difficulty–advantage correlation in Figure 7.21 indicates that structural complexity does not prevent MC from working effectively; the benefit depends more on topological structure and unknown fraction than on the difficulty tier alone.

Reproducibility

All code and models associated with this chapter are publicly available at `https://github.com/f0120050/local_map_completion_and_integration_to_exploration`. Specifically: model training and evaluation pipelines for all model families are provided in `local_map_completion.zip`; the `MapCompletionBatch` ROS service node is provided in `integrated_code_for_map_completion.zip`; the completion-aware `explore_lite` package is provided in `m-explore_20260312.zip`; exploration run logs for all eight environments (BASE and MC, three runs each) are provided in `exploration_logs_20260315.zip`; and the structural metrics in Table 7.11 together with all system-level figures presented in this chapter are fully reproducible using `compute_env_metrics.py` and `generate_thesis_figures_final.py`, both available in `env_classifier_and_figure_generator.zip`.

8
DISCUSSION

This chapter interprets the experimental findings reported in Chapter 7 and situates them within the broader research landscape. The discussion is organised around six themes: (i) the relationship between completion quality and exploration efficiency, (ii) convolutional priors and the geometry-dominant nature of local completion, (iii) the role of spatial scale in patch-based prediction, (iv) the gap between isolated model metrics and system-level benefit, (v) limitations of the current system, and (vi) relation to prior work.

Throughout this chapter, p-values are reported as indicators of statistical significance. A result is considered statistically significant when $p < 0.05$, meaning the observed difference would arise by chance with probability less than 5% under the null hypothesis of no difference. The p-values reported here are computed from paired comparisons across repeated runs using a Wilcoxon signed-rank test.

8.1 Completion Quality and Exploration Efficiency

A central motivation for this thesis is the hypothesis that better map completion translates into more efficient autonomous exploration. The experimental results provide strong but nuanced support for this hypothesis.

At the system level, the UNet-based MC system outperforms the classical frontier baseline in 80.4% of milestone comparisons across eight environments, with an average speedup of 7.2% and a consistent $+10.4\%$ advantage at the 95% coverage milestone across all eight environments. The benefit is most pronounced at the 95% milestone—all eight environments, $+10.4\%$, $p < 0.001$—and at the 70–80% band ($+9$–10%, $p < 0.01$). These are precisely the phases where selecting the correct frontier matters most: at 70–80% the robot must choose which unexplored room cluster to enter first; at 95% it must locate the last remaining unknown pockets. Both decisions benefit directly from a model that can hypothesise structure beyond the current frontier.

8.2 Convolutional Priors and the Geometry-Dominant Nature of Local Completion

A key finding of this research is the consistent superiority of the residual UNet over transformer-based, latent-variable, and sequence-based alternatives in the frontier-centered local completion setting. This result is stable across both dataset sizes (SDS and BDS), confirming that the advantage is not a matter of data scale but of inductive bias.

Occupancy grids possess a structural regularity—straight walls, orthogonal corners, consistent corridor widths—that is inherently suited to the local receptive fields of convolutional layers. UNet's skip connections allow predictions to be anchored to known SLAM fragments, ensuring that hallucinated regions are topologically consistent with the observed environment.

The failure of CVAE in frontier-centered tasks, despite its success in directional (upper-to-lower) completion, is particularly instructive. In the upper-to-lower setting, the conditioning structure is explicit and fixed: the upper half is always known, the lower half always unknown. This regularity allows the CVAE decoder to learn a reliable mapping from condition to target. In frontier-centered patches, not only does the unknown region occupy any irregular sub-region of the patch with a shape that changes from frontier to frontier, but the training data itself is derived from real SLAM exploration rather than synthetic masking—introducing additional distributional differences. The CVAE's latent space, shaped during training by full-patch encoder posteriors, does not generalise to this dynamic masking regime—a direct empirical demonstration of the task-formulation dependency identified in Chapter 3.

The GEOM baseline offers a complementary insight. In global 128×128 completion, GEOM achieves the highest SSIM (0.848) among all models, outperforming learned alternatives on this metric. This is attributable to SSIM's sensitivity to structural preservation—straight lines, contrast at boundaries—which rectilinear floor plans provide in abundance. Importantly, this SSIM result holds only for FULL evaluation; under the MASKED protocol, GEOM's advantage disappears entirely.

In the frontier-centered setting, GEOM scores approximately zero on the masked region. The fundamental reason is that 32×32 patches are too small to contain the geometric cues—door-frame pixel triplets, extended wall segments—on which the GEOM algorithm relies. When a patch is centred on a frontier, the adjacent wall is typically only partially visible, and any door structure at the

boundary of the known region is likely cropped out of the patch entirely. As a consequence, the door-detection and wall-extension routines of the GEOM algorithm find nothing to act on and produce no prediction. If patch sizes were increased substantially, allowing more structural context to be included, GEOM performance in the frontier-centered setting would be expected to improve. The divergence between GEOM's FULL and MASKED performance illustrates why task-appropriate evaluation matters: FULL metrics capture overall structural similarity, while MASKED evaluation directly measures prediction quality over the regions the model must actually infer.

8.3 The Role of Spatial Scale

The `4bb74477` environment provides a clear demonstration that the spatial scale of the prediction window must be matched to the scale of the relevant structural features. The floor plan contains a a suspected topological feature whose spatial scale appears to exceed the 32×32 (3.2 m at 0.1 m/pixel) prediction window. The GT oracle operating at 32×32 closely tracks baseline performance, confirming that even perfect completion at this scale provides no useful information about the bottleneck. Increasing the prediction window to 64×64 (6.4 m) in the oracle setting recovers a substantial advantage (-11.6% to -13.5% improvement across the 70–95% band).

This result generalises a known principle from the receptive field literature in computer vision: a model's ability to predict a structural feature is bounded by whether that feature fits within its effective context window. In robotic exploration, this translates to a requirement that the prediction patch size be commensurate with the topological granularity of the environment. Environments with fine-grained topology are well served by 32×32 patches; environments with coarse topology require larger patches.

A larger patch size will not, however, automatically improve results across all environments. In environments with complex cost landscapes, a larger predicted free-space region may amplify the r_f reward disproportionately, causing the robot to over-commit to distant high-gain frontiers. Determining which patch size is appropriate for a given environment therefore requires understanding that environment's topological granularity. A more principled approach would involve examining a broader set of environments and identifying which topological properties predict MC benefit most reliably, then using that knowledge to drive an adaptive patch sizing strategy.

The current system uses a fixed 32×32 patch size for all environments. An adaptive patch sizing strategy or a multi-scale completion architecture that processes patches at multiple resolutions simultaneously would allow more robust generalisation across diverse environments. Both directions are prioritised in Chapter 9.

The offline comparison of 32×32 and 64×64 trained models is subject to an additional evaluation constraint: the 64×64 test set spans only 4 environments (compared to 12 for 32×32), because patches are extracted from the same source pool and larger patches reduce the available count. The observed performance differences between resolutions therefore reflect the combined effect of patch size and reduced training and test diversity, rather than patch size alone. This limits the strength of conclusions that can be drawn from the resolution comparison; a controlled study with independently sampled, resolution-matched test sets remains as future work.

8.4 Isolated Metrics vs. System-Level Benefit

8.4.1 FULL vs. MASKED Evaluation

A recurring theme throughout the experimental evaluation is the gap between isolated model quality metrics and system-level exploration benefit. Several models—most notably GEOM and CVAE—score competitively under FULL evaluation but show substantially lower performance under MASKED evaluation. FULL metrics are reported to enable comparison with prior work that reports full-patch metrics (see Table 2.6 in Chapter 2); the majority of reviewed studies use this convention and capture overall structural similarity(see Chapter 2); MASKED evaluation is the primary criterion as it directly measures prediction quality over the regions the model must actually infer. The UNet achieves the highest masked IoU and also the best exploration performance, supporting the general hypothesis that better completion translates to better exploration.

8.4.2 Oracle Observation and Achieved Gain

The supplementary GT oracle navigation study [8] recorded a $+48\%$ reduction in *navigation duration* on a single environment under ideal local completion combined with wall closing and frontier filtering. As discussed in Chapter 5, this result should not be compared directly with the UNet exploration results: the two studies measure fundamentally different aspects of the pipeline. The oracle serves as a preliminary investigation confirming that the integration architecture can produce substantial navigation benefit under ideal conditions, motivating tighter pipeline

coupling (Chapter 9).

8.5 Limitations

Simulation-only evaluation.

All experiments are conducted in Gazebo using HouseExpo-derived environments. Real-world SLAM maps contain additional noise and drift not fully captured by the simulation parameters used here. Physical robot validation remains as future work.

Static environments.

The system assumes a static environment throughout exploration. Dynamic obstacles and non-stationary layouts are not modelled. Map completion predictions trained on static floor plans would be systematically wrong in dynamic environments.

2D-only representation.

The occupancy grid representation is limited to 2D. Multi-floor buildings, ramps, and 3D obstacle arrangements are not captured. The HouseExpo environments are derived from the SUNCG 3D dataset, so the ground-truth floor plans reflect realistic 3D structures projected to 2D; however, any vertical variation in those structures is lost in the occupancy representation.

Fixed patch size.

The deployed UNet uses a fixed 32×32 prediction window, which is insufficient for environments whose relevant structural features exceed the patch scale. The `4bb74477` oracle analysis demonstrates this limitation directly.

Prediction window and planning decoupling.

Completion predictions are used only at the frontier selection stage; the navigation stack plans on the unmodified SLAM map. The supplementary oracle navigation study [8] observed that combining navigation-map injection with wall closing and frontier filtering produced a $+48\%$ improvement in navigation duration on a single environment, though this result is partly influenced by SLAM–path-planner artifacts arising from narrow wall gaps in the SLAM map, and synchronisation risks remain with the gap-closing operation. The current integration deliberately limits completion to the frontier-scoring stage for robustness reasons. Future work could

investigate global planner parameter tuning and synchronisation mechanisms that would allow tighter coupling without introducing the failure modes that motivated the current conservative design (Chapter 9).

Latency-constrained scoring and planning.

Several completion-aware enhancements were implemented but disabled for the reported experiments due to per-cycle latency constraints on the evaluation hardware. These include a completion-aware A* heuristic that routes the planner through predicted-free corridors rather than treating all unknown cells uniformly, and richer frontier scoring terms: *known-side free pixel counting* (counting only the free pixels on the currently observed side of the frontier boundary, giving a more conservative estimate of immediately reachable gain); *door-gap reward and penalty* (amplifying the reward when the completion model predicts a gap adjacent to a wall—indicating a likely doorway—and increasing the penalty when no gap is detected, signalling an unreachable wall); and *direction consistency weighting* (penalising sharp directional reversals relative to the robot's current heading). The current system therefore represents a conservative deployment: the $+10.4\%$ gain at the 95% milestone is achieved using only the simplest scoring signal (r_f), and these richer signals could widen this margin on hardware with greater compute headroom.

Conservative navigation integration by design.

The decision not to inject completed patches into the navigation costmap and not to permanently discard wall-predicted frontiers reflects a deliberate risk management choice rather than a technical limitation. A model operating at moderate masked-region accuracy will occasionally predict a wall where none exists; in a frontier-sparse environment, acting on such a prediction by removing a frontier could prevent exploration of an entire sub-region with no recovery mechanism. The penalty-based approach preserves all frontiers as fallback targets while still communicating the model's structural prediction to the cost function. This conservative integration captures a fraction of the potential benefit suggested by the oracle navigation observation, but does so without introducing failure modes that could make the system worse than the baseline in adversarial prediction cases.

Offline-only completion evaluation.

The frontier-centered completion models are evaluated on offline held-out patches. Patches encountered during live exploration may differ from this distribution due to SLAM noise accumulation and frontier geometry at later exploration stages. The

system-level exploration experiments provide indirect evidence of deployment-time behavior, but a dedicated online evaluation protocol remains as future work. It is worth noting that the offline evaluation distribution is likely to become more representative as the training set grows: models trained on a substantially larger corpus of frontier patches from diverse environments would be expected to better reflect the patch statistics encountered during live runs, partly compensating for distribution shift.

Milestone start sensitivity.

All aggregate statistics are reported from the 70% coverage milestone. An analysis of alternative start thresholds (75%, 80%, 85%, 90%) shows that the overall advantage of MC over BASE is robust to this choice: at each threshold, MC wins a majority of remaining milestone comparisons and the 95% advantage ($+10.4\%$, $p < 0.001$) is unaffected by the choice of lower boundary. The absolute win-count percentage increases slightly when lower early milestones are excluded (e.g. starting from 80% yields MC wins in $\approx$ 83% of comparisons vs. 80.4% from 70%), because the contested 85–90% band contributes a smaller proportion of the total. The reported figure of 80.4% is therefore a conservative estimate of the system's advantage.

Clustering suitability and MC inference cost.

The clustering analysis (Section 7.7.2) reveals that frontier clustering is beneficial for MC—where it reduces per-cycle UNet inference calls—but is neutral or mildly detrimental in small environments with diffuse frontier distributions, such as `46155736`. The MC configuration incurs additional per-cycle latency relative to BASE because each cluster representative requires one UNet inference call; in environments with many small frontiers this cost is amortised by clustering, but in environments where clustering prematurely merges useful targets, both the exploration strategy and the inference cost could be optimised by an adaptive clustering radius. This trade-off is not resolved by the current fixed-radius implementation and is identified as a direction for future work (Chapter 9).

8.6 Relation to Prior Work

The improvement in exploration efficiency is qualitatively consistent with MapEx [12] and SenseExpo [11], both of which demonstrate that prediction-informed frontier scoring reduces exploration time in indoor

environments. MapEx relies on a globally reconstructed occupancy map produced by an ensemble of LaMa networks and is tightly coupled to that prediction engine, making it non-modular with respect to the completion model or the exploration framework. SenseExpo, by contrast, operates on local 256×256 observation patches and is explicitly designed as a modular, plug-and-play ROS node; however, it extracts frontiers directly from the predicted map rather than the observed SLAM map, coupling frontier generation to prediction quality. The present work demonstrates that comparable efficiency gains are achievable with a lightweight local UNet on 32×32 patches within an extended `explore_lite` framework, with frontier detection restricted to observed data and predictions used solely for frontier scoring, without requiring global map reconstruction or retraining the exploration policy end-to-end.

The topological bottleneck analysis in `4bb74477` suggests that prediction-based exploration yields larger gains in environments with structurally ambiguous connectivity—a finding that aligns with the motivation behind uncertainty-driven methods such as that of Spinos et al. [17], even if their evaluation context differs. Their global prediction approach is complementary to the local approach here: local prediction is efficient and modular, requiring no global map reconstruction and remaining compatible with standard ROS navigation stacks; global prediction provides richer spatial context and can reason about topological structure at scales that exceed a fixed local patch size, at higher computational cost. A hybrid approach—local patches for efficient per-frontier scoring, global prediction for long-horizon path planning—is identified as a promising future direction (Chapter 9).

8.7 Summary

This chapter has discussed the key themes emerging from the experimental findings. Completion quality and exploration efficiency are positively correlated in general; however, the relationship is non-monotonic. Excessive prediction sharpness can probably cause over-commitment to distant frontiers, as observed in `1260131f`; this interpretation should be treated as a hypothesis pending dedicated experiments.

The empirical results indicate a clear dominance of convolutional residual architectures, specifically the UNet family, in reconstructing localized frontier patches. This suggests that localized map prediction is primarily a structural-inference problem, where inductive biases toward rectilinear and geometric regularity are more decisive than the raw scale of the training dataset.

The spatial scale of the prediction window must match the topological granularity of the environment. The 32×32 patch is insufficient for coarse-topology bottleneck structures, and increasing patch size must be accompanied by broader environment characterization to avoid introducing new over-commitment artifacts.

System-level evaluation is required to verify whether isolated metric gains translate into deployment benefits.

A supplementary single-environment GT oracle navigation study provides context for what tighter pipeline coupling could achieve under ideal conditions.

The modular extended `explore_lite` architecture provides a testbed for future integration experiments. These findings directly motivate the future directions described in Chapter 9.

9
FUTURE WORK

The experimental findings and limitations identified in Chapter 8 point to several concrete directions for extending the present work. This chapter organises these directions into several groups: improvements to the completion model and its integration, benchmarking against related systems, extensions to the exploration pipeline, broader generalisation and deployment considerations, and longer-term research directions.

9.1 Model and Integration Improvements

9.1.1 Larger and Adaptive Patch Sizes

The `4bb74477` analysis demonstrates that a 32×32 prediction window is insufficient for environments whose relevant structural features exceed 3.2 m. The GT oracle at 64×64 recovers a -11.6% to -13.5% advantage in the critical 70–95% band, confirming that the bottleneck is predictable with a wider context. A natural next step is to train the UNet at 64×64 resolution and evaluate the resulting system across all eight environments. Given that the GT oracle at 32×32 and the UNet at 32×32 perform comparably on average, it is reasonable to expect that the improvement seen in the GT oracle at 64×64 would carry over to a trained model at the same resolution—though this expectation should be validated empirically rather than assumed.

Increasing the patch size alone is not, however, a universal solution. A larger predicted free-space region may amplify the r_f reward disproportionately in open environments, potentially contributing to over-commitment toward distant high-gain frontiers. Determining the appropriate patch size therefore requires understanding the topological granularity of the environment. A principled approach would involve characterizing a broader set of environments to identify which topological properties predict MC benefit most reliably, then using that knowledge to drive an *adaptive patch sizing* strategy that selects the patch

size per frontier based on locally estimated corridor width or map density.Such characterisation would require graph-based topological analysis of the navigability graph—for example, articulation point detection to identify single bottleneck passages, or betweenness centrality to quantify how much traffic is routed through a given corridor. The `4bb74477` outlier, whose underperformance is not reflected in any of the aggregate structural metrics reported in Table 7.11, is a concrete motivating case: its behaviour is consistent with an articulation point that lies just beyond the 32×32 prediction window, a feature invisible to corridor-ratio or openness statistics.

A controlled replication of the 64×64 offline comparison is also warranted: the current evaluation uses a test set of only 4 environments due to the shared source pool, making it difficult to draw strong conclusions about the effect of patch size independent of training and test set size. An independently constructed 64×64 dataset with environment counts matched to the 32×32 baseline would enable a fairer resolution comparison.

9.1.2 Multi-Scale Completion Architectures

Distinct from adaptive sizing—which selects a single patch size per frontier—a multi-scale architecture would process frontier patches at multiple resolutions *simultaneously*. A dual-branch design, for instance, could combine a 32×32 local branch capturing fine wall geometry with a 64×64 or 128×128 context branch capturing coarse room topology, fusing the two representations before predicting the unknown region. This would retain the computational efficiency of local inference while addressing the scale mismatch identified in coarse-topology environments, without requiring a full global map reconstruction. The primary engineering challenge is maintaining the per-cycle latency budget of the exploration loop, which already constrains the complexity of the scoring pipeline.

9.1.3 Tighter Completion–Navigation Coupling

Currently, completion predictions influence only the frontier selection stage; path planning operates on the unmodified SLAM map. The supplementary GT oracle navigation study suggests that wall closing and frontier filtering contribute additional navigation benefit beyond frontier scoring alone [8]. However, as discussed in Chapter 5, the $+48\%$ figure from that single-environment study is partly attributable to SLAM–path-planner artifacts arising from narrow wall gaps, and direct injection of completed patches into the navigation costmap carries synchronisation and reliability risks.

A more tractable near-term direction toward tighter coupling would investigate global planner parameter tuning—specifically, adjusting obstacle inflation radii and costmap resolution to reduce the likelihood of the planner routing through sub-pixel wall gaps in the raw SLAM map, since A* operates cell-by-cell on the occupancy grid and will exploit any gap that falls below the inflation threshold—and synchronization mechanisms that gate the gap-closing operation on explicit SLAM update events rather than a fixed timer. Both changes could be implemented without modifying the completion pipeline and would address the two primary sources of artifact identified in the oracle study.

On the cost function side, a completion-aware A* distance estimator was implemented in the codebase: cells whose neighborhood is predicted to be free receive a lower traversal bias, while predicted-obstacle neighborhoods incur a higher penalty, computed in $O(1)$ per cell using integral images. This variant was disabled in the reported experiments because the additional per-cycle computation exceeded the latency budget on the evaluation hardware. Two directions exist for enabling it in practice: deploying on a platform with greater compute headroom, or investigating a more lightweight implementation—for example, restricting the completion-aware heuristic to a subset of high-scoring frontiers rather than applying it globally. Either approach could meaningfully improve frontier-to-frontier distance estimates by routing the planner's heuristic through corridors the model predicts as open—a particularly useful property in environments with long occluded hallways.

Additionally, exploring alternative global and local planning strategies (e.g., sampling-based planners, learning-based planners, or completion-aware frontier planners) could further improve the integration between prediction and navigation. Such approaches may enable more informed trajectory generation by directly incorporating completion signals into the planning process rather than limiting their use to frontier ranking.

A more immediate step toward tighter coupling is to address the synchronisation issues identified during the GT oracle study: the gap-closing operation should be gated on explicit SLAM update events rather than a fixed timer, and a gap re-opening mechanism should be developed to handle dynamic SLAM updates. Additionally, systematic investigation of global planner parameters—specifically obstacle inflation radii and costmap resolution—could reduce the likelihood of the planner routing through sub-pixel wall gaps in the raw SLAM map without requiring any modification to the completion pipeline. Both directions represent near-term engineering steps prior to full navigation map injection.

9.1.4 Robust Stall and Oscillation Detection

The current progress monitoring mechanism detects stalls by measuring whether the robot-to-target distance decreases within a fixed window. This does not capture forward–backward oscillation, where the robot repeatedly attempts a frontier along alternating paths without net progress. In the reported experiments, such cases were handled via manual intervention applied consistently across all configurations. An automatic oscillation detector—for example, monitoring heading variance or path length relative to displacement over a sliding window—would eliminate the need for manual intervention and enable fully automated multi-run experiments.

9.1.5 Online Model Adaptation

The current UNet is trained offline and remains fixed during deployment. An online fine-tuning mechanism that updates model parameters as the robot observes new regions could improve prediction accuracy in environments that differ from the training distribution. Whether such adaptation is feasible on the embedded hardware of a standard robotic platform remains an open question: the lightweight architecture of the deployed UNet keeps inference fast, but gradient-based updates during active exploration would require careful management of memory and compute budgets. A more tractable variant might involve periodic re-training during exploration pauses, or a small-scale adaptation module that adjusts a learned offset without modifying the full network.

9.1.6 Loss Function Refinement

Alternative loss formulations, including class-weighted cross-entropy and Dice-augmented objectives, represent a direction for further refinement of the U-Net training procedure. Preliminary experiments with these variants showed modest IoU improvements (≤ 0.020) over the unweighted baseline, insufficient to justify rerunning the full exploration pipeline; however, more systematic hyperparameter tuning combined with larger datasets may yield more substantial gains.

9.1.7 Uncertainty-Aware Frontier Scoring

The current cost function uses the scalar free-space gain r_f as the completion signal. A richer signal could incorporate the predicted uncertainty over the completed patch—through an ensemble of UNet models, MC dropout, or the latent variance of a CVAE—to distinguish frontiers whose completion is confident from those that are ambiguous. This would allow the robot to deliberately visit high-uncertainty frontiers when rapid information gain is prioritised, or

to avoid them when navigation plan stability is more important. The primary constraint is computational: the exploration loop already incurs latency from the batch completion service, and any uncertainty estimation mechanism must be lightweight enough to remain within the per-cycle budget. Ensemble or dropout approaches that reuse existing UNet weights without additional training are the most hardware-compatible starting points.

9.1.8 Richer Frontier Scoring Terms

Several additional scoring signals were implemented and tested but ultimately disabled due to latency constraints:

- **Known-side free pixel count:** Rather than counting all free pixels in the completed patch, counting only those on the currently observed side of the frontier boundary provides a more conservative estimate of immediately reachable gain.
- **Door-gap reward and penalty:** If the completion model predicts a gap adjacent to a wall at the frontier, the reward is amplified because the gap suggests a doorway. Conversely, a solid wall prediction increases the penalty.
- **Direction consistency weight:** A score penalising sharp directional reversals relative to the robot's current heading.

These components are fully implemented in the codebase and represent a natural next step both for hardware with sufficient compute headroom and for lightweight variants—such as reduced ensemble sizes or single-pass approximations—that remain feasible on the current platform. A dedicated ablation study evaluating each component independently would also clarify their individual contributions.

9.2 Benchmarking Against Related Systems

9.2.1 Direct Comparison with Information-Gain-Based Frontier Scoring

A direct experimental comparison with the map-completion-augmented exploration framework of [15] remains an open near-term direction. Both approaches use predicted maps to guide frontier selection but differ on several axes: completion task (80×80 center inpainting within a 256×256 context vs. 32×32 frontier-centered outpainting), dataset (KTH floor plans vs. HouseExpo), completion architecture (ResNet VAE vs. residual U-Net), information gain computation (flood-fill over predicted free cells vs. free-space ratio r_f), planner

integration (exploration transform injected into planner internals vs. single-layer frontier cost augmentation), and simulation realism (Stage with ground-truth localization vs. Gazebo with SLAM). The use of ground-truth localization in [15] removes a significant source of uncertainty that is present in real deployments, which may inflate their reported exploration efficiency. Furthermore, their approach requires substantially more training data (1.5 million samples from KTH) compared to the frontier patch datasets used here, raising questions about scalability to new environments. A controlled study on a common benchmark with realistic SLAM would clarify which design choices most strongly drive exploration efficiency.

9.3 Exploration Pipeline Extensions

9.3.1 Late-Stage Exploration Strategy

The 85–90% coverage band is the most contested in the experiments: two environments (`10161ca6` and `3a4b51bf`) show statistically significant BASE advantage in this band. This corresponds to the transition from the main exploration phase to the final clean-up of isolated unknown pockets. A hybrid scoring strategy that switches from completion-based scoring to a pure distance-or-size heuristic when coverage exceeds a threshold (e.g., 85%) could address this mid-range reversal without affecting performance at the more impactful earlier milestones.

One caveat is that mid-run policy switching may not behave additively with prior exploration decisions. The frontier set and the robot's position at 85% coverage depend on all prior selection decisions; switching to a different policy at that point produces a trajectory that is neither a pure MC run nor a pure BASE run. Controlled experiments comparing MC-only, BASE-only, and hybrid (MC then BASE) conditions across multiple environments would be necessary to establish whether the hybrid policy reliably outperforms both pure alternatives.

Beyond 95% coverage, the remaining unknown cells correspond to geometrically constrained corner pockets that frontier-based exploration deliberately defers. A global map completion pass could be triggered at this stage—functioning more like inpainting on a nearly complete map—either offline or in a semi-automated human-in-the-loop setting where an operator confirms or corrects the completed map before the robot resumes.

A coverage-adaptive variant would make this threshold dynamic rather than fixed: by estimating the total floor area from the predicted global map or from a prior such as building plans, the system could activate the hybrid strategy when estimated coverage progress exceeds a computed threshold rather than a hardcoded

percentage. This would allow the transition point to adapt to environments of different sizes and topologies, potentially recovering the MC advantage in the 85–90% band more reliably than a fixed threshold.

9.3.2 Geometric Filtering Heuristics

The interior noise, exterior noise, trapped-corridor, thin-bridge, and small corner-pocket filters implemented in the codebase were disabled for the reported experiments to isolate the effect of completion. Each filter assigns a cost penalty to flagged frontiers rather than discarding them permanently, preserving fallback targets in frontier-sparse situations. Evaluating these heuristics in a dedicated ablation study would quantify their independent contribution to the exploration pipeline and determine whether their combination with completion provides additive or complementary benefits.

9.3.3 Map Completion–Aware Global Path Planning

The current system uses completion predictions only at the frontier selection stage; the global A* planner operates on the unmodified SLAM costmap. A natural extension is to expose the completed map as an additional cost layer within the navigation stack, allowing the planner to route through corridors the model predicts as open rather than treating all unknown cells as equally traversable.

To preserve the frontier-level spatial reasoning already developed in this work, the same per-frontier completed patches could serve a dual role: scoring frontiers *and* seeding a local completion layer around the planned path. This would avoid the overhead of a full global completion pass while still providing the planner with structural context near the selected goal. The primary constraint is computational: any additional per-cycle map processing must remain within the planning loop latency budget, which already limits the scoring pipeline on the current evaluation platform. A faster embedded compute board is one enabler for this direction; alternatively, lightweight approximations—such as restricting late-stage scoring to a reduced frontier subset or estimating uncertainty from a single forward pass via an auxiliary prediction head—could bring these components within budget without requiring a hardware upgrade.

9.3.4 Synchronisation-Aware Closed Live Map

A synchronization mechanism that gates the closing step on explicit SLAM update events would resolve the timing issue. Improving the closing algorithm to detect

and re-open gaps that have been filled by the SLAM algorithm would resolve the divergence issue. Both changes are well-defined engineering tasks.

It is worth noting that if global planner parameter tuning—such as adjusting costmap resolution and obstacle inflation radii—proves sufficient to prevent the planner from routing through sub-pixel wall gaps, the gap-closing operation and its associated republication overhead may not be necessary at all. If parameter tuning alone is insufficient, however, a lightweight closing strategy restricted to locally updated regions would be the next step, with a more capable compute platform as a fallback. Once reliable and within budget, the closed live map could serve as the primary costmap for both path planning and frontier detection, replacing the raw SLAM map throughout the pipeline. The expected benefit is geometrically cleaner frontier cells and fewer spurious gaps.

9.4 Generalisation and Deployment

9.4.1 Physical Robot Validation

All experiments in this thesis are conducted in Gazebo simulation. Validating the system on a physical robot platform with real SLAM noise, sensor limitations, and mechanical uncertainty is the most important next step for establishing practical relevance. The modular design of the pipeline facilitates this transition: the completion service, the extended `explore_lite` node, and the `move_base` stack can all be deployed on a real TurtleBot3 or equivalent platform without architectural changes.

Key challenges in physical deployment include map coordinate alignment between the floor plan oracle and the SLAM map, SLAM drift accumulation over long missions, and increased sensor noise that may degrade both frontier patch quality and the reliability of the completion model's predictions.

9.4.2 Online Evaluation of Map Completion Quality

The frontier-centered completion models are evaluated offline on held-out patches collected from SLAM exploration trajectories. Patches encountered during live exploration may differ systematically from the test distribution. A dedicated online evaluation protocol that measures completion quality on patches extracted during active exploration runs would provide a more direct assessment of deployment-time model behaviour.

9.4.3 Multi-Environment Generalisation

The current evaluation covers eight HouseExpo environments. Extending the evaluation to a broader set—including more non-Manhattan layouts, irregular room shapes, and multi-floor structures—would provide a more comprehensive characterization of the system's generalization boundaries and would enable the environment-topology analysis needed to ground the adaptive patch sizing strategy described in Section 9.1.1.

9.4.4 Sensor Modality Extension

The current system uses 2D LiDAR-based SLAM and produces single-channel occupancy grids. Extending the completion model to condition on additional modalities—RGB images, depth maps, or semantic labels—could provide richer spatial priors for environments where pure occupancy information is ambiguous.

9.4.5 Dynamic Environments

The static-environment assumption is a fundamental limitation of the present system. Extending it to dynamic environments would require a mechanism to detect changed regions and invalidate or re-generate predictions in those regions. A temporal model that explicitly conditions on the difference between consecutive map snapshots—motivated by the t and $t{+}3$ pair used in dataset construction—could provide a natural framework for this extension. Rather than treating each snapshot independently, such a model would maintain a representation of how the map evolves over time, learning to associate incremental changes with exploration progress. In the static environments considered here, map changes arise solely from robot motion as previously unknown cells become observed; a temporal model could exploit this structured growth pattern to produce more consistent predictions across successive snapshots. Extending this to environments with dynamic obstacles would further require distinguishing robot-motion updates from state changes in already-observed regions, but this remains a longer-term challenge beyond the scope of the current static-world assumption.

9.4.6 PseudoSLAM–ROS Integration for Accelerated Data Collection

The HouseExpo floor plans were used in this thesis via a static conversion pipeline and a full Gazebo/ROS simulation stack. The HouseExpo dataset is distributed together with PseudoSLAM, a lightweight Python-based simulator that bypasses full SLAM computation by operating directly on ground-truth maps [3].

PseudoSLAM achieves map-building speeds substantially faster than GMapping in Stage, making it attractive for large-scale data generation.

However, PseudoSLAM currently lacks a ROS interface: it exposes no `nav_msgs/OccupancyGrid` topics, no `tf` coordinate frames, and no `move_base` navigation stack, making direct integration with the exploration pipeline developed in this thesis infeasible without a bridge layer. Developing such a bridge—mapping PseudoSLAM's occupancy output to standard ROS message types and exposing a compatible navigation interface—would enable large-scale frontier patch collection at substantially lower computational cost than full Gazebo simulation, since PseudoSLAM bypasses physics-based rendering while remaining compatible with the existing ROS infrastructure. This would significantly expand the training data available for data-hungry models such as RePaint and LaMa. Beyond data volume, PseudoSLAM's built-in configurable features—including user-defined sensor noise models, adjustable laser range and field of view, resolution scaling, and support for dynamic obstacles specified via user-provided trajectories—could further increase the diversity and realism of the training corpus.

9.4.7 Coverage-Based Dataset Construction

The global sequential map dataset currently pairs snapshots at fixed temporal offsets ($t \rightarrow t{+}3$). An alternative construction strategy—motivated by the interval-based sampling used in [17]—would pair maps at fixed *coverage* intervals (e.g., every 10% of explored area). This would produce semantically more consistent input–target pairs across environments with different topologies and exploration speeds. Additionally, the current temporal pairing requires careful metric-based or manual filtering to discard near-identical consecutive snapshots where little new area has been explored between t and $t{+}3$; coverage-based sampling would eliminate this step by construction, as each pair is guaranteed to differ by a meaningful increment of explored area.

9.5 Longer-Term Research Directions

9.5.1 Semantic and Task-Aware Completion

The current models predict occupancy (free/occupied/unknown) without semantic understanding. Incorporating semantic labels—room type, object category, surface material—into the completion output would allow the robot to reason not only about *where* space is likely to be but also about *what* it is likely to contain. This would

enable task-aware exploration strategies that prioritise regions predicted to contain task-relevant objects or areas. This direction represents a substantial expansion of scope beyond the current thesis and is most appropriately considered as a long-term research program rather than an incremental extension.

9.5.2 Multi-Robot and Collaborative Completion

In multi-robot exploration, each robot observes a different sub-region of the environment. Sharing completion predictions across robots would allow them to benefit from each other's local observations even before maps are explicitly merged. The generative entropy-based approach of Spinos et al. provides a related framework [17]; combining their global prediction strategy with the local patch-based approach of this thesis could yield a system that is both computationally efficient and spatially comprehensive.

9.6 Summary

The most impactful near-term directions are: (i) training and evaluating the UNet at 64×64 patch resolution across all eight environments, informed by broader environment characterisation; (ii) investigating global planner parameter tuning and synchronization-aware gap closing to enable tighter pipeline coupling without the reliability risks identified in the oracle study; and (iii) validating the system on a physical robot platform. Medium-term directions include adaptive patch sizing, multi-scale architectures, late-stage hybrid scoring strategies, and richer scoring terms on faster hardware. A direct experimental comparison with the information-gain-based exploration framework of [15] on a common benchmark with realistic SLAM represents a separate open benchmarking direction. Longer-term directions include semantic completion and multi-robot collaborative mapping. Together, these extensions would move the system from a simulation-validated research prototype toward a deployable, general-purpose completion-aware exploration framework.

10
CONCLUSION

This thesis has investigated learned map completion—the prediction of unobserved spatial regions from partial occupancy grids—as a standalone inference problem and as an integrated component of frontier-based autonomous robot exploration, evaluating its downstream effect on exploration efficiency. This research contributes across several interrelated dimensions: a modular dataset and preprocessing pipeline for realistic occupancy map collection; a comprehensive comparative evaluation of generative model families under unified FULL and MASKED protocols; a completion-aware frontier scoring mechanism integrated into a ROS-based exploration pipeline; and a system-level experimental evaluation across eight simulated environments demonstrating a measurable exploration efficiency gain.

10.1 Summary of Contributions

10.1.1 Contribution 1: Dataset and Preprocessing Pipeline

Two complementary dataset generation pipelines were developed and released publicly [4, 5]. The static pipeline converts HouseExpo floor plans into occupancy grids via a lightweight mathematical transformation, producing 97,050 training patches for controlled experiments (upper-to-lower formulation). The simulation-based pipeline deploys a TurtleBot3 in Gazebo, collects sequential SLAM maps at Voronoi crossing points, and extracts frontier-centered patch pairs, yielding a global dataset and two frontier-centered patch datasets. Both pipelines enforce environment-level train/val/test splitting to prevent data leakage.

10.1.2 Contribution 2: Comprehensive Model Evaluation

A broad range of model families were implemented, trained, and evaluated under multiple completion formulations including a training-free geometric baseline. The evaluation adopted a task-appropriate protocol computing metrics exclusively over

originally unknown cells as the primary criterion for frontier-centered models. Key findings across formulations are:

- **Upper-to-lower:** CVAE substantially outperforms ImageGPT with a $>$ $3000\times$ inference advantage.
- **Global:** UNet (3+3) achieves the best aggregate accuracy; the training-free GEOM baseline attains the highest SSIM under FULL evaluation, demonstrating the competitiveness of geometric priors under rectilinear layouts—though this SSIM advantage does not hold under the MASKED protocol.

  Frontier-centered (MASKED): UNet variants dominate consistently across both dataset sizes. CVAE fails under irregular-mask conditions; LaMa (12.67 img/s) and $\text{ImageGPT}_{\text{local}}$ ($\approx$0.05 img/s) are excluded from deployment consideration due to insufficient throughput; RePaint, despite passing the throughput threshold (35.43 img/s), is unsuitable due to its iterative denoising cost and high variance under limited training data; LaMa's large receptive field additionally provides limited benefit at 32×32 scale; GAN variants are competitive but prone to hallucination and training instability; ViT's coarse patch tokenization produces discontinuous wall predictions at 32×32 resolution.

The analysis establishes that frontier-centered local completion is a geometry-dominant problem: UNet performance is stable across SDS and BDS, suggesting that models with strong 2D convolutional inductive biases are relatively insensitive to moderate increases in training data—though substantially larger datasets may still yield marginal gains. For models such as RePaint and LaMa that are known to scale with dataset size in natural image domains, future work with substantially larger frontier patch datasets may reveal whether data-hungry models can eventually close the gap with UNet.

10.1.3 Contribution 3: Frontier Scoring with Predicted Information Gain

A completion-aware frontier scoring mechanism was developed and integrated into an extended `explore_lite` ROS package. The mechanism extracts a 32×32 patch centred on each candidate frontier, completes it using the deployed UNet via a batch ROS service, and computes a free-space gain score r_f (Chapter 4) that augments the classical distance-size cost function. A predicted-on-wall flag

penalises frontiers whose neighbourhood is predicted to lie inside a wall, placing them below all feasible alternatives without permanently discarding them.

Optional pipeline components—frontier clustering, gap closing, and filtering heuristics—are implemented as modular stages controlled by launch parameters, enabling clean head-to-head comparison between the extended and baseline configurations.

10.1.4 Contribution 4: System-Level Integration and Evaluation

The modular pipeline was evaluated across eight HouseExpo environments at 56 coverage milestone comparisons (8 environments $\times$ 7 milestones, 3 runs each). The UNet-based MC system outperforms the classical baseline in 80.4% of comparisons, with an average speedup of 7.2% and a consistent advantage at the 95% milestone across all environments ($+10.4\%$, $p < 0.001$). At maximum coverage, the two methods converge, confirming that MC improves *efficiency*, not final coverage extent.

A supplementary GT oracle navigation study [8] recorded a $+48\%$ reduction in navigation duration under ideal local completion with wall closing and frontier filtering, conducted on a single environment across four independent SLAM runs. This result is reported as an exploratory observation rather than a performance target for the UNet system: it demonstrates that the integration architecture is capable of producing substantial navigation benefit under ideal conditions, and provides a concrete motivation for future work on tighter pipeline coupling. The improvement is partly attributable to SLAM–path-planner artifacts arising from narrow wall gaps, and the study scope is limited to a single environment. The detailed breakdown of contributing factors is discussed in Chapter 5.

10.2 Key Findings

The following findings emerge from the combined model evaluation and system-level experiments:

1. **Convolutional inductive biases outperform attention and sequence models at** 32×32 **scale.** UNet dominates ViT, LSTM, and ImageGPT in MASKED evaluation regardless of dataset size, confirming that local rectilinear occupancy structure provides adequate signal for 2D convolutional models to produce exploration-useful completions without requiring large training sets or global context.

2. **Completion improves exploration efficiency but not final coverage.** MC reaches each milestone faster than BASE up to 95%, but the two methods converge at maximum coverage. The benefit is in *efficiency*, not the ultimate extent of exploration.

3. **The 95% milestone is the most robust indicator of completion benefit.** MC wins at 95% in all eight environments ($p < 0.001$), reflecting that late-stage exploration—locating the last unknown pockets—is precisely the task for which local patch completion is designed.

4. **Patch scale must match environment topology.** The `4bb74477` outlier shows that a prediction window smaller than the relevant structural feature provides no benefit and can actively mislead frontier selection. The GT oracle at 64×64 recovers an 11–14% advantage in the same environment, confirming that the bottleneck is predictable at the appropriate scale. Given the close correspondence between the GT oracle and UNet results at 32×32, a similar improvement is anticipated for a UNet trained at 64×64, though this requires empirical validation.

5. **Wall closing and frontier filtering contribute to navigation benefit in the oracle setting.** The single-environment oracle observation suggests that frontier scoring alone accounts for approximately 36% improvement in navigation duration, and the addition of wall closing with filtering raises this to 48%. These components remain candidates for future integration once the synchronisation and reliability issues identified in Chapter 5 are addressed.

10.3 Closing Remarks

This thesis has demonstrated that scene extrapolation is not merely an image completion problem applied to occupancy grids, but a robotics problem whose value must be measured by its downstream effect on exploration. The evaluation framework developed here, centred on task-appropriate model selection criteria and milestone-based elapsed time for system evaluation, provides a principled methodology for assessing completion-based exploration systems that can be adopted and extended by the research community. The open-source release of the dataset generation pipeline, the frontier-centered patch datasets, and the complete codebase—including the extended `explore_lite` node and the UNet batch completion service—is intended to facilitate reproducibility and to lower the barrier for future work on prediction-guided robotic exploration. The modular design ensures that each component—completion model, scoring function, gap closing,

filtering—can be replaced or extended independently, allowing the system to serve as a testbed for new completion models, scoring strategies, and integration policies.

In summary, this research establishes that occupancy map completion from partial observations is both a tractable inference problem and a practical enhancement to frontier-based exploration: unobserved regions can be reliably predicted from observed structure, and when the completion model is selected using task-appropriate evaluation protocols, integrated at the correct abstraction level, and evaluated with system-level metrics rather than isolated quality scores, these predictions translate into measurable gains in exploration efficiency.

REFERENCES

[1] S. Güzel, E. Uslu, "Map completion using generative models on a custom dataset," *Signal, Image and Video Processing*, vol. 19, p. 878, 2025. doi: 10.1007/s11760-025-04431-x.

[2] ——, "Global completion of occupancy maps using generative models," *Electronic Letters on Computer Vision and Image Analysis*, vol. 24, pp. 273–284, 2026.

[3] Y. Chen, P. Sun, S. Lee, Z. Li, X. Chen, A. Yu, P. Guo, X. Xie, "Houseexpo: A robotic indoor navigation dataset for vision-based exploration and mapping," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1956–1963. doi: 10.1109/IROS40897.2019.8968274.

[4] S. Güzel, E. Uslu, *Houseexpo_based_custom_dataset_with_gazebo*, https://github.com/f0120050/houseexpo_based_custom_dataset_with_gazebo, Accessed: 2025-05-14, 2025.

[5] ——, *Houseexpo_based_custom_map_generation_simple*, https://github.com/f0120050/houseexpo_based_custom_map_generation_simple, Accessed: 2025-05-14, 2025.

[6] R. N. S. Maintainers, *explore_lite: Lightweight frontier-based exploration for ros*, https://github.com/hrnr/m-explore, Accessed: 2025-05-15, 2020.

[7] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, I. Sutskever, "Generative pretraining from pixels," *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 1691–1703. doi: 10.48550/arXiv.2006.11216.

[8] S. Güzel, E. Uslu, "Evaluating the effect of local map completion on navigation," *Proceedings of the International Conference on Applied Robotics and Automation (ICARA)*, 2025.

[9] S. Güzel, E. Uslu, S. Yavuz, "Occupancy grid map completion based on geometric features and door discovery," *Proceedings of the 12th International Symposium on Digital Forensics and Security (ISDFS)*, 2024, pp. 1–6. doi: 10.1109/ISDFS60797.2024.10527317.

[10] S. Güzel, S. Yavuz, "Survey on learning-based scene extrapolation in robotics," *International Journal of Intelligent Robotics and Applications*, vol. 8, no. 1, pp. 251–268, 2024.

[11] H. Q. H. Gao, H. Au, W. Shan, M. Liu, Y. Qin, L. Mu, X. Y. R. Zhao, Q. Wei, F. Qiao, "Senseexpo: Efficient autonomous exploration with prediction information from lightweight neural networks," 2025. https://doi.org/10.48550/arXiv.2503.16000, 2025.

[12] C. Ho, S. Kim, B. Moon, A. Parandekar, N. Harutyunyan, C. Wang, K. Sycara, G. Best, S. Scherer, "MapEx: Indoor structure exploration with probabilistic information gain from global map predictions," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025. doi: 10.48550/arXiv.2409.15590.

[13] K. Katyal, K. Popek, C. Paxton, P. Burlina, G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5453–5459. doi: 10.1109/ICRA.2019.8793500.

[14] K. Katyal, K. Popek, C. Paxton, J. Moore, K. Wolfe, P. Burlina, G. D. Hager, "Occupancy map prediction using generative and fully convolutional networks for vehicle navigation," *arXiv preprint arXiv:1803.02007*, 2018. [Online]. Available: https://arxiv.org/abs/1803.02007.

[15] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1197–1204. doi: 10.1109/ICRA.2019.8793769.

[16] A. Elhafsi, B. Ivanovic, L. Janson, M. Pavone, *Map-predictive motion planning in unknown environments*, 2020.

[17] A. Spinos, B. Woosley, J. Rokisky, C. Korpela, J. G. Rogers III, B. A. Bittner, "Map prediction and generative entropy for multi-agent exploration," *arXiv preprint arXiv:2501.13189*, 2025. doi: 10.48550/arXiv.2501.13189.

[18] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, K. Daniilidis, "Uncertainty-driven planner for exploration and navigation," *Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA: IEEE, 2022, pp. 11 295–11 302. doi: 10.1109/ICRA46639.2022.9812423.

[19] Y. Katsumata, A. Kanechika, A. Taniguchi, L. E. Hafi, Y. Hagiwara, T. Taniguchi, "Map completion from partial observation using the global structure of multiple environmental maps," *Advanced Robotics*, vol. 36, no. 5–6, pp. 279–290, 2022.

[20] L. Davies, B. Li, M. Saada, "Occupancy grid artefact removal and error correction using gans," *Proceedings of the 4th International Conference on Computer, Control and Robotics*, 2024.

[21] M. Luperto, F. Amadelli, M. Di Berardino, F. Amigoni, "Mapping beyond what you can see: Predicting the layout of rooms behind closed doors," *Robotics and Autonomous Systems*, vol. 159, p. 104 282, 2023, issn: 0921-8890. doi: 10.1016/j.robot.2022.104282. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022001713.

[22] S. K. Ramakrishnan, Z. Al-Halah, K. Grauman, "Occupancy anticipation for efficient exploration and navigation," *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, Springer, 2020, pp. 400–418. doi: 10.1007/978-3-030-58523-5_24.

[23] S. S. Y. Liang B. Chen, "Sscnav: Confidence-aware semantic scene completion for visual semantic navigation," pp. 13194–13200 (2021), 2021.

[24] M. Narasimhan, E. Wijmans, X. Chen, T. Darrell, D. Batra, D. Parikh, A. Singh, "Seeing the un-scene: Learning amodal semantic maps for room navigation," *Computer Vision – ECCV 2020*, 2020, pp. 513–529.

[25] S. Ramakrishnan, D. Jayaraman, K. Grauman, "Emergence of exploratory look-around behaviors through active observation completion," *Science Robotics*, vol. 4, 2019.

[26] Z. Yang, J. Pan, L. Luo, X. Zhou, K. Grauman, Q. Huang, "Extreme relative pose estimation for rgb-d scans via scene completion," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] G. Georgakis, K. Schmeckpeper, K. Wanchoo, S. Dan, E. Miltsakaki, D. Roth, K. Daniilidis, "Cross-modal map learning for vision and language navigation," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] S. Song, A. Zeng, A. Chang, M. Savva, S. Savarese, T. Funkhouser, "Im2pano3d: Extrapolating 360° structure and semantics beyond the field of view," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3847–3856.

[29] Y. S. A. Abbasi S. Kalkan, "Deep 3d semantic scene extrapolation," *The Visual Computer*, 2019, vol. 35 (2019).

[30] S. Gupta, D. Fouhey, S. Levine, J. Malik, *Unifying map and landmark based representations for visual navigation*, arXiv preprint, 2017.

[31] A. Aydemir, P. Jensfelt, J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2012.

[32] L. Co, "Lifull home's dataset," *Informatics Research Data Repository*, 2015, 2015.

[33] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, S. Savarese, "Gibson Env: Real-world perception for embodied agents," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9068–9079. doi: 10.1109/CVPR.2018.00945.

[34] A. Chang, A. Makadia, D. Savva, R. Gupta, S. M. Avidan, J. H. T. Kim, J. Malik, "Matterport3d: Learning from rgb-d data in indoor environments," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 3200–3209. doi: 10.1109/ICCV.2017.344.

[35] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, T. Funkhouser, "Semantic scene completion from a single depth image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 190–198. doi: 10.1109/CVPR.2017.28.

[36] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2432–2443. doi: 10.1109/CVPR.2017.261.

[37] J. Xiao, K. Ehinger, A. Oliva, A. Torralba, "Recognizing scene viewpoint using panoramic place representation," *2012 IEEE Conference On Computer Vision And Pattern Recognition*, pp. 2695–2702 (2012), 2012.

[38] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, "3d shapenets: A deep representation for volumetric shapes," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[39] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, "3d semantic parsing of large-scale indoor spaces," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1534–1543. doi: 10.1109/CVPR.2016.170.

[40] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, R. Salakhutdinov, "Learning to explore using active neural slam," *International Conference on Learning Representations (ICLR)*, 2020.

[41] G. Georgakis, B. Bucher, A. Arapin, K. Schmeckpeper, N. Matni, K. Daniilidis, "Uncertainty-driven planner for exploration and navigation," *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 11 295–11 302. doi: 10.1109/ICRA46639.2022.9812423.

[42] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[43] R. O. Duda, P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972. doi: 10.1145/361237.361242.

[44] X. Y. K. Sohn H. Lee, "Learning structured output representation using deep conditional generative models," 2015. https://arxiv.org/abs/1506.05587, 2015.

[45] D. P. Kingma, M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. doi: 10.1561/2200000056. [Online]. Available: http://dx.doi.org/10.1561/2200000056.

[46] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1125–1134. doi: 10.1109/CVPR.2017.632.

[47] J. Ho, A. Jain, P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv: 2006.11239 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2006.11239.

[48] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, L. Van Gool, "RePaint: Inpainting using denoising diffusion probabilistic models," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 461–11 471. doi: 10.1109/CVPR52688.2022.01117.

[49] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.

[50] O. Ronneberger, P. Fischer, T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer, 2015, pp. 234–241.

[51] R. Suvorov, A. Mashikhin, E. Logacheva, A. Ashukha, T. Khakhulin, V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 2149–2159. doi: 10.1109/WACV51458.2022.00323.

[52] C. Olah, *Understanding LSTM networks*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, Accessed: 2025, 2015.

[53] Y. Yamauchi, "Frontier-based exploration for autonomous robots," *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, IEEE, 1997, pp. 146–151. doi: 10.1109/CIRA.1997.615442.

[54] R. Padilla, S. L. Netto, E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242. doi: 10.1109/IWSSIP48289.2020.9145130.

[55] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, K.-R. Müller, Eds., Berlin, Heidelberg: Springer, 2012, pp. 437–478. doi: 10.1007/978-3-642-35289-8_26.

[56] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. doi: 10.2307/3001968.

[57] J. D. Gibbons, S. Chakraborti, *Nonparametric Statistical Inference*, 5th. Boca Raton, FL: CRC Press, 2011, isbn: 978-1-4200-7762-9.

[58] W. H. Kolence, P. J. Kiviat, "Software unit profiles and Kiviat figures," *ACM SIGMETRICS Performance Evaluation Review*, vol. 2, no. 3, pp. 2–12, 1973. doi: 10.1145/1113644.1113647.

PUBLICATIONS FROM THE THESIS

Papers

1. S. Güzel and S. Yavuz, "Survey on Learning-Based Scene Extrapolation in Robotics," *International Journal of Intelligent Robotics and Applications*, vol. 8, pp. 251–268, 2024. https://doi.org/10.1007/s41315-023-00303-0

2. S. Güzel and E. Uslu, "Map Completion Using Generative Models on a Custom Dataset," *Signal, Image and Video Processing*, vol. 19, p. 878, 2025. https://doi.org/10.1007/s11760-025-04431-x

3. S. Güzel and E. Uslu, "Global Completion of Occupancy Maps Using Generative Models," *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)*, vol. 24, pp. 273–284, 2026.

Conference Papers

1. S. Güzel, E. Uslu, and S. Yavuz, "Occupancy Grid Map Completion Based on Geometric Features and Door Discovery," *Proceedings of the 12th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–6, 2024. https://doi.org/10.1109/ISDFS60797.2024.10527317

2. S. Güzel and E. Uslu, "Evaluating the Effect of Local Map Completion on Navigation," *Proceedings of the International Conference on Applied Robotics and Automation (ICARA)*, 2025.