



Scalable and generalizable path planning for robotic navigation using transformer-based heuristic learning

Elie Thellier , Adolfo Perrusquía* , Antonios Tsourdos 

Faculty of Engineering and Applied Sciences, Cranfield University, Bedford MK43 0AL, UK

ARTICLE INFO

Keywords:

Heuristic learning
Correction factor
Path probability map
Weighted A*

ABSTRACT

Efficient and scalable path planning is a critical challenge for autonomous robotic systems, particularly in complex real-world scenarios. Traditional heuristic search algorithms like A* often struggle with scalability and adaptability in such environments. To address these limitations, we improve a search framework that integrates learned, instance-specific heuristics with conventional pathfinding techniques. Leveraging autoencoder transformer networks, we predict two key heuristic functions—Correction Factor (CF) and Path Probability Map (PPM)—trained on diverse datasets—the Motion Planning (MP) and Tiled-MP datasets—to cover a wide range of path planning scenarios. When integrated with Weighted A* (WA*) algorithm, this approach optimally solves 88% of MP instances, with paths averaging less than 0.7% longer than optimal, and requiring nearly five times fewer node expansions. The framework demonstrates the advantages of heuristic learning in handling larger path planning problems, with inference time accounting for just 10% of the total search duration. It solves nearly half of the most complex instances optimally, showcasing strong scalability for real-time robotics applications. The framework performs well in unseen environments, solving over 25% of new problems perfectly, finding near-optimal solutions with paths less than 7% longer than optimal, and requiring fewer than two-thirds of the typical expansions. Our framework outperforms learnable planners in both scalability and generalization.

1. Introduction

Path planning is a critical component of robotic navigation, enabling autonomous systems to find efficient routes in complex environments. For mobile robots operating in real-world scenarios such as search and rescue, autonomous driving, or exploration missions—efficient path planning is essential to ensure timely and resource-efficient operation, especially in dynamic and obstacle-dense settings [1,2].

Robots deployed in environments such as disaster zones or cluttered indoor spaces must navigate efficiently around obstacles while optimizing for criteria like travel time, energy consumption, and safety. For example, in post-disaster scenarios, efficient path planning allows robots to locate survivors quickly by minimizing search time and unnecessary expansions [3]. Similarly, optimizing UAV flight paths in complex terrains can significantly reduce energy consumption and extend mission endurance [4].

Traditional search-based path planning methods, such as Dijkstra's and A* [5,6] have been widely used but often struggle with scalability in large, complex environments. These algorithms tend to be computationally expensive, making real-time navigation infeasible in scenarios with numerous obstacles or highly dynamic conditions [7]. Additionally, they typically rely on static, instance-independent heuristics like Manhattan distance, which may perform poorly in environments with irregular obstacle distributions, leading to suboptimal or excessively long paths.

* Corresponding author.

Email address: adolfo.perrusquia-guzman@cranfield.ac.uk (A. Perrusquía).

<https://doi.org/10.1016/j.ins.2026.123149>

Received 14 September 2025; Received in revised form 23 January 2026; Accepted 24 January 2026

Available online 24 January 2026

0020-0255/© 2026 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

To overcome these limitations, more adaptive heuristics are needed that incorporate information from the structure of each specific environment rather than relying solely on static, hand-crafted estimates. In this context, Path Probability Maps (PPM) and Correction Factors (CF) provide two complementary mechanisms for generating instance-dependent heuristic guidance through learned models. A Path Probability Map offers a global, data-driven estimate of which regions of the environment are more likely to lie on an optimal path. By learning spatial patterns from numerous training environments, PPM provides a coarse but informative representation of navigation difficulty across the entire map. However, PPM alone may lack the local precision required when dealing with tight passages, sharp corners, or densely clustered obstacles. The Correction Factor, in contrast, provides a localized refinement that adjusts the global PPM-derived heuristic based on finer-scale environmental cues. CF compensates for local irregularities that probability-based global models may overlook, enabling more accurate heuristic estimation near complex obstacle boundaries. Thus, while PPM supplies broad global structure, CF enhances local sensitivity, ensuring the heuristic remains relevant at both large and small spatial scales.

Motivated by their complementary strengths, combining PPM and CF enables the construction of heuristics that are both globally informed and locally precise. This integrated approach reduces unnecessary node expansions, improves scalability, and enhances generalization across diverse environments without requiring extensive manual tuning. This paper builds on the concept of instance-dependent heuristics by proposing a framework that improves scalability and generalization for robotic path planning. By leveraging deep learning-based heuristic prediction, our approach aims to reduce computational complexity, improve efficiency, and adapt to a wide range of environments, making it suitable for real-time applications in autonomous robotics.

1.1. Related work

Heuristic learning has recently emerged as a promising extension to classical path planning, enabling search-based and sampling-based algorithms to benefit from learned, instance-dependent guidance. Existing work can be broadly categorized into three approaches: path probability map generation, cost-to-go regression, and self-supervised heuristic learning [8].

The path probability map (PPM) approach predicts distributions over grid cells indicating their likelihood of belonging to an optimal path [9]. Fully convolutional networks have been used to generate these maps and reconstruct paths for single or multi-agent settings [10], and subsequent work has explored end-to-end prediction of shortest and lowest-cost paths across various grid sizes [11]. Additional research uses encoder architectures to capture high-level structure in large, cluttered, or dynamic environments, helping search algorithms focus on more relevant spatial regions [12]. However, PPM-based methods typically provide coarse, global predictions, often assuming static environments and single start–goal pairs. Their limited local resolution and sensitivity to map scale can hinder performance in environments with irregular or highly complex obstacle structures.

The cost-to-go regression approach predicts the estimated cost from any state to the goal, which is then used directly as the heuristic in A*-based search. Prior work extends this idea using CNNs, autoencoders, or GAN-based architectures [13,14]. Other studies investigate cost-to-go learning for real-world terrain navigation [15,16]. While effective in reducing node expansions, these methods often struggle when the environment distribution shifts (e.g., out-of-distribution maps), and their predictions can become unreliable in highly cluttered or fine-grained obstacle configurations [17]. Moreover, many regression models require extensive supervised labels or exhibit high memory or runtime overhead.

Self-supervised heuristic learning eliminates the need for manually labeled heuristic values by utilizing differentiable planning frameworks. Neural A* reformulates A* into a differentiable module trained to imitate expert-generated solutions [18], while iA* further introduces a bilevel optimization structure to refine learned guidance maps [19]. Although these methods enhance data efficiency, they remain computationally intensive, rely on differentiable planning approximations, and often fail to generalize effectively to large maps or unseen obstacle distributions.

Across all three categories, existing methods share common limitations:

1. Scalability challenges when map size increases or obstacle density becomes irregular.
2. Limited robustness to distribution shifts, particularly in environments not represented in the training data.
3. Imbalanced heuristic behavior, where models capture either global structure (PPM) or local detail (regression) but rarely both.
4. High computational or memory requirements, which restrict real-time applicability.

This paper addresses these gaps by combining complementary heuristics leveraging both global, probability-driven guidance and locally refined corrective signals—to improve search efficiency, generalization, and scalability across diverse environments.

1.2. Original contributions

This research presents several key contributions that address current limitations in heuristic-based path planning, particularly in scalability, generalization, and efficiency. These contributions are summarized as follows

1. An augmented dataset is designed to enhance both generalization and scalability, featuring a custom size distribution that supports a broader range of environments.
2. We present a synergistic integration of the Path Probability Map (PPM) and Correction Factor (CF), resulting in a novel framework that significantly improves search efficiency by harnessing their complementary strengths while simultaneously mitigating their individual weaknesses.

3. The method enhances robustness to out-of-distribution (OOD) scenarios, mitigating divergence arising from suboptimal heuristic predictions.

The outline of this paper is as follows. [Section 2](#) introduces the preliminaries and problem formulation. [Section 3](#) describes the proposed methodology. [Section 4](#) presents the experiments and discusses the results. Finally, [Section 5](#) concludes the paper and outlines potential directions for future research.

2. Preliminaries

This section outlines the technical definitions and problem formulation.

2.1. Path planning problem

The goal of path planning in a 2D grid-based environment is to find a valid path from a start position to a goal position while avoiding obstacles. We model the environment as a 2D grid map where each cell is either free or blocked, and the agent can only move between adjacent free cells (no diagonal movement) with a movement cost of 1.

A path $\pi(\text{start}, \text{goal})$ is a sequence of adjacent cells from the *start* to the *goal*. A path is considered valid if all cells in the path are free. The cost of a valid path is the sum of the costs between consecutive cells. The optimal path, π^* , is the one with the minimal cost among all valid paths. A sub-optimal path, π^w , is one whose cost is at most w times the cost of the optimal path, where $w > 1$ is the sub-optimality factor.

In this paper, solving the path planning problem means finding either the optimal or a bounded sub-optimal valid path for a specific instance using heuristic-based search algorithms.

2.2. A* algorithm and its variants

The A* algorithm [5] is a popular method for heuristic-based path planning. It iteratively expands nodes in a search space and selects the node with the lowest f -value for expansion, where $f(n) = g(n) + h(n)$. Here, $g(n)$ is the cost of reaching node n from the start, and $h(n)$ is the heuristic estimate of the cost to reach the goal from n .

2.2.1. Weighted A* algorithm

The WA* algorithm [20] introduces a weight factor $w \geq 1$ to adjust the heuristic. The evaluation becomes $f(n) = g(n) + w \cdot h(n)$. By inflating the heuristic, the algorithm reduces the number of nodes expanded, which speeds up the search but sacrifices optimality. For instance, with $w = 2$, the algorithm often performs faster while maintaining solutions within a reasonable sub-optimality bound.

2.2.2. Focal search algorithm

The Focal Search (FS) algorithm [21] enhances the search process by maintaining a secondary list of nodes called FOCAL. This list contains nodes whose f -values are within a factor ϵ of the smallest f -value encountered. Nodes in FOCAL are prioritized using a secondary heuristic, h_{FOCAL} , which guides the search toward promising areas of the search space. The FS algorithm ensures bounded sub-optimality while allowing for flexible control over the computational effort by adjusting ϵ and h_{FOCAL} . However, this added flexibility introduces complexity, as selecting appropriate parameter values is crucial for balancing efficiency and generalizability across diverse problem settings.

2.3. Heuristics

Heuristics play a crucial role in guiding the search process. A heuristic function estimates the cost to reach the goal from a given node. An admissible heuristic never overestimates the true cost, ensuring that the search remains optimal. A consistent heuristic satisfies the triangle inequality, which guarantees that the search remains both optimal and efficient.

Common admissible heuristics include Manhattan distance and Euclidean distance. However, the Manhattan distance does not account for obstacles, often resulting in less efficient search paths.

This paper uses two learned instance-dependent heuristics [22] to improve the efficiency of the search: the Correction Factor (CF) and the Path Probability Map (PPM).

2.3.1. Correction factor

The Correction Factor adjusts the instance-independent heuristic (Manhattan distance) by incorporating the ratio of the Manhattan distance $h(n)$ to the optimal cost-to-go $h^*(n)$. This factor refines the heuristic and improves the accuracy of the search by scaling it within the range $[0, 1]$. The CF is calculated as

$$cf(n) = \frac{h(n)}{h^*(n)} \quad (1)$$

2.3.2. Path probability map

The Path Probability Map represents the likelihood that each grid cell is part of the shortest path between the start and goal. The PPM serves as a secondary heuristic, helping prioritize nodes that are likely to lead to the goal. While generating an accurate PPM can be challenging, combining it with the CF enhances the efficiency of the search, as detailed below.

Algorithm 1 Corrected WA* Search and Focal Search.

```

1: Input: map, start, goal, h, w, cf,  $h_{FOCAL}$ ,  $\epsilon$ 
2: Output: Valid Path  $\pi$ 
3: Initialize OPEN, CLOSED, and FOCAL
4: while OPEN  $\neq \emptyset$  do
5:   sort FOCAL by  $h_{FOCAL}$ 
6:    $n \leftarrow$  node in FOCAL or OPEN with lowest  $f$ -value
7:   remove  $n$  from OPEN and FOCAL, add to CLOSED
8:   if  $f_{min}$  has changed then
9:     update FOCAL using  $\epsilon$ 
10:  end if
11:  if  $n = goal$  then
12:    reconstruct and return path  $\pi$ 
13:  end if
14:  for  $n'$  in validNeighbors(map,  $n$ ) do
15:    if  $n' \in CLOSED$  then
16:      continue
17:    end if
18:    if  $g(n') > g(n) + cost(n, n')$  then
19:       $f(n') \leftarrow g(n') + w \cdot h(n', goal) / cf(n')$ 
20:      update or insert  $n'$  in OPEN
21:      if  $f(n') \leq \epsilon \cdot f_{min}$  then
22:        update or insert  $n'$  in FOCAL
23:      end if
24:    end if
25:  end for
26: end while
27: return failure

```

2.4. Generic search algorithm

Algorithm 1 presents the A* algorithm and its variants: WA* and FS. The base A* algorithm, in black, maintains two primary lists: OPEN (which stores generated but unexpanded nodes) and CLOSED (which stores expanded nodes). Each node has an associated f -value. In each iteration, the node with the lowest f -value is selected from OPEN for expansion.

For the WA* variant, in black and red, the heuristic is scaled by a weight factor w . The *corrected* version of the algorithm, adding the gold color, uses the Correction Factor to adjust the f -value, making the heuristic instance-dependent.

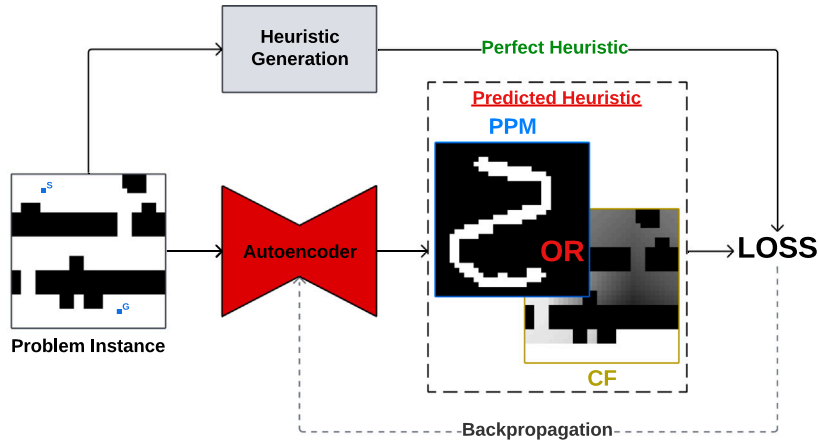
In the FS variant, in black and blue, the FOCAL list prioritizes nodes based on the secondary heuristic h_{FOCAL} . This improves the search efficiency by focusing on nodes that are likely to lead to the goal more quickly.

The algorithm continues by expanding nodes, updating their scores, and re-inserting them into OPEN or FOCAL as necessary, until the goal is reached and the path is reconstructed. Otherwise no path can be found.

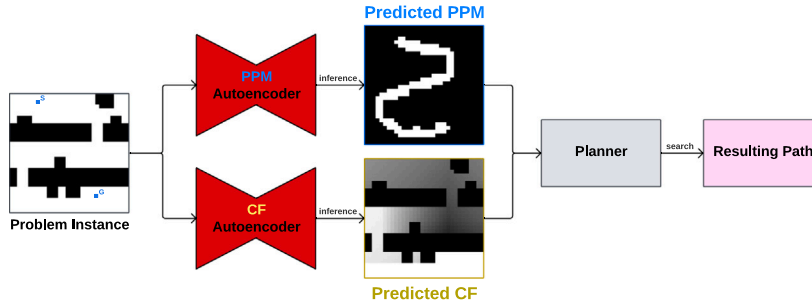
3. Methodology

We propose a framework that integrates two learned heuristics, Correction Factor (CF) and Path Probability Map (PPM), to address challenges in heuristic-based path-finding. Specifically, this framework targets: (i) enhancing scalability across diverse input sizes, (ii) improving generalization to unseen problem instances, and (iii) reducing over-expansion caused by inaccuracies in secondary heuristics $h_{FOCAL} = PPM$. Fig. 1 illustrates the training and inference procedures of the proposed approach.

Fig. 1(a) has two independent autoencoder networks (we discuss their design in the next subsection) that are trained to predict two heuristics: CF or PPM using standard supervised learning. These models are trained on two different datasets: one with a uniform size distribution and another with a varied size distribution. This dual training approach allows us to explore the model's adaptability to different data scales. After training, the framework undergoes a rigorous evaluation across all planners, as shown in Fig. 1(b). This evaluation involves testing the framework on both uniform and non-uniform datasets to assess its scalability and generalization capabilities. In addition, we test the proposed framework on data that differ from the training sets to evaluate its robustness and ability to generalize to new scenarios.



(a) Training Process: Two autoencoders networks are trained to predict CF and PPM heuristics



(b) Inference Process: Generalisation across uniform and non-uniform datasets, as well with out-of-distribution instances

Fig. 1. High-level diagrams of both training and inference of the proposed methodology.

3.1. Autoencoder network

The autoencoder architecture for learning the Correction Factor (CF) and Path Probability Map (PPM) is shown in Fig. 2. The diagram shares a common structure for both tasks, differing only in the input data. Both models take a grid map as a binary image, where cells are marked as free or occupied. For CF prediction, an additional matrix marks the goal position with a single “1”, while for PPM prediction, a secondary matrix marks the start and goal with two “1”s.

The architecture, referred to as *TransAE* (Transformer Autoencoder), consists of a convolutional encoder, spatial transformer blocks, and a convolutional decoder. TransAE CF refers to the CF prediction task and TransAE PPM to the PPM generation. The encoder employs ResNet blocks [23] to extract local features like obstacle shapes and pathways, while spatial transformer blocks, based on self-attention [24], capture global relationships, linking start and goal positions. The decoder reconstructs the transformed feature maps into the output, either a CF or a PPM. The model is lightweight and efficient with just 871,745 parameters and a size of 3400 KB, which is suitable for computationally constrained settings.

The integration of CF and PPM into the WA^* algorithm forms the foundation of the search process, eliminating reliance on the node selection parameter ϵ used in Focal Search. Two heuristic combination methods are developed: $CF + \alpha \cdot PPM$ and CF/PPM . These methods aim to refine the heuristic function for improved performance across diverse environments.

3.2. Datasets

This study employs three datasets: the Motion Planning (MP) dataset, the Tiled-MP dataset, and an Out-of-the-Distribution dataset. Each contains annotated obstacle maps and problem instances tailored for path planning challenges.

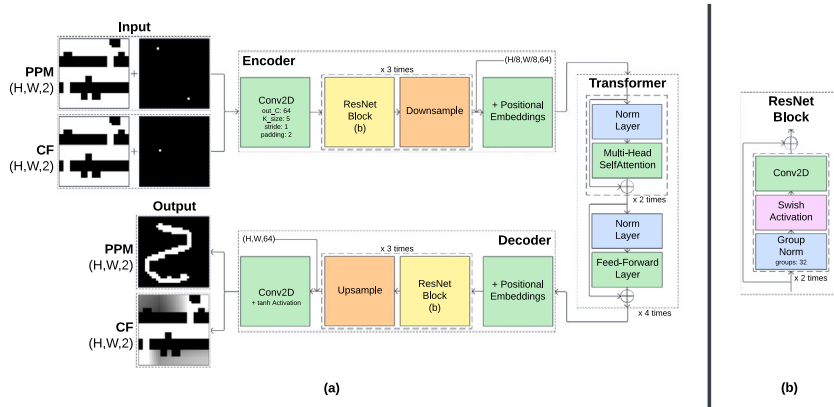


Fig. 2. Overview of the autoencoder architecture. (a) Design of the whole model. (b) Architecture of the ResNet block. Adapted from [22].

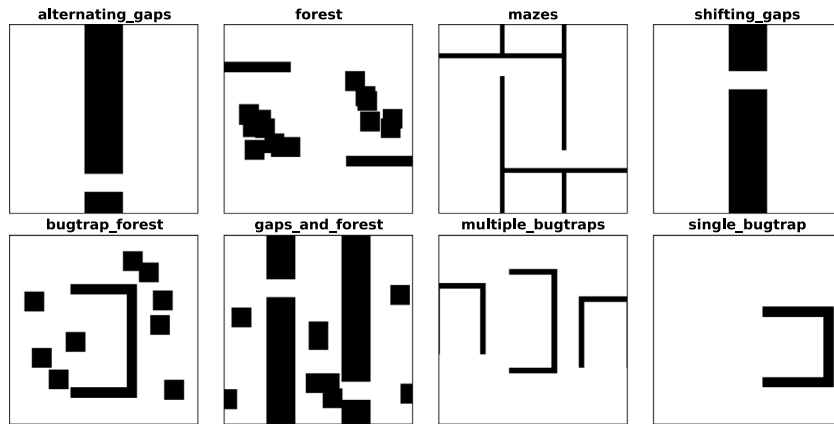


Fig. 3. MP dataset and diversity of maps.

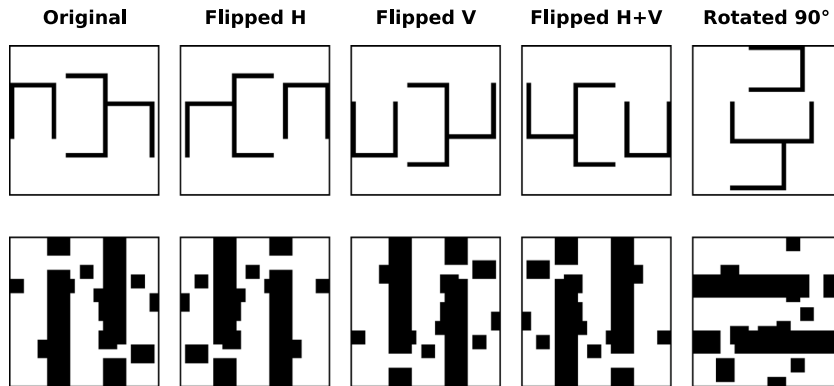


Fig. 4. Augmentation techniques applied in the MP dataset.

3.2.1. Augmented MP dataset

The MP dataset [25] consists of maps with eight distinct obstacle types, including mazes, gaps, and bug traps (as shown in Fig. 3). Each obstacle type has 1000 maps, split into training, validation, and test sets in an 8:1:1 ratio. The original maps (201×201 pixels) were resized to 32×32 pixels for faster processing. Augmentation techniques (see Fig. 4), such as mirroring and rotation, expanded the dataset to 32,000 maps then instance generation with random start and goal positions resulted in 400,000 total samples.

Remark 1. The maps of the MP dataset do not contain complex geometric details or narrow passages, and the down-scaling to 32×32 does not affect the obstacles.

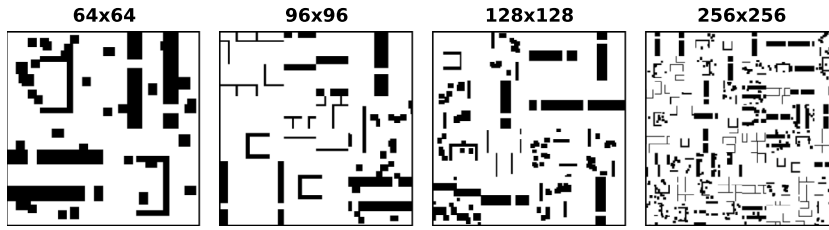


Fig. 5. Tiled-MP dataset with maps of different sizes.

Table 1
Size distribution in the Tiled-MP dataset.

Size	Training Samples	Validation Samples	Test Samples
32×32	160,000	20,000	312
64×64	40,000	5000	312
96×96	17,777	2222	312
128×128	10,000	1250	312
256×256	2500	312	312

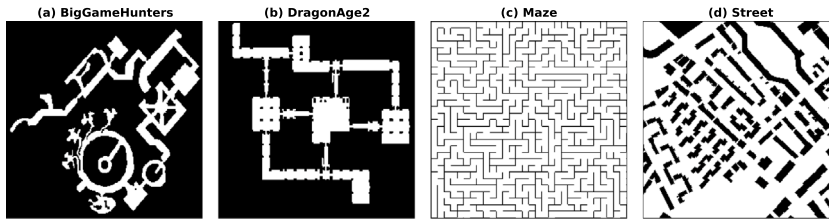


Fig. 6. Examples of the OOD dataset.

3.2.2. Tiled-MP dataset

The Tiled-MP dataset [18] was created by tiling smaller MP maps into larger grids, generating maps of varying sizes (32×32 to 256×256 pixels). This increases the map complexity by creating narrow passages and complex obstacle structures. This dataset evaluates scalability by testing models on larger maps with increased structural diversity. Fig. 5 depicts some examples of the Tiled-MP dataset.

Table 1 provides the dataset's size distribution. Each map includes five problem instances with non-trivial start-goal placements.

3.2.3. Out-of-the-distribution dataset

To test model generalization, we use an out of distribution (OOD) dataset, featuring maps from two video games (Big Game Hunters and Dragon Age 2), mazes, and city layouts. This dataset contains maps ranging from 50×50 to 1024×1024 pixels. Its diversity ensures robust evaluation across varying structures and obstacle densities. Fig. 6 shows some examples of the OOD dataset.

3.2.4. Training on MP and tiled-MP dataset

Because the neural network follows a fully convolutional design, it can handle input images of any dimension. This flexibility enables the same TransAE architecture to be applied to both the MP dataset, which contains uniformly sized samples, and the Tiled-MP dataset, which includes images of varying sizes.

However, training on the Tiled-MP dataset introduces a challenge regarding the sampling batches of images with different dimensions. In other words, there is no simple or efficient way to create mixed-size batches. Although resizing all images to a common resolution could resolve this issue, doing so would undermine the goal of maintaining diverse input sizes. To effectively train the model on data with variable dimensions, two possible strategies can be explored.

1. The first approach is online learning, a simpler variant of batch learning. In this method, the model updates its parameters incrementally as each new sample is processed, instead of waiting to accumulate a full batch. A common simplification is to set the batch size to one, so the model learns from each data point individually. While this technique reduces memory usage and simplifies processing, it also introduces drawbacks: the updates can become noisy, since each single example may poorly represent the overall data distribution. This can result in unstable learning dynamics, slower convergence, and higher computational costs due to frequent updates.
2. The second approach is image padding, which involves adding border pixels around smaller images so that all samples in a batch share the same dimensions. As illustrated in Fig. 7, the padding pixels are excluded from the loss calculation using a mask. Although this method enables batch training, it comes with trade-offs. Padding increases computational load, as the model must process extra, non-informative pixels. Moreover, using constant or zero padding can introduce artifacts that distort image

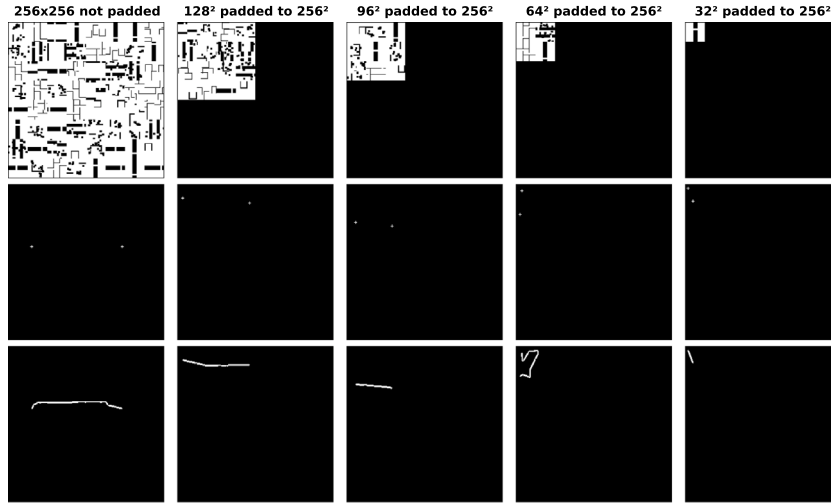


Fig. 7. Examples of input batch padding. The start and goal points in the second row are slightly thickened for visibility.

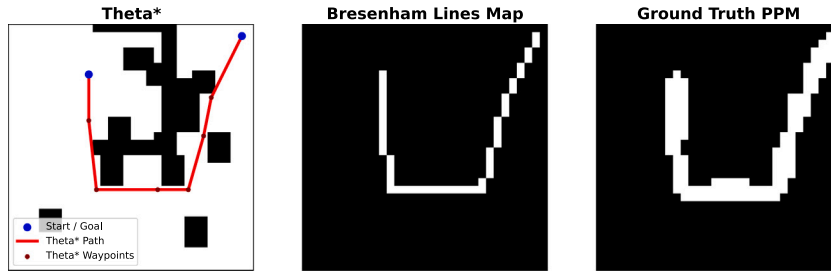


Fig. 8. Steps in the generation of ground truth path probability maps.

features. Finally, padding to different sizes may reduce the model's ability to generalize across varying image dimensions, since inconsistencies in feature extraction can lead to inefficiencies in learning.

In this paper, we decided against using image padding because of its potential drawbacks, such as negatively affecting model performance. Instead, we opted for the simpler method of online learning with a batch size of one, accepting the inevitable increase in the training computational complexity.

3.3. Generating ground truth maps

Ground truth CF and PPM maps are generated for supervised learning in both MP and Tiled-MP datasets.

1. Correction Factor: Derived using backward Dijkstra's algorithm, CF is calculated as the ratio of the Manhattan distance to the Dijkstra-obtained distance for each grid point.
2. Path Probability Map: Generated using the Theta* algorithm [26], which computes a waypoint-based path. Bresenham's line algorithm fills in the path, with neighboring pixels thickened to improve visibility in narrow passages. The steps are illustrated in Fig. 8.

3.4. Instance complexity

The complexity of each problem instance is quantified using the Instance Complexity (IC) metric representing the ratio of the shortest path length (h^*) to the instance-independent heuristic (Manhattan distance, h):

$$IC = \frac{h^*(\text{start})}{h(\text{start})} = \frac{\text{cost}(\pi^*(\text{start}, \text{goal}))}{h(\text{start})} \quad (2)$$

An IC value close to 1 indicates simpler environments with fewer obstacles, while higher values reflect increased complexity due to denser obstacles. Fig. 9 shows different examples of instance complexities.

3.5. Heuristics combination

We propose two methods for combining CF and PPM heuristics to refine the WA^* search:

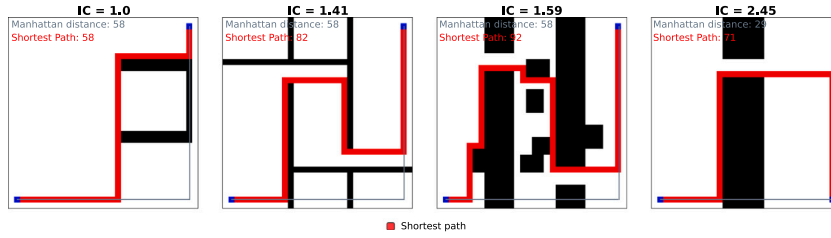


Fig. 9. Example of different instance complexities.

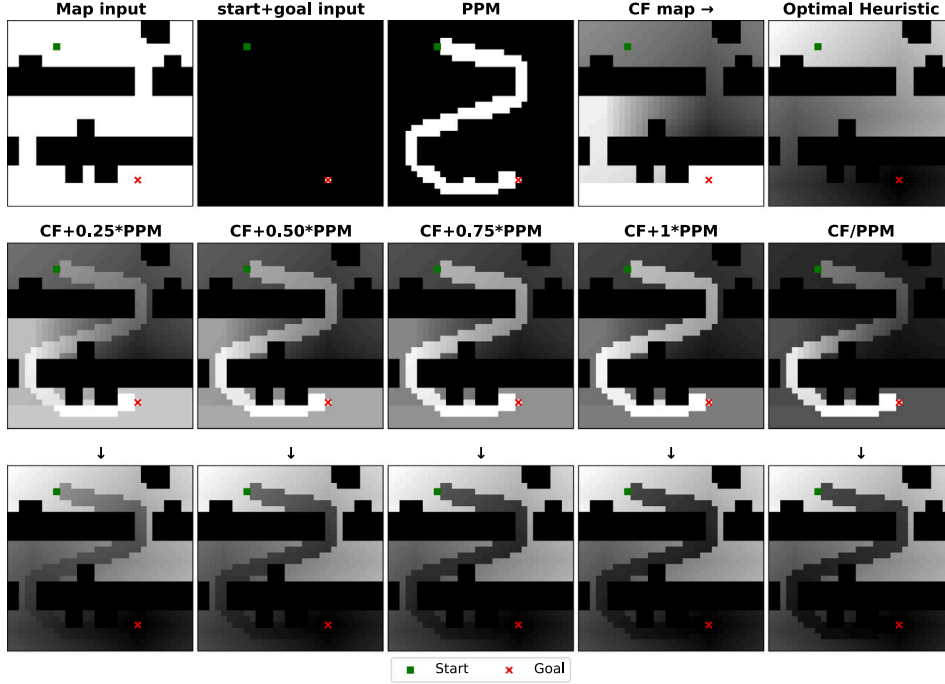


Fig. 10. Examples of the proposed correctors and the reconstructed heuristics.

- CF + α · PPM Method: The CF is combined with the PPM, weighted by a factor α . Different values of α (0.25, 0.5, 0.75, and 1) are explored to determine the optimal balance. The corrected heuristic function is computed as

$$f(n) = g(n) + w \cdot \frac{h(n)}{c f(n) + \alpha \cdot ppm(n)}. \quad (3)$$

This approach adjusts the heuristic values based on the relative contributions of CF and PPM, improving search accuracy for varying obstacle distributions.

- CF/PPM Method: The heuristic is scaled by the ratio of PPM to CF. To handle low PPM values, any PPM values below a threshold of 0.75 (based on experimental results) are set to 3, a value exceeding the typical maximum of 1 in PPM. The heuristic function is given by

$$f(n) = g(n) + w \cdot h(n) \cdot \frac{ppm(n)}{c f(n)}. \quad (4)$$

This scaling ensures robust heuristic corrections, focusing on nodes likely to belong to optimal paths.

These combination methods aim to leverage the strengths of both CF and PPM, ensuring robustness in scenarios where one of the components may be inaccurate. Fig. 10 shows some example results. The first row shows the input map with start and goal points, the initial heuristics PPM and CF, and the optimal heuristic derived from the CF map. The second row depicts the proposed CF+ α ·PPM for different α values and CF/PPM combinations. Lastly the third row shows the heuristic reconstructed using these corrected heuristics.

3.6. Training procedure

Two training implementations are used for the TransAE CF and TransAE PPM models, corresponding to the MP and Tiled-MP datasets. As a result, a total of four models are trained: MP TransAE CF, MP TransAE PPM, Tiled-MP TransAE CF, and Tiled-MP TransAE PPM.

- **MP TransAE Training:** For models trained on the MP dataset, the Adam optimizer was employed. The training ran for 50 epochs with a batch size of 64. The learning rate was adjusted dynamically using the OneCycleLR scheduler, with a peak learning rate set to 4×10^{-4} . These hyperparameters were selected based on previous works and were not further tuned.
- **Tiled-MP TransAE Training:** For the models trained on the Tiled-MP dataset, the same Adam optimizer was used, but with a batch size of 1 due to the varied image sizes. The training ran for 20 epochs, and the OneCycleLR scheduler was applied with the same peak learning rate as for the MP models.

In both implementations, the models are trained to minimize the Masked Mean Squared Error (Masked MSE) loss function, which computes the error over the free cells in the map, excluding obstacles from the calculation. This approach ensures a more accurate evaluation of the models' predictions for both CF and PPM, focusing on the areas that are traversable.

Remark 2. The proposed approach could be mapped from 2D to 3D environments by adding a new dimension that varies the gray-scale of the map (e.g., obstacle = 0, flat/free space = 1, and the z-axis between the interval (0, 1)) which allows identifying free-zones for path planning. Also, it is important to add some gradient constraints into the A* algorithm (e.g., it is not possible to jump from 0.1 to 0.9 or vice-versa). Future work will study the generalization of the proposed approach to 3D environments.

3.7. Evaluation metrics

To compare the performance of the proposed planners, we employ several standard evaluation metrics, that assess path optimality, efficiency, and computational performance. These metrics, widely used in the literature [11,13,22], are described as follows.

3.7.1. Optimality ratio (OR)

OR measures the percentage of instances for which a planner finds the shortest possible path. A higher OR indicates better performance, with a maximum value of 100% signifying that the planner consistently identifies optimal paths. The OR is calculated by $OR = 100 \times \frac{N_{opt}}{N}$.

3.7.2. Path length ratio (PLR)

PLR evaluates how close a planner's path cost is to the optimal cost. It is expressed as a percentage, where 100% represents an optimal path. Higher percentages reflect sub-optimal performance, as they indicate longer paths relative to the optimal solution. Lower PLR values are preferred. The PLR is calculated by $PLR = 100 \times \frac{\text{cost}(\pi)}{\text{cost}(\pi^*)}$.

3.7.3. Expansions ratio (ER)

ER compares the number of node expansions a planner performs to the expansions made by A* with the Manhattan heuristic. An ER of 100% indicates equal performance to the baseline, while values below 100% suggest greater efficiency due to fewer expansions. The ER is calculated by $ER = 100 \times \frac{E}{E_{Man}}$.

3.7.4. Timing metrics

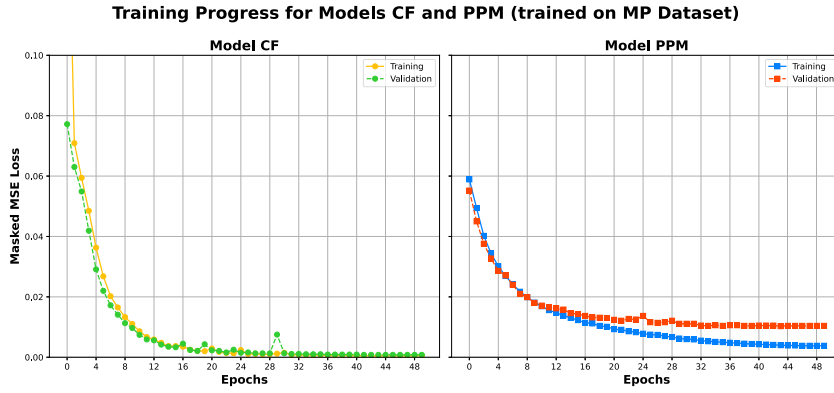
Two additional metrics are used to assess the computational performance. (i) Search Time: The time required for the planner to solve a path planning problem, which typically correlates with the number of node expansions. (ii) Inference Time: The fixed time taken by the neural network to generate heuristic maps, independent of input size. This metric highlights the computational overhead of incorporating learned heuristics.

We evaluate key attributes of the proposed planners using the aforementioned metrics on three different test sets: MP (overall efficiency), Tiled-MP (scalability), and out-of-distribution (generalization). In addition, we perform statistical analysis by calculating the mean and standard deviation of the metrics for different levels of instance complexity.

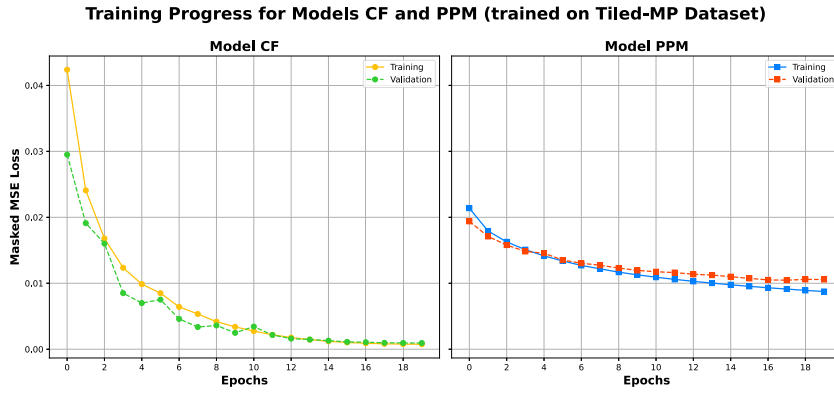
4. Experiments and results

Training for each model in the MP dataset took approximately 2.5 hours on an NVIDIA GeForce GTX 1050 with Max-Q Design GPU, while the Tiled-MP dataset models required 12 hours. The additional time for Tiled-MP was due to the varying input sizes and challenges in batching. Fig. 11 shows the training results.

The MP-trained models converged effectively, producing accurate predictions but occasionally struggled with narrow passages in PPM predictions. In contrast, the Tiled-MP models showed smoother and more accurate results across both datasets, handling complex environments better due to exposure to varied input sizes during training. However, minor gaps were observed in large maps for Tiled-MP models, though they consistently outperformed MP models.



(a) MP dataset training



(b) Tiled-MP dataset training

Fig. 11. Training results.

4.1. Planners and baseline

To evaluate the proposed heuristic combinations, we developed planners that integrate CF and PPM, aiming to harness the predictive accuracy of PPM while addressing the limitations of the Focal Search algorithm. These planners, named as the search algorithm + the heuristic (or corrector), include:

- Four $WA^* + CF + \alpha \cdot PPM$ with $\alpha \in 0.25, 0.50, 0.75, 1$
- $WA^* + CF/PPM$

The performance of these planners is compared against several baselines:

- $A^* + \text{Manhattan}$: The standard A^* algorithm using the Manhattan distance heuristic, which guarantees optimal paths but does not account for obstacles.
- $WA^* + CF$: A weighted planner leveraging only the Correction Factor.
- $FS + PPM + CF$: The Focal Search algorithm with PPM as the secondary heuristic and CF as the primary heuristic.
- $FS + PPM + \text{MAN}$: Focal Search with PPM as the secondary heuristic and Manhattan distance as the primary heuristic.

The sub-optimality parameters for the search algorithms were set to $w = 2$ for WA^* and $\epsilon = 1.25$ for FS, ensuring a balance between path length and computational cost. These baselines, including those from prior work [22], provide a comparison to assess the effectiveness of the heuristic combinations in improving pathfinding accuracy and efficiency.

4.2. Efficiency results

The evaluation procedure was applied to assess the overall performance of planners on the MP test set, focusing on heuristic predictors trained on either MP or Tiled-MP (T-MP) datasets (DS).

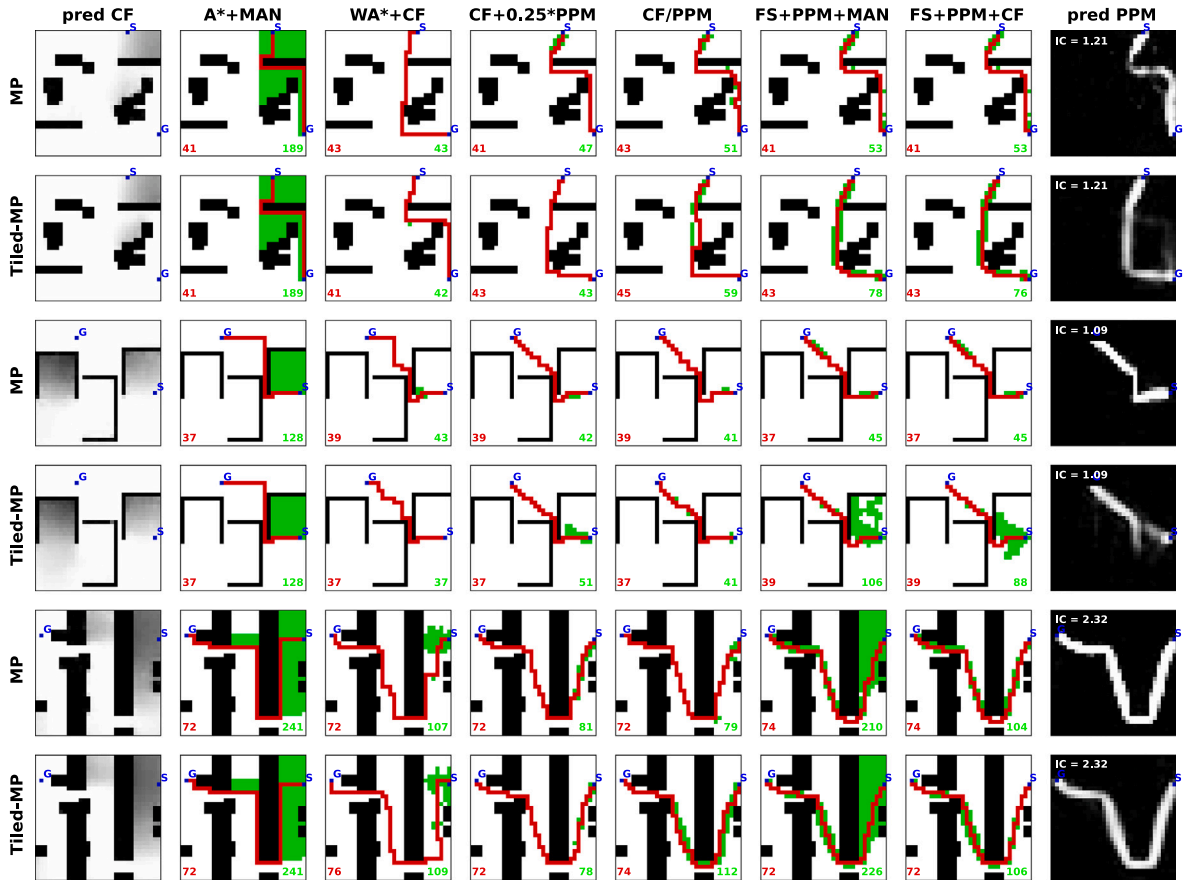


Fig. 12. Examples of results from the MP and Tiled-MP planners on the MP test set.

4.2.1. Qualitative overview

Fig. 12 shows the efficiency results in the MP dataset. All proposed planners consistently outperformed the baseline $A^* + \text{Manhattan}$ in terms of node expansions, though they exhibited some path sub-optimality. The $CF + \alpha \cdot PPM$ and CF/PPM planners frequently matched or surpassed $WA^* + CF$, particularly when the PPM predictions were reliable while the CF was less accurate. $FS + PPM$ planners, however, showed a tendency for over-expansion, particularly in cases where PPM predictions were less accurate.

4.2.2. Quantitative results

Table 2 highlights efficiency results on the MP test set. $WA^* + CF + 0.25PPM$ achieved the highest OR and lowest ER, particularly when trained on the Tiled-MP dataset (DS), demonstrating a balance between optimality and efficiency. Other values of α are not shown as they yielded similar results.

4.2.3. Analysis by instance complexity

Statistical analysis (see Fig. 13) revealed that the $CF + \alpha \cdot PPM$ planners consistently outperformed baselines across different levels of instance complexity. For higher-complexity problems, the proposed methods significantly reduced node expansions compared to instance-independent heuristics, demonstrating their adaptability to complex obstacle configurations.

In summary, $WA^* + CF + 0.25PPM$, especially when trained on Tiled-MP, emerged as the most effective planner, achieving near-optimal paths while reducing computational overhead.

4.3. Scalability results

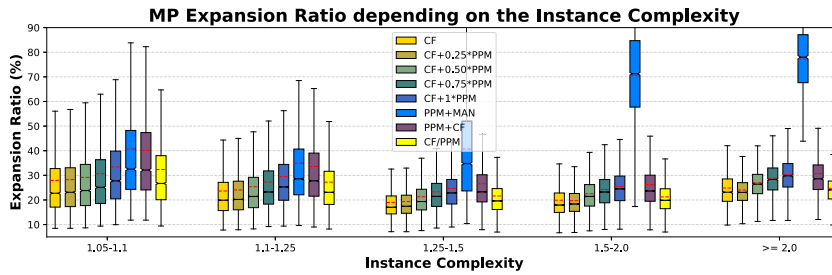
The evaluation procedure was extended to assess the scalability performance of planners on the Tiled-MP test set, focusing on heuristic predictors trained on either MP or Tiled-MP (T-MP) datasets (DS). The results are shown in Fig. 14.

4.3.1. Qualitative overview

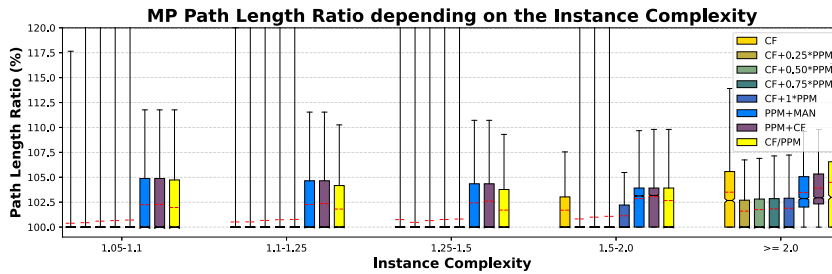
All proposed planners demonstrated notable improvements over the baseline $A^* + \text{Manhattan}$, especially in scenarios with larger, more complex instances. The $WA^* + CF + \alpha \cdot PPM$ planners consistently reduced node expansions while achieving near-optimal paths,

Table 2
Experimental results on the MP test set.

Planners	DS	OR (%)	PLR (%) ↓	ER (%) ↓
A* + MAN		100	100	100
WA* + CF	MP	72.07	102.02 ± 4.11	23.96 ± 12.92
	T-MP	82.60	101.02 ± 2.67	22.32 ± 11.34
WA* + CF	MP	85.08	100.86 ± 2.73	22.93 ± 11.71
+ 0.25*PPM	T-MP	87.92	100.63 ± 2.13	22.15 ± 11.27
WA* + CF	MP	85.42	100.88 ± 2.85	24.81 ± 12.23
+ 0.50*PPM	T-MP	85.07	100.80 ± 2.41	24.19 ± 11.70
WA* + CF	MP	85.28	100.91 ± 2.95	26.68 ± 12.86
+ 0.75*PPM	T-MP	83.27	100.88 ± 2.48	26.02 ± 12.26
WA* + CF	MP	85.20	100.92 ± 2.98	28.79 ± 14.03
+ 1*PPM	T-MP	82.44	100.93 ± 2.53	27.82 ± 13.07
WA*	MP	72.46	101.90 ± 4.26	24.34 ± 13.15
+ CF/PPM	T-MP	66.03	102.16 ± 3.87	24.85 ± 12.62
FS + PPM	MP	44.75	103.17 ± 4.36	48.14 ± 27.23
+ MAN	T-MP	54.30	102.52 ± 3.90	47.96 ± 26.48
FS + PPM	MP	42.17	103.49 ± 5.00	31.40 ± 18.97
+ CF	T-MP	52.83	102.68 ± 4.24	30.67 ± 17.44



(a) ER



(b) PLR

Fig. 13. Box-and-whiskers plots depending on instance complexity of MP dataset.

with the best results observed for $\alpha = 1$. Planners trained on the Tiled-MP dataset generally outperformed their MP-trained counterparts, particularly in handling larger-scale instances. However, FS + PPM planners exhibited a tendency to over-expand, particularly in cases where PPM predictions lacked accuracy or consistency.

4.3.2. Quantitative results

Table 3 summarizes the scalability results on the Tiled-MP test set. The WA* + CF + 1PPM achieved the best balance of OR, PLR, and ER, highlighting its effectiveness in scaling to more challenging scenarios. Notably, planners trained on Tiled-MP outperformed those trained on MP across all metrics, further validating the benefits of tailoring predictors to instance characteristics.

4.3.3. Analysis by instance complexity and map size

Fig. 15 shows the statistical analysis of instance complexity using the Tiled-MP dataset. The results revealed that WA* + CF + α *PPM planners significantly outperformed the baselines in reducing expansions across all complexity levels. For higher-complexity and size

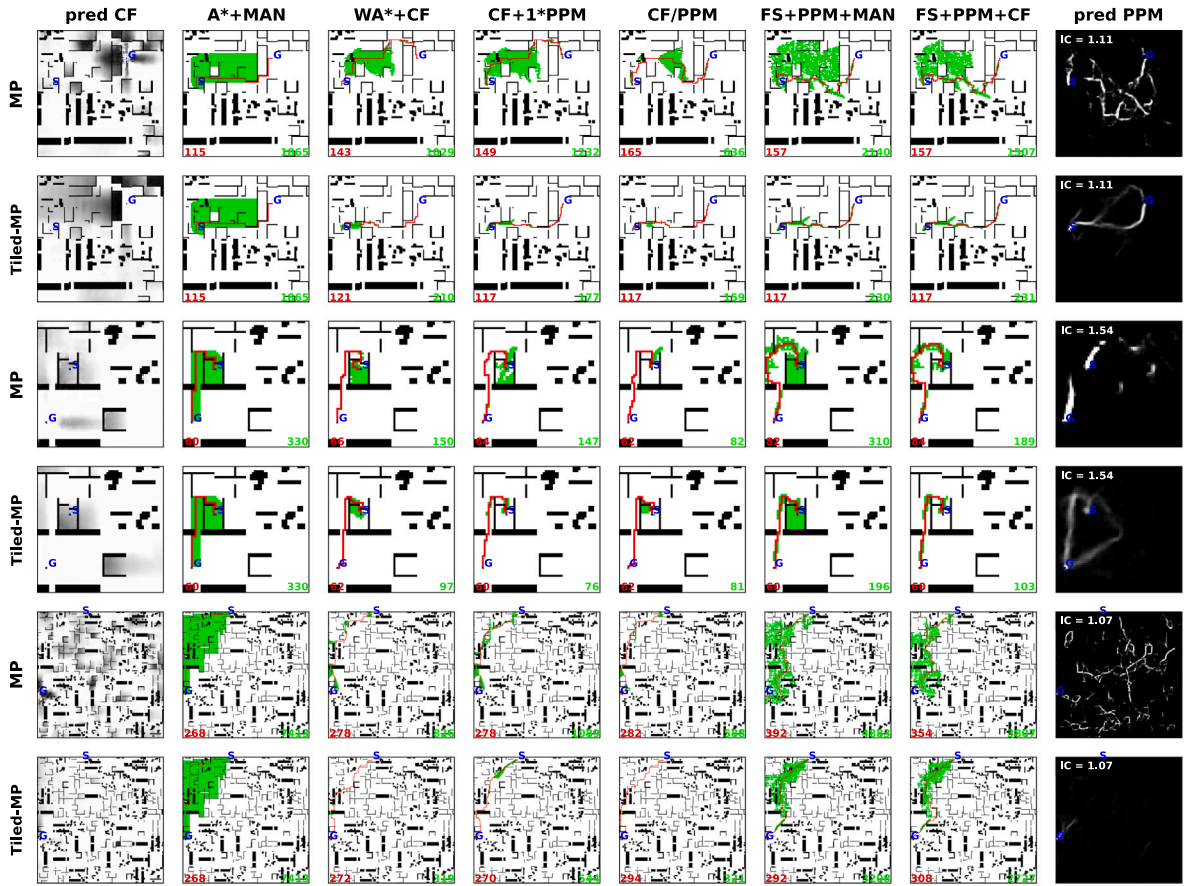


Fig. 14. Examples of results from the MP and Tiled-MP planners on the Tiled-MP test set.

Table 3

Experimental results on the Tiled-MP test set.

Planners	DS	OR (%)	PLR (%) ↓	ER (%) ↓
A* + MAN		100	100	100
WA* + CF	MP	27.27	109.23 ± 10.01	58.39 ± 63.52
	T-MP	18.92	114.78 ± 14.68	64.72 ± 105.71
WA* + CF	MP	27.92	109.55 ± 10.61	50.07 ± 41.12
+ 0.25*PPM	T-MP	42.94	106.14 ± 8.95	37.85 ± 36.66
WA* + CF	MP	26.95	110.01 ± 11.24	48.67 ± 39.10
+ 0.50*PPM	T-MP	43.84	105.16 ± 7.58	32.67 ± 29.81
WA* + CF	MP	26.30	111.01 ± 12.35	49.74 ± 43.69
+ 0.75*PPM	T-MP	45.05	105.01 ± 7.51	32.12 ± 26.54
WA* + CF	MP	27.47	111.64 ± 13.28	50.65 ± 42.98
+ 1*PPM	T-MP	45.04	104.73 ± 7.18	32.09 ± 24.17
WA*	MP	21.10	116.18 ± 16.89	38.74 ± 37.26
+ CF/PPM	T-MP	19.22	116.03 ± 17.48	46.83 ± 99.74
FS + PPM	MP	12.99	132.38 ± 27.76	104.78 ± 103.37
+ MAN	T-MP	39.64	106.92 ± 10.90	49.54 ± 42.69
FS + PPM	MP	12.66	132.58 ± 27.76	106.51 ± 108.66
+ CF	T-MP	34.53	109.93 ± 14.94	60.13 ± 94.54

problems, training on the Tiled-MP dataset enabled planners to maintain a balance between path optimality and expansion efficiency, further demonstrating the scalability of the approach. However, FS-based planners struggled with large instances, exhibiting substantial over-expansion.

In summary, the WA* + CF + 1PPM planner, trained on Tiled-MP, demonstrated the most robust performance across all metrics, achieving near-optimal paths while effectively scaling to larger and more complex scenarios.

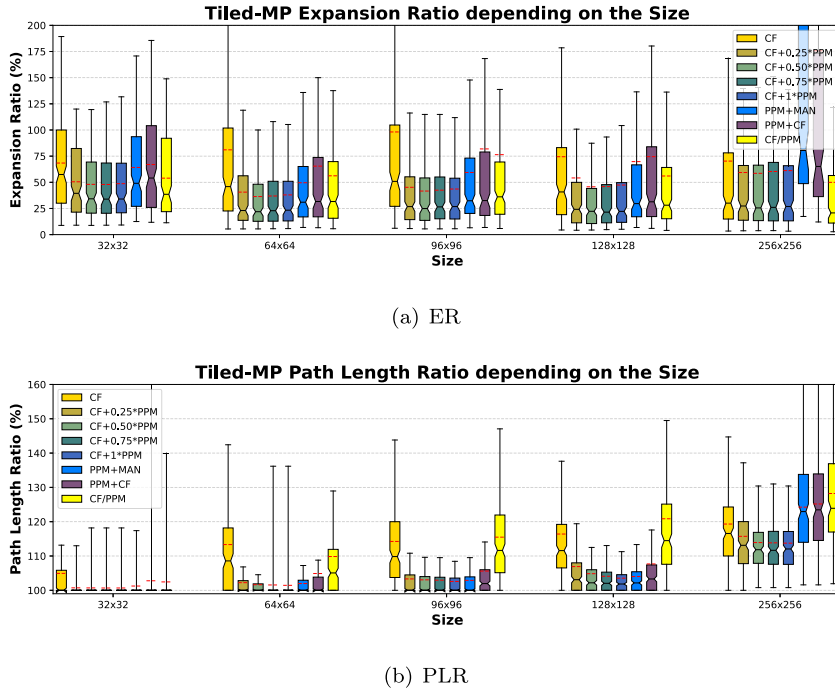


Fig. 15. Box-and-whiskers plots depending on instance complexity of the Tiled-MP dataset.

Table 4
Runtime breakdown on the MP test set.

Planners	Search Time (ms)	Total Run Time (ms)
A* + MAN	22.85	22.85
WA* + CF	7.77	22.67
WA* + CF + 0.25*PPM	6.64	35.24
WA* + CF/PPM	6.92	35.52
FS + PPM + MAN	27.31	41.01
FS + PPM + CF	15.45	44.05

4.4. Runtime analysis

For the MP dataset, the average inference time was 14.9 ms for CF and 13.7 ms for PPM, measured on an NVIDIA GeForce GTX 1050 Max-Q Design GPU. Table 4 shows the average search times and total times (search + inference) for each planner on the MP test set, using predictors trained on the Tiled-MP dataset.

On the Tiled-MP dataset, the average inference time increased to 18.1 ms for CF and 15.2 ms for PPM. The models trained on this dataset can predict outputs of varying sizes in constant time, which is particularly useful for handling larger inputs, especially when search times increase exponentially with instance size.

4.4.1. Search time complexity

A detailed comparison of the WA* + CF planner and the baseline (A* + MAN) is shown in Fig. 16. Each dot represents a problem instance, plotted according to the resulting path length and total runtime. For larger problem instances, the WA* + CF planner significantly reduces search time due to fewer search expansions. This reduction is critical since search expansions are the primary factor driving search time. As a result, the WA* + CF planner shows a much more favorable exponential runtime relationship compared to the baseline.

4.4.2. Inference VS search time

We further evaluate the impact of inference on total search time by defining the importance of an action (whether it be inference or search) as the percentage of time that action contributes to the overall search time. Fig. 17 reveals that as the size of the instance increases, the contribution of inference to the total runtime decreases. Specifically, for paths longer than 30 nodes, the search time becomes as significant as the inference time in the overall runtime. Additionally, the importance of inference drops below 10% for paths longer than 170 nodes and converges toward about 2.5% for very long paths. This trend confirms that the time spent on constant heuristic predictions is negligible when compared to the exponentially increasing search time.

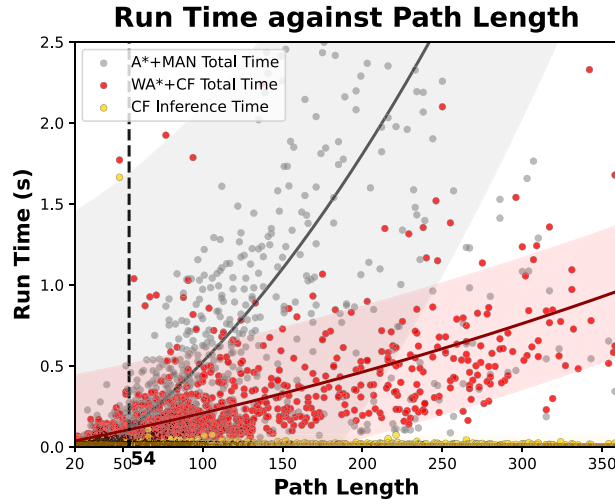


Fig. 16. Evolution of search runtime as a function of path length.

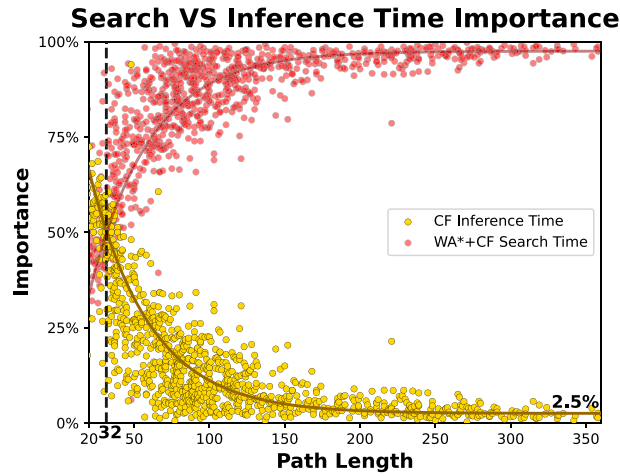


Fig. 17. Evolution of inference time contribution as a function of instance path length.

4.5. Generalization results

We further evaluated the performance of the proposed planners on an out-of-distribution (OOD) dataset in order to assess the generalization ability of the planners trained on the MP and Tiled-MP datasets in unfamiliar settings. The results are shown in Fig. 18.

4.5.1. Qualitative overview

The $WA^* + CF + \alpha^*PPM$ planners consistently reduced node expansions while achieving near-optimal paths, with the best results observed for $\alpha = 0.5$. Planners trained on the Tiled-MP dataset generally outperformed those trained on the MP dataset, particularly in handling more complex, out-of-distribution instances. However, models relying on the TransAE-generated PPM experienced some challenges, especially when trained on the simpler MP dataset. These models struggled to accurately predict the CF map and PPM, leading to less efficient searches. Despite this, the overall performance of the proposed planners remained robust, even in cases where the PPM predictions were unreliable.

4.5.2. Quantitative results

Table 5 presents the generalization results on the Out-of-the-Distribution test set. The generalization capabilities are highlighted through the optimal solving of 25% of the unseen problem instances. For sub-optimal solutions, the resulting paths are only 7% longer compared to those generated by instance-independent heuristics. Additionally, the number of expanded nodes is reduced to less than 70% of that required by conventional heuristics. Moreover, the limitations of the FS algorithm are once again shown, as well as the improved performance of training on the more complex Tiled-MP dataset.

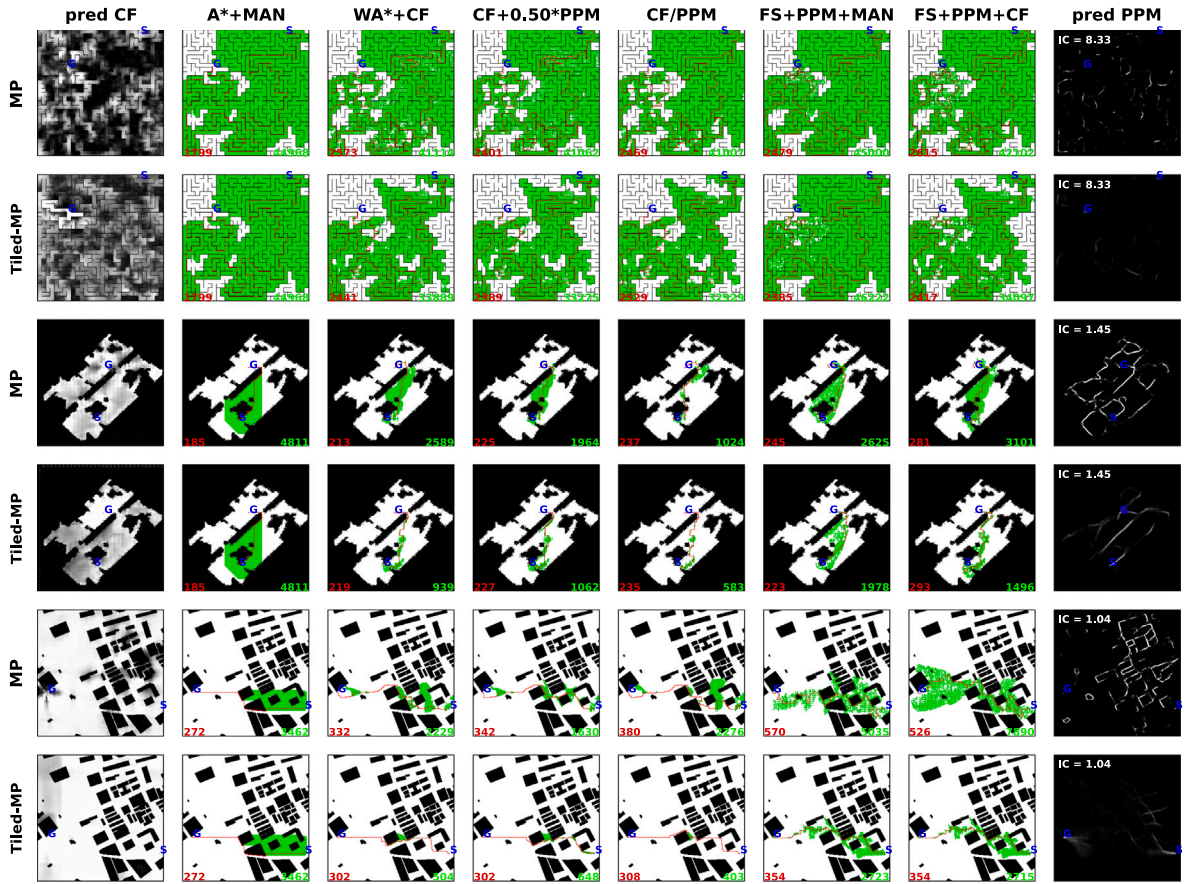


Fig. 18. Examples of results from the MP and Tiled-MP planners on the OOD test set.

Table 5

Experimental results on the out-of-distribution test set.

Planners	DS	OR (%)	PLR (%) ↓	ER (%) ↓
A* + MAN		100	100	100
WA* + CF	MP	13.64	117.65 ± 21.43	93.62 ± 116.56
	T-MP	22.09	108.94 ± 17.82	84.25 ± 146.55
WA* + CF	MP	17.53	116.37 ± 20.46	88.09 ± 115.68
+ 0.25*PPM	T-MP	25.58	107.25 ± 17.63	76.90 ± 137.17
WA* + CF	MP	18.83	116.66 ± 19.84	85.40 ± 115.19
+ 0.50*PPM	T-MP	25.00	107.11 ± 17.57	75.42 ± 131.99
WA* + CF	MP	16.88	117.00 ± 20.44	84.44 ± 107.95
+ 0.75*PPM	T-MP	26.16	107.38 ± 17.34	75.69 ± 132.75
WA* + CF	MP	16.23	117.50 ± 21.06	85.18 ± 110.00
+ 1*PPM	T-MP	25.58	107.18 ± 17.29	74.41 ± 129.73
WA*	MP	15.58	124.28 ± 25.91	82.05 ± 167.30
+ CF/PPM	T-MP	18.02	115.45 ± 23.48	65.62 ± 99.57
FS + PPM	MP	11.04	132.14 ± 26.48	155.31 ± 276.45
+ MAN	T-MP	22.67	118.34 ± 20.07	223.70 ± 811.62
FS + PPM	MP	10.39	138.17 ± 34.47	260.17 ± 931.16
+ CF	T-MP	18.02	120.41 ± 19.55	209.56 ± 760.82

4.5.3. Analysis by instance complexity and map type

The statistical analysis of Fig. 19 shows that both PLR and ER increased as the map complexity grew due to the difficulty in predicting accurate heuristics in complex areas. Additionally, mazes present challenges for both pathfinding and node expansion, as the instance-independent baseline planner is likely to be more efficient for this type of map.

In summary, the WA* + CF + 0.50*PPM planner performed better in terms of OR and PLR, while the WA* + CF/PPM shone in reducing the number of expansions on unseen data, demonstrating decent generalization capabilities.

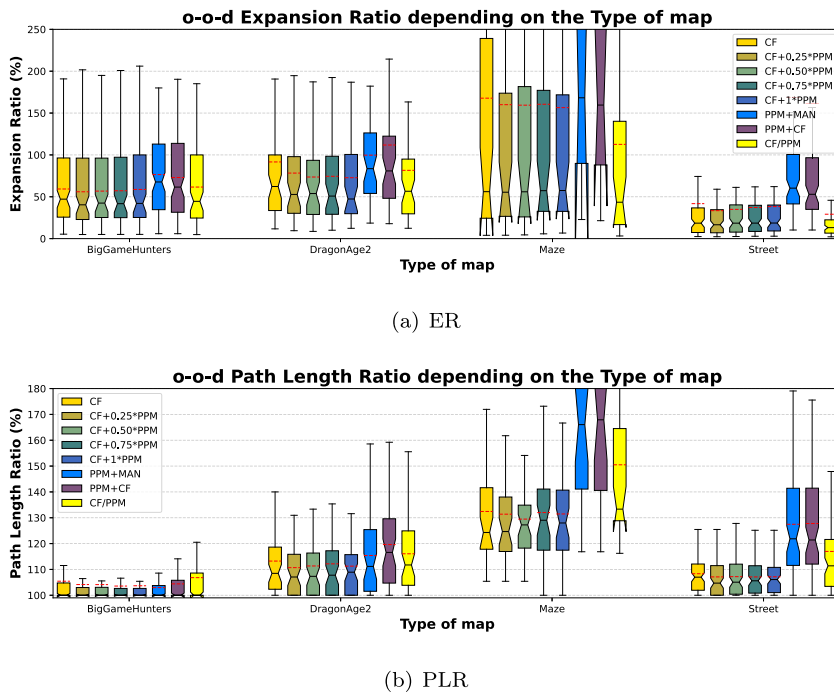


Fig. 19. Box-and-whiskers plots depending on instance complexity of OOD dataset.

5. Conclusions and future work

This research aimed to address the challenges of scalable and efficient path planning in autonomous robotics by integrating instance-specific heuristic learning with traditional search-based algorithms. The proposed framework successfully combines autoencoder transformer (TransAE) networks for predicting two key heuristic functions—Correction Factor (CF) and Path Probability Map (PPM)—enhancing the performance of the Weighted A* algorithm in both known and unseen environments. Our experiments demonstrated the efficacy of this approach across multiple datasets, including the Motion Planning (MP) and Tiled-MP datasets, where our method consistently outperformed traditional pathfinding techniques. The integration of CF and PPM significantly reduced the number of node expansions and maintained near-optimal path quality. Notably, our approach was able to handle larger instances with ease, demonstrating strong scalability and generalization across diverse path planning problems.

The proposed framework does face limitations, particularly in scenarios with highly complex environments or unseen map types. The reliability of the predicted heuristics can decrease in these cases, leading to minor inconsistencies in path planning. Moreover, while our results demonstrate the potential of heuristic learning, the variability in performance across different environments and map types highlights the need for further refinement and testing. Future work will refine the transformer architecture for better scalability and generalization in real-time applications as well as extend the framework to 2.5D or 3D for broader robotic applications.

CRedit authorship contribution statement

Elie Thellier: Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Adolfo Perrusquía:** Writing – original draft, Supervision, Project administration, Funding acquisition, Conceptualization. **Antonios Tsourdos:** Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

- [1] Y. Na, Y. Li, D. Chen, Y. Yao, T. Li, H. Liu, K. Wang, Optimal energy consumption path planning for unmanned aerial vehicles based on improved particle swarm optimization, *Sustainability* 15 (16) (2023), <https://doi.org/10.3390/su151612101>

- [2] A. Eliades, E. Thellier, H. Tian, S. Mehta, N. Almesafri, H. Chen, Z. Wei, A. Perrusquía, C. Liu, W. Guo, Drone pollution tracking in cities using recurrent proximal policy optimization learning, in: 2024 IEEE International Smart Cities Conference (ISC2), IEEE, 2024, pp. 1–6.
- [3] F. Leon, G.M. Atanasiu, Probabilistic Path Finding Method for Post-Disaster Risk Estimation, Springer International Publishing, Cham, 2014, pp. 61–76.
- [4] K. Tamanakijprasart, A. Perrusquía, S. Mondal, A. Tsourdos, Autonomous path selection of unmanned aerial vehicle in dynamic environment using reinforcement learning, in: AIAA SCITECH 2025 Forum, 2025, pp. 1998.
- [5] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
- [6] A.M. Ali, A. Perrusquía, W. Guo, A. Tsourdos, Flight plan optimisation of unmanned aerial vehicles with minimised radar observability using action shaping proximal policy optimisation, Drones 8 (10) (2024) 546.
- [7] C. Liu, L. Wu, G. Li, H. Zhang, W. Xiao, D. Xu, J. Guo, W. Li, Improved multi-search strategy a* algorithm to solve three-dimensional pipe routing design, Expert Syst. Appl. 240 (2024) 122313.
- [8] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, M.Q.-H. Meng, A survey of learning-based robot motion planning, IET Cyber-Syst. Robot. 3 (4) (2021) 302–314.
- [9] E. Thellier, A. Perrusquía, A. Tsourdos, Instance-specific heuristic learning for scalable UAV path planning, in: AIAA AVIATION FORUM AND ASCEND 2025, 2025, pp. 3397.
- [10] T. Kulvicius, S. Herzog, T. Lüddecke, M. Tamosiunaite, F. Wörgötter, One-shot multi-path planning for robotic applications using fully convolutional networks, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 1460–1466, <https://doi.org/10.1109/ICRA40945.2020.9196719>
- [11] T. Bian, Y. Xing, A. Zolotas, End-to-end one-shot path-planning algorithm for an autonomous vehicle based on a convolutional neural network considering traversability cost, Sensors 22 (24) (2022), <https://doi.org/10.3390/s22249682>
- [12] J. Ferreira, A.A.F. Júnior, Y.M. Galvão, P. Barros, S.M.M. Fernandes, B.J.T. Fernandes, Performance improvement of path planning algorithms with deep learning encoder model, in: 2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), IEEE, 2020, pp. 1–6.
- [13] T. Takahashi, H. Sun, D. Tian, Y. Wang, Learning heuristic functions for mobile robot path planning using deep neural networks, in: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 29, 2019, pp. 764–772.
- [14] Y. Ariki, T. Narihira, Fully convolutional search heuristic learning for rapid path planners, ArXiv abs/1908.03343 (2019).
- [15] C.M. Weber, E.P. Freitas, L. Silva al, Safe and topographic path planning with deep neural networks, in: 2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), IEEE, 2022, pp. 1–6.
- [16] C. Neisse, J.L. Soares, L. Silva al, E.P. Freitas, Investigating deep neural networks as heuristic functions for path planning with topographic terrain characteristics in agent-based simulation, in: 2022 IEEE 31st International Symposium on Industrial Electronics (ISIE), IEEE, 2022, pp. 174–181.
- [17] H.L. Krever, T.R.S. Leão, J.M. Pasa, E.P. de Freitas, R.C. Nunes, L.A.L. Silva, Learning heuristics for topographic path planning in agent-based simulations, in: SIMULTECH, 2023, pp. 115–125.
- [18] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, A. Kanezaki, Path planning using neural a* search, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, vol. 139 of PMLR, 2021, pp. 12029–12039 Proceedings of Machine Learning Research.
- [19] X. Chen, F. Yang, C. Wang, ia *: Imperative learning-based a* search for pathfinding, arXiv preprint arXiv:2403.15870, 2024.
- [20] I. Pohl, Heuristic search viewed as path finding in a graph, Artif. Intell. 1 (3–4) (1970) 193–204.
- [21] J. Pearl, J.H. Kim, Studies in semi-admissible heuristics, IEEE Trans. Pattern Anal. Mach. Intell. (4) (1982) 392–399.
- [22] D. Kirilenko, A. Andreychuk, A. Panov, K. Yakovlev, Transpath: learning heuristics for grid-based pathfinding via transformers, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 12436–12443.
- [23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, arXiv preprint Tech report arXiv:1512.03385, 2015, <https://doi.org/10.48550/arXiv.1512.03385>
- [24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv preprint arXiv:1706.03762, 2017, <https://doi.org/10.48550/arXiv.1706.03762>
- [25] M. Bhardwaj, S. Choudhury, S. Scherer, Learning heuristic search via imitation, arXiv:1707.03034, 2017.
- [26] A. Nash, K. Daniel, S. Koenig, A. Felner, *Theta**: any-angle path planning on grids, in: AAAI, vol. 7, 2007, pp. 1177–1183.



DSPACE

<https://dspace.org/>

Scalable and generalizable path planning for robotic navigation using transformer-based heuristic learning

Thellier, Elie; Perrusquía, Adolfo; Tsourdos, Antonios

2026-05-25

Elsevier

<https://doi.org/10.1016/j.ins.2026.123149>

Thellier E, Perrusquía A, Tsourdos A. (2026) Scalable and generalizable path planning for robotic navigation using transformer-based heuristic learning. Information Sciences, Volume 739, May 2026, Article number 123149