2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

# Neural Informed RRT*: Learning-based Path Planning with Point Cloud State Representations under Admissible Ellipsoidal Constraints

Zhe Huang, Hongyu Chen, John Pohovey, and Katherine Driggs-Campbell

***Abstract*— Sampling-based planning algorithms like Rapidly-exploring Random Tree (RRT) are versatile in solving path planning problems. RRT* offers asymptotic optimality but requires growing the tree uniformly over the free space, which leaves room for efficiency improvement. To accelerate convergence, rule-based informed approaches sample states in an admissible ellipsoidal subset of the space determined by the current path cost. Learning-based alternatives model the topology of the free space and infer the states close to the optimal path to guide planning. We propose Neural Informed RRT* to combine the strengths from both sides. We define point cloud representations of free states. We perform *Neural Focus*, which constrains the point cloud within the admissible ellipsoidal subset from Informed RRT*, and feeds into PointNet++ for refined guidance state inference. In addition, we introduce *Neural Connect* to build connectivity of the guidance state set and further boost performance in challenging planning problems. Our method surpasses previous works in path planning benchmarks while preserving probabilistic completeness and asymptotic optimality. We deploy our method on a mobile robot and demonstrate real world navigation around static obstacles and dynamic humans. Code is available at `https://github.com/tedhuang96/nirrt_star`.**

## I. Introduction

Path planning is the task of finding a path for a robot to traverse from a start to a goal safely and efficiently [1]–[3]. An effective path planning algorithm should be (1) complete and optimal: a solution is guaranteed to be found if one exists, and the optimal solution is guaranteed to be achieved with sufficient run time; (2) efficient in optimal convergence: the solution should be quickly improved towards near optimal; and (3) versatile and scalable: the implementation should be modified with minimal effort to generalize across different problems, environments, and robots.

Multiple branches of planning algorithms have been developed to meet these requirements, including grid-based search, artificial potential field, and sampling-based algorithms [7]–[11]. Sampling-based algorithms are popular due to their versatility, scalability, and formal properties of probabilistic completeness and asymptotic optimality [4]. To accelerate convergence to the optimal path, various sampling strategies are introduced to replace the default uniform sampling [4]. The rule-based informed strategy enforces sampling in an admissible ellipsoidal subset of states which are more promising to improve the current path solution [5], [12]. The learning-based methods harness grid-based neural networks to make inference of states close to the optimal path, and bias sampling towards these states, which we define as guidance states [6], [13]–[19].

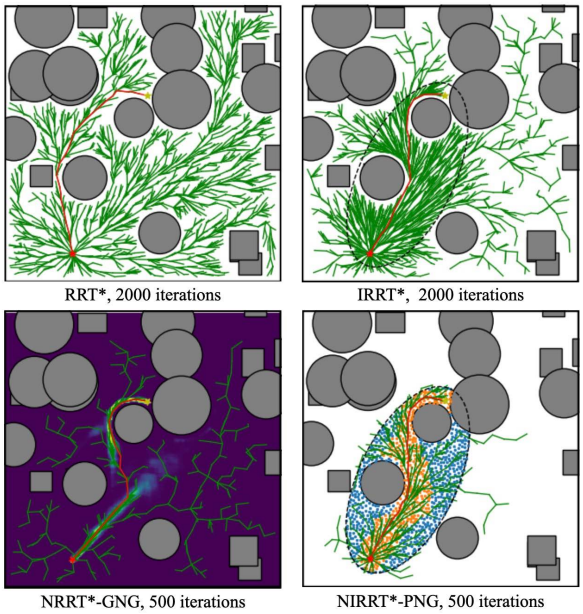


Fig. 1: Solutions of a 2D random world found by RRT* [4], Informed RRT* (IRRT*) [5], Neural RRT* with Grid-based Network Guidance (NRRT*-GNG) [6], and Neural Informed RRT* with Point-based Network Guidance (NIRRT*-PNG). NIRRT* effectively integrates IRRT* and point-based network, so IRRT* helps point-based network focus guidance state inference on the important region for solution improvement, and point-based network helps IRRT* sample critical states in the admissible ellipsoidal subset for convergence acceleration.

While these works improve performance, we observe three limitations. First, learning-based methods encode whole state space to generate guidance states without iterative improvement, where inference speed and accuracy are affected by modeling features of irrelevant region or obstacles. Second, rule-based informed sampling does not favor topologically critical states in the ellipsoidal subset (e.g., narrow corridors). Finally, learning-based methods do not consider connectivity of the guidance state set, which severely affects the convergence rate in complex planning problems.

We introduce Neural Informed RRT* (NIRRT*) to address these limitations (Figure 1). We represent free states with a point cloud, and apply PointNet++ [20] to classify guidance states. Sampling from the guidance states is mixed with the random sampling step of Informed RRT* (IRRT*) [5]. Using a point cloud instead of an occupancy grid allows us to

Z. Huang, H. Chen, J. Pohovey, and K. Driggs-Campbell are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. emails: {zheh4, hongyuc5, jpohov2, krdc}@illinois.edu

This work was supported by the National Science Foundation under Grant No. 2143435.

perform *Neural Focus*: constraining the point clouds by the admissible ellipsoidal subset of the free space, from which the critical states are inferred by PointNet++. The quality of the guidance states is continually improved during iteration, because the inference is always made on the informed subset created by an improved path cost. In addition, we build connectivity of the guidance state set by following a *Neural Connect* scheme similar to RRT-Connect [21], where the point-based network is called to solve a subproblem with a closer pair of start and goal states.

In short, our contributions are threefold: (1) we use a Point-based Network (PointNet++) to directly take free states as point cloud input to generate multiple guidance states in one run; (2) we present Neural Informed RRT*, by introducing Neural Focus to integrate Point-based Network and Informed RRT*; and (3) we propose Neural Connect to address the connectivity issue of inferred guidance state set.

## II. Related Work

Grid-based search methods like A* [7] and D* [8] guarantee to find the optimal path in a discretized state space if a solution exists, at the cost of poor scaling with the problem complexity. Sampling-based algorithms like probabilistic roadmap (PRM) [10] and RRT [11] guarantee to find a feasible path solution if one exists as the number of iterations approaches infinity. PRM* and RRT* [4] provide asymptotic optimality, which requires exploring the planning domain globally. Informed RRT* and Batch Informed Trees improve the convergence rate by constraining the sampling space to a ellipsoidal subset based on start state, goal state, and current best path cost [5], [12].

Another line of works accelerates path planning by investigating the search space of the problem, such as Vonoroi bias [22], [23], evolutionary algorithms [24], and A* initialization [25]. Neural RRT* represents the square-shaped search space of 2D planning problems by images and uses U-Net [26] to predict a probabilistic heatmap of states used for guiding RRT* [6]. MPNet voxelizes environment point cloud and feed into 3D Convolution Neural Networks to make recursive inference for path generation [15]. Grid-based neural networks are prevalent in previous works to encode the search space [6], [13]–[15], [19], which requires discretization operations and the results are dependent on resolution. Previous works use PointNet to encode the point cloud of obstacle states [27], [28], but modeling the obstacle interior is inefficient for finding a path in the free space. Recent works apply graph neural networks to a sampled random geometric graph in configuration space, and select edges from the graph to build a near-optimal path [29], [30]. However, the path feasibility and path quality are highly dependent on the sampled graph, while continuous improvement is not discussed in these works.

## III. Method

### A. Problem Definition

We define the optimal path planning problem similar to related works [4]–[6]. The state space is denoted as $X \subseteq \mathbb{R}^d$. The obstacle space and the free space are denoted as $X_{\text{obs}}$ and $X_{\text{free}}$. A path $\sigma : [0,1] \to X_{\text{free}}$ is a sequence of states. The set of paths is denoted as $\Sigma$. The optimal path planning problem is to find a path $\sigma^*$ which minimizes a given cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$, connects a given start state $x_{\text{start}} \in X_{\text{free}}$ and a given goal state $x_{goal} \in X_{\text{free}}$, and has all states on the path in free space. Formally:

$$\begin{aligned} \sigma^* &= \arg\min_{\sigma \in \Sigma} c(\sigma) \\ \text{s.t. } \sigma(0) = x_{\text{start}}, &\sigma(1) = x_{\text{goal}}, \forall s \in [0,1], \sigma(s) \in X_{\text{free}} \end{aligned} \tag{1}$$

### B. Neural Informed RRT*

We present NIRRT* in Algorithm 1, where the unhighlighted part is from RRT*, the blue part is from IRRT*, and the red part is our contribution. We track the best path solution cost $c^i_{\text{best}}$ through each iteration, which is initialized as infinity (line 3). We initialize update cost $c_{\text{update}}$ with the value of $c^0_{\text{best}}$ (line 4). We call the neural network to infer an initial guidance state set $X_{\text{guide}}$ based on the complete free state space (line 5). As better solutions are found, the guidance state set $X_{\text{guide}}$ may be updated by the neural network calls depending on how much the path cost has been improved, and random samples $x_{\text{rand}}$ are sampled using both $X_{\text{guide}}$ and informed sampling (line 8).

`PointNetGuidedSampling`: When the current best path cost $c_{\text{curr}}$ is less than the path cost improvement ratio $\alpha \leq 1$ of $c_{\text{update}}$, the neural network is called to update $X_{\text{guide}}$, and $c_{\text{update}}$ is updated by $c_{\text{curr}}$. The random sample $x_{\text{rand}}$ is sampled with a mixed strategy: if a random number Rand() $\in (0,1)$ is smaller than 0.5, we use the sampling strategy of IRRT* to sample $x_{\text{rand}}$; otherwise, we sample $x_{\text{rand}}$ uniformly from $X_{\text{guide}}$. Similar to [6], [15], [18], our mixed sampling strategy guarantees probabilistic completeness and asymptotic optimality by implementing the sampling procedure of IRRT* with a non-zero probability.

Note the frequency of calling neural networks for guidance state inference is controlled by the path cost improvement ratio $\alpha$. If we do not update $X_{\text{guide}}$ after initial inference, and remove IRRT* components, NIRRT* is reduced to NRRT*. While NIRRT* is generic in that any neural network that infers guidance states can fit into the framework, we emphasize the use of a point-based network. In the next subsection, we discuss the details of Point-based Network Guidance (PNG), and explain the preference of point representations over grid representations.

### C. Point-based Network Guidance

**Point-based Network.** We represent the state space by a point cloud $X_{\text{input}} = \{x_1, x_2, \ldots, x_N\} \subset X_{\text{free}}$. The density of point cloud should allow a reasonable amount of neighbors around each point in radius of step size $\eta$. We oversample points uniformly from $X_{\text{free}}$, and perform minimum distance downsampling to obtain the point cloud with even distribution. We create a one-hot vector for each point, indicating whether the point is within radius $\eta$ of $x_{\text{start}}$ or $x_{\text{goal}}$. We concatenate the one-hot vectors with normalized point coordinates to generate point cloud representations of

**Algorithm 1** Neural Informed RRT*

$V \leftarrow \{x_{\text{start}}\}; E \leftarrow \varnothing;$
$X_{\text{soln}} \leftarrow \varnothing;$
$c^0_{\text{best}} \leftarrow \infty;$
$c_{\text{update}} \leftarrow c^0_{\text{best}};$
$X_{\text{guide}} \leftarrow \texttt{PointNetGuide}(x_{\text{start}}, x_{\text{goal}}, c^0_{\text{best}}, X_{\text{free}});$
**for** $i = 1$ to $n$ **do**
  $c^i_{\text{best}} \leftarrow \min_{x_{\text{soln}} \in X_{\text{soln}}} \{\texttt{Cost}(x_{\text{soln}})\};$
  $x_{\text{rand}}, X_{\text{guide}}, c_{\text{update}} \leftarrow \texttt{PointNetGuidedSampling}(X_{\text{guide}}, x_{\text{start}}, x_{\text{goal}}, c_{\text{update}}, c^i_{\text{best}}, X_{\text{free}});$
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}});$
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}});$
  **if** $\texttt{CollisionFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, r_{\text{RRT}^*});$
    $V \leftarrow V \cup \{x_{\text{new}}\};$
    $x_{\text{min}} \leftarrow x_{\text{nearest}};$
    $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}));$
    **for all** $x_{\text{near}} \in X_{\text{near}}$ **do**
      **if** $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
        $x_{\text{min}} \leftarrow x_{\text{near}};$
        $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}));$
      **end if**
    **end for**
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\};$
    **for all** $x_{\text{near}} \in X_{\text{near}}$ **do**
      **if** $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$ **then**
        $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}});$
        $E \leftarrow (E \backslash \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\};$
      **end if**
    **end for**
    **if** $\texttt{InGoalRegion}(x_{\text{new}})$ **then**
      $X_{\text{soln}} \leftarrow X_{\text{soln}} \cup \{x_{\text{new}}\};$
    **end if**
  **end if**
**end for**
**return** G=(V, E);

**Algorithm 2** $\texttt{PointNetGuidedSampling}(X_{\text{guide}}, x_{\text{start}}, x_{\text{goal}}, c_{\text{update}}, c_{\text{curr}}, X_{\text{free}})$

**if** $c_{\text{curr}} < \alpha\, c_{\text{update}}$ **then**
  $X_{\text{guide}} \leftarrow \texttt{PointNetGuide}(x_{\text{start}}, x_{\text{goal}}, c_{\text{curr}}, X_{\text{free}});$
  $c_{\text{update}} \leftarrow c_{\text{curr}};$
**end if**
**if** Rand() $< 0.5$ **then**
  **if** $c_{\text{curr}} < \infty$ **then**
    $x_{\text{rand}} \leftarrow \texttt{InformedSampling}(x_{\text{start}}, x_{\text{goal}}, c_{\text{curr}});$
  **else**
    $x_{\text{rand}} \leftarrow \texttt{UniformSampling}(X_{\text{free}});$
  **end if**
**else**
  $x_{\text{rand}} \leftarrow \texttt{UniformSampling}(X_{\text{guide}});$
**end if**
**return** $x_{\text{rand}}, X_{\text{guide}}, c_{\text{update}};$

the free states. The processed point cloud $\bar{X}_{\text{input}}$ is fed into a point-based network $f$. The network $f$ maps each point to a probability $p_i \in [0, 1]$, where the points with probability greater than 0.5 form the set of guidance states $X_{\text{guide}}$. Formally,

$$\{p_1, p_2, \ldots, p_N\} = f(\bar{X}_{\text{input}}),\ X_{\text{guide}} = \{x_i | p_i > 0.5\}. \quad (2)$$

We implement PointNet++ [20] as the model architecture of the point-based network. Since PointNet++ is originally designed for 3D point cloud, we set $z$ coordinates as zero for 2D problems. We collect 4,000 2D random worlds as the training dataset. For each random world, we run A* in pixel space with step size of unit pixel and clearance of 3 pixels to generate the pixel-wise optimal path. We generate a point cloud of number $N = 2048$, and generate guidance state labels by checking whether each point is around any point of the pixel-wise optimal path in radius of $\eta$, which is set as 10 pixels. We train PointNet++ by Adam optimizer [31] with an initial learning rate of 0.001 and batch size of 16 for 100 epochs. We use the trained model across all types of 2D planning problems. For 3D random world problems, we follow a similar scheme, but the clearance is set as 2 voxels.

**Neural Focus.** Informed RRT* outperforms RRT* by proposing a heuristic ellipsoidal subset of the planning domain $X_{\text{focus}}$ in terms of the current best solution cost $c_{\text{curr}}$, in order to sample $x_{\text{rand}}$ which is more likely to improve the current solution. The reasoning behind this sampling strategy is that for any state $x_{\text{rejected}}$ from $X \backslash X_{\text{focus}}$, the minimum cost of a feasible path from $x_{\text{start}}$ to $x_{\text{goal}}$ through $x_{\text{rejected}}$ is greater than $c_{\text{curr}}$:

$$X_{\text{focus}} = \{x \in X \,|\, ||x - x_{\text{start}}||_2 + ||x - x_{\text{goal}}||_2 \leq c_{\text{curr}}\} \quad (3)$$

Neural Focus is to constrain the point cloud input to the point-based network inside the $X_{\text{focus}}$, which is equivalent as changing the domain of oversampling from $X_{\text{free}}$ to $X_{\text{focus}} \cap X_{\text{free}}$. Since we normalize point coordinates when processing point cloud inputs, the trained point-based network can handle point clouds sampled from domains at different scales. With the same number of points $N$, a smaller volume of $X_{\text{focus}}$ leads to a denser point cloud, which describes important regions with finer details. For example, Figure 3(b) shows that Neural Focus fills the narrow passage with a large number of points, which is captured by the point-based network to produce more effective inference on guidance states compared to Figure 3(a).

**Neural Connect.** The points close to $x_{\text{start}}$ or $x_{\text{goal}}$ are usually classified as guidance states with greater probabilities than the points around midway of the path (e.g., Figure 3(d)). When the distance between $x_{\text{start}}$ and $x_{\text{goal}}$ gets longer, the guidance state set $X_{\text{guide}}$ is more likely to be separated into disconnected "blobs". This phenomenon of probability polarization is reported in NRRT* work [6]. Our experiments show lack of connectivity limits the performance in large and complex planning problems.

We address this issue by introducing Neural Connect, which is inspired by RRT-Connect [21]. We initialize $X_{\text{guide}}$

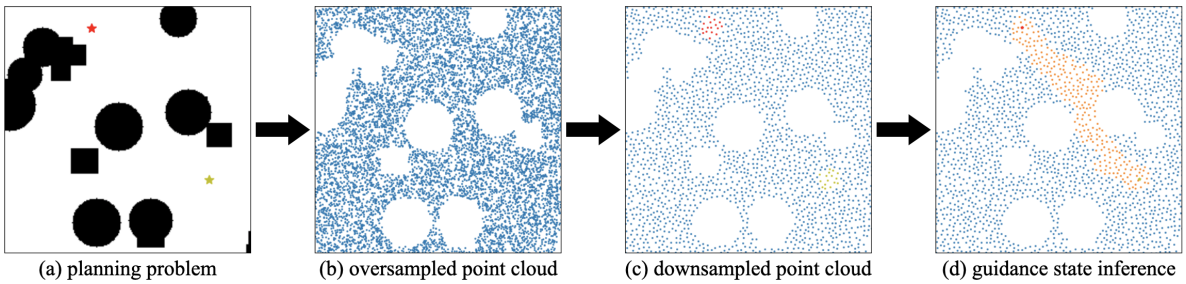


Fig. 2: Guidance state inference by point-based network. Red is start, yellow is goal, blue is free states, and orange is guidance states.

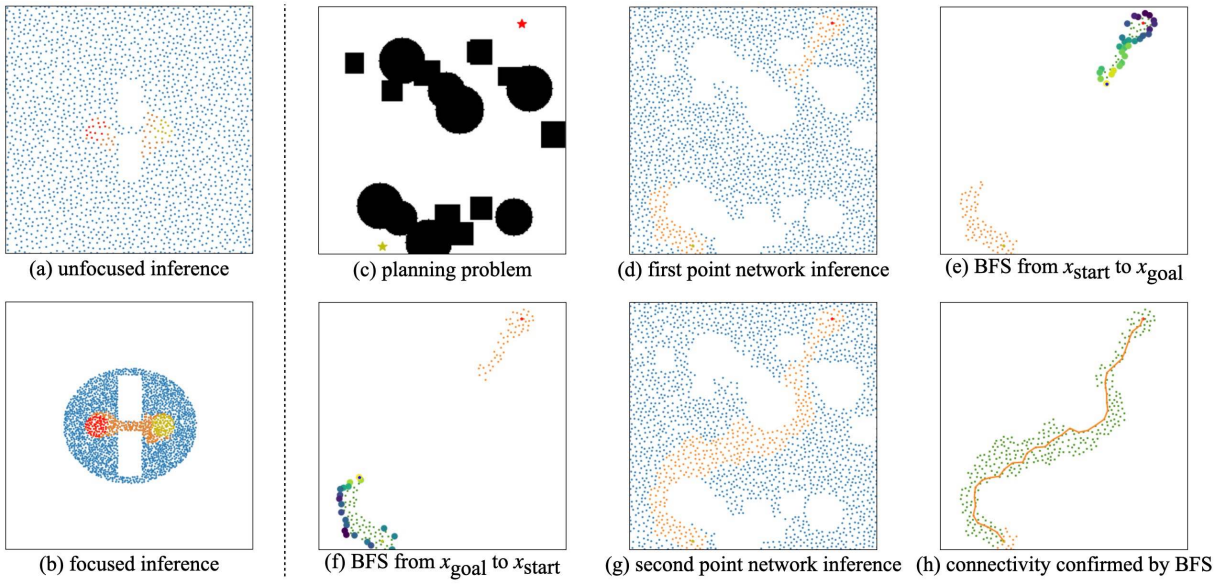


Fig. 3: (a-b) Neural Focus in a 2D narrow passage and (c-h) Neural Connect in a 2D random world. Green dots denote states visited by Breadth First Search (BFS). Circles of larger size around green dots denote boundary points. The colors of circles denote the heuristic scores, where brighter colors represent higher scores. The boundary point which is $x_{\text{start}}^{i+1}$ or $x_{\text{goal}}^{i+1}$ has a blue marker on the circle. The orange line denotes the path found by BFS which represents the connectivity of $X_{\text{guide}}$.

as an empty set, $x_{\text{start}}^1$ as $x_{\text{start}}$, and $x_{\text{goal}}^1$ as $x_{\text{goal}}$. During iteration, we first call the point-based network with $x_{\text{start}}^i$ and $x_{\text{goal}}^i$ as start and goal, and add inferred guidance states to $X_{\text{guide}}$. Second, We run Breadth First Search (BFS) from $x_{\text{start}}$ to $x_{\text{goal}}$ through the guidance states in $X_{\text{guide}}$. The neighbor radius of BFS is set as $\eta$, and no collision check is performed. After BFS is finished, connectivity of $X_{\text{guide}}$ is confirmed if $x_{\text{goal}}$ is reached. Otherwise, we find the boundary points $X_{\text{bound}}$ of the states visited by BFS by checking whether any points in $X_{\text{input}} \backslash X_{\text{guide}}$ are around the visited state of radius $\eta/2$. We select $x_{\text{start}}^{i+1}$ from $X_{\text{bound}}$ which is one of the states heuristically the furthest from $x_{\text{start}}$ and one of the states to reach $x_{\text{goal}}$ with minimum total heuristic cost. Third, we perform the same operation as the second step, with the start of BFS as $x_{\text{goal}}$, and the goal of BFS as $x_{\text{start}}$. We obtain $x_{\text{goal}}^{i+1}$ if connectivity is negative. We perform the iteration until connectivity is built or the limit of iteration $n_{\text{guide}}$ is reached, which we set as 5 in practice. We illustrate Neural Connect in Figure 3(c-h). Note the orange path found by BFS in Figure 3(h) does not go through collision check, so the path is not a feasible solution but a visual demonstration on the connectivity of $X_{\text{guide}}$.

`PointNetGuide`: We apply both Neural Focus and Neural Connect to the point-based network, and obtain the complete module of Point-based Network Guidance, which is presented in Algorithm 3.

**Point versus Grid.** We prefer using points over grids to represent state space due to compatibility with geometric constraints and convenience of extension to different problems. To apply Neural Focus to a CNN, grid representations require masking of the complement set of the ellipsoidal subset, where the mask quality depends on grid resolution. CNN also has to process the irrelevant masked region within the rectangular/box grid input. In contrast, point representations naturally confine states within arbitrary geometry by modifying the sampling domain, and the point-based network only needs to model free states. Moreover, while the point-based network just needs adjustment of the input format to extend to different dimensions, changing input dimensions usually requires redesign of CNN architecture.

## IV. Experiments

### A. Simulation Experiments

**Planning Problems.** We conduct simulation experiments on 2D center block, 2D narrow passage, 2D random world, and 3D random world problems. The center block and the narrow passage problems are defined similar to IRRT*

**Algorithm 3** $\texttt{PointNetGuide}(x_{\text{start}}, x_{\text{goal}}, c_{\text{curr}}, X_{\text{free}})$

$x^1_{\text{start}} \leftarrow x_{\text{start}}$;
$x^1_{\text{goal}} \leftarrow x_{\text{goal}}$;
$X_{\text{guide}} \leftarrow \varnothing$;
**if** $c_{\text{curr}} < \infty$ **then**
    $X_{\text{focus}} \leftarrow \texttt{InformedSubset}(x_{\text{start}}, x_{\text{goal}}, c_{\text{curr}})$;
    $X_{\text{input}} \leftarrow \texttt{PointCloudSampling}(X_{\text{focus}} \cap X_{\text{free}})$;
**else**
    $X_{\text{input}} \leftarrow \texttt{PointCloudSampling}(X_{\text{free}})$;
**end if**
**for** $j = 1$ to $n_{\text{guide}}$ **do**
    $\bar{X}_{\text{input}} \leftarrow \texttt{AddOneHotFeatures}(X_{\text{input}}, x^j_{\text{start}}, x^j_{\text{goal}})$;
    $\bar{X}_{\text{input}} \leftarrow \texttt{NormalizeCoordinates}(\bar{X}_{\text{input}})$;
    $X_{\text{guide}} \leftarrow X_{\text{guide}} \cup \text{PointBasedNetwork}(\bar{X}_{\text{input}})$;
    connectivity, $x^{j+1}_{\text{start}} \leftarrow \texttt{BFS}(X_{\text{guide}}, x_{\text{start}}, x_{\text{goal}}, \eta)$;
    **if** connectivity **then**
        **return** $X_{\text{guide}}$;
    **end if**
    connectivity, $x^{j+1}_{\text{goal}} \leftarrow \texttt{BFS}(X_{\text{guide}}, x_{\text{goal}}, x_{\text{start}}, \eta)$;
    **if** connectivity **then**
        **return** $X_{\text{guide}}$;
    **end if**
**end for**
**return** $X_{\text{guide}}$;

work [5] (Figure 4). The center block problem examines the efficiency of planners to sample states relevant to the problem in a wide free space. The narrow passage problem studies the capability of planners to focus sampling in topologically critical area. The random world problems evaluate versatility and scalability of planners.

In the center block problems, we specify 5 different map sizes with respect to a fixed start-goal distance, and set the block width randomly for 100 independent runs. In the narrow passage problems, we specify 5 different gap heights, and set random positions of the passage for 100 independent runs. We generate 500 random worlds for each 2D and 3D cases for evaluation. Note we use clearance of 3 pixels for 2D random world, zero clearance for 2D center block and 2D narrow passage, and 2 voxels for 3D random world. The default size of 2D planning problems is $224 \times 224$, and the default size of 3D planning problems is $50 \times 50 \times 50$.

**Metrics.** For the center block problems, we measure the number of iterations to reach within a path cost threshold, which is some percentage above the optimal cost. For the narrow passage problems, we measure the number of iterations to find a path through the passage. For the random world problems, we examine the iterations each planner spends on finding the initial solution, and path cost improvement after certain numbers of iterations.

**Baselines.** We compare NIRRT* to RRT* [4], IRRT* [5], NRRT*-GNG [6], and variants of our method across the experiments. NIRRT*-PNG(FC) is our complete algorithm, where F is Neural Focus and C is Neural Connect. NIRRT*-PNG(F) removes Neural Connect from the complete version. NRRT*-PNG is Neural RRT* with the point-based network. NRRT*-PNG(C) uses Neural Connect in addition to NRRT*-PNG. We train a U-Net [26] with pretrained ResNet50 weights [32] for NRRT*-GNG for 2D problems.

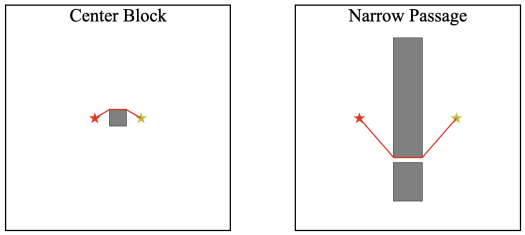

Fig. 4: Center block and narrow passage experiments.

**Experiment Results.** The center block experiments show in Figure 5(b) that NIRRT*-PNG(FC) outperforms IRRT* in terms of the speed to find near-optimal paths across different problem sizes. The point-based network is able to infer guidance states from the informed subset which are the most promising to converge the path solution to optimum. Both Figure 5(a)(c) show that NIRRT*-PNG(FC) and NIRRT*-PNG(F) have similar performance. The informed subset effectively constrains the region of the point cloud to be around the center block, and significantly simplifies the task of guidance state inference. Therefore, the point-based network performs well even without Neural Connect. Similar to the claim by [6] that initial path solution cost of NRRT* is better than RRT*, we see in Figure 5(c) that NRRT*-PNG and NRRT*-PNG(C) are faster than RRT* in terms of reaching within a more relaxed threshold above optimal cost such as 7-10%. However, the convergence speeds of NRRT* variants tend to be slow and are often worse than RRT* when approaching a tighter bound such as 2-4%. In contrast, NIRRT* variants work consistently better than IRRT* across thresholds of optimal cost, since the informed subset allows the point-based network to provide finer distribution of the guidance states to continuously refine the path towards the optimal solution.

In the narrow passage setting, NIRRT*-PNG(FC) finds a difficult path through the passage faster and more frequently than IRRT*, as represented in Figure 5(f). The convergence speed of NIRRT*-PNG(FC) outperforms all baselines as shown in Figure 5(e). NIRRT*-PNG(F) performance is similar to IRRT* because the guidance state set usually ends up separated on left and right sides of the gap without Neural Connect, whereas NIRRT*-PNG(FC) is able to connect the guidance state set together through the gap, which helps sampling critical states inside the gap. Both NRRT*-GNG and NRRT*-PNG are worse than RRT*, but NRRT*-PNG(C) works consistently better than RRT*, which indicates the effectiveness of Neural Connect in planning problems with critical states. Note we collect the training dataset for point-based network with optimal paths which requires clearance of 3 pixels, which is equivalent to 7 pixels of the gap height. Figure 5(e) demonstrates that our point-based network generalizes well to planning problems with clearances tighter than training distribution by Neural Focus and Neural Connect, while the CNN model is sensitive to clearance [6].

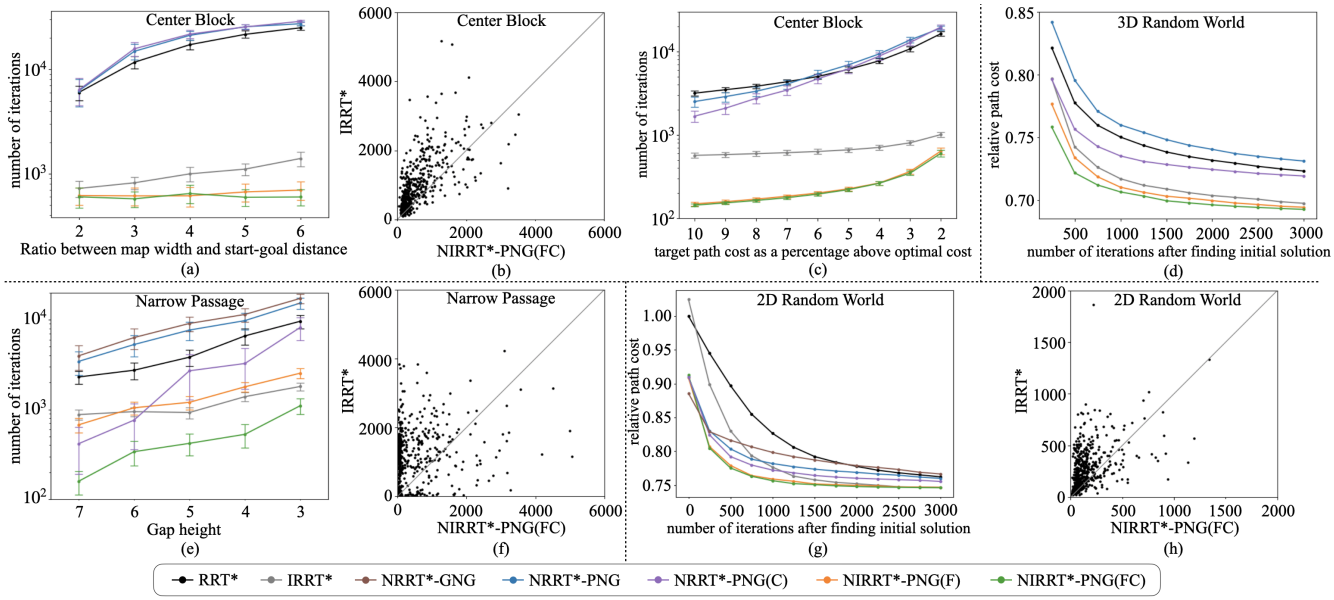


Fig. 5: Experiment results. **Center block**: (a) The average number of iterations to find a path within 2% of the optimal cost for different map widths; (b) Comparison of the number of iterations IRRT* and NIRRT*-PNG(FC) take to find a path within 2% of the optimal cost for each center block problem; (c) The average number of iterations to find a path within the specified tolerance of the optimal cost. **3D Random world**: (d) The average path cost relative to the initial solution of RRT* at different numbers of iterations after finding an initial solution. **Narrow passage**: (e) The average number of iterations to find a path better than flanking the obstacle for different gap heights; (f) Comparison of the number of iterations IRRT* and NIRRT*-PNG(FC) take to find a path better than flanking the obstacle for each narrow passage problem. **2D Random world**: (g) The average path cost relative to the initial solution of RRT* at different numbers of iterations after finding an initial solution; (h) Comparison of the number of iterations IRRT* and NIRRT*-PNG(FC) to find an initial solution for each random world problem. Error bars denote 95% confidence interval. The error bars are not plotted for random worlds for clarity of figures. NRRT*-GNG is not implemented for center block and random world 3D due to incompatible grid sizes and incompatible number of dimensionality.

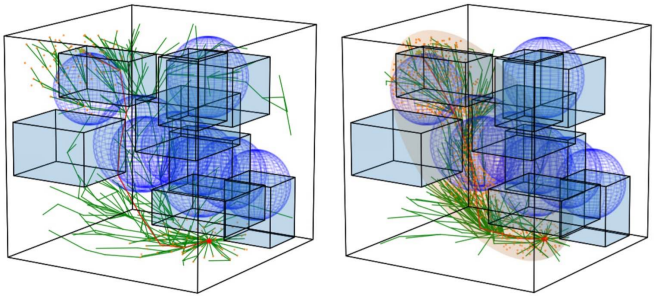

Fig. 6: Visualization on planning in 3D random world at 500 iterations. Left: NRRT*-PNG. Right: NIRRT*-PNG(FC).

For each random world problem, we record the cost of path solution at certain number of iterations after the initial solution is found, and plot these costs relative to the cost of the initial path solution from RRT*. The 2D and 3D results are presented in Figure 5(g) and (d) respectively, and planning in 3D random worlds are visualized in Figure 6. We observe NRRT*-GNG has the best initial solution in 2D since the grid representations are denser than point representations in terms of the whole state space. However, NIRRT* variants converge faster due to continuous improvement of the guidance states. We find that both Neural Connect and Neural Focus contribute to improvement of convergence speed in both 2D and 3D cases. Figure 5(h) shows that NIRRT*-PNG(FC) is faster than IRRT* in terms of finding initial path solution.

### *B. Real World Deployment*

We deploy our method and the model trained in 2D random world to a TurtleBot 2i. The demonstration of real world navigation with static obstacles and dynamic humans is available at `https://sites.google.com/view/nirrt-star`.

## V. Conclusions

We present Neural Informed RRT* approach to accelerate optimal path planning by incorporating a point-based network into Informed RRT* for guidance state inference. We introduce Neural Focus to naturally bridge the point-based network and the informed sampling strategy with point cloud representations of free states. We propose Neural Connect to improve quality of the inferred guidance state set by enforcing connectivity. Our simulation experiments show that Neural Informed RRT* outperforms RRT*, Informed RRT*, and Neural RRT* in terms of convergence rate towards optimal solutions in planning problems with varying sizes, critical states, and randomized complicated patterns.

In future work, we want to study how to further improve our algorithm when the planning problem sizes are significantly different from the training distribution. We would like to explore the effectiveness of our work in higher-dimensional problems. It is also interesting to study if we can denoise the guidance state set inferred by the point-based network to offer an end-to-end option for generating feasible and near-optimal paths [33], [34].

## REFERENCES


[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[3] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[6] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[8] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 3310–3317.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 3067–3074.

[13] N. Pérez-Higueras, F. Caballero, and L. Merino, "Learning human-aware path planning with fully convolutional networks," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 5897–5902.

[14] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 9535–9541.

[15] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[16] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 3965–3972.

[17] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 1488–1495.

[18] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 7087–7094.

[19] N. Ma, J. Wang, J. Liu, and M. Q.-H. Meng, "Conditional generative adversarial networks for optimal path planning," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 662–671, 2021.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[21] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2000, pp. 995–1001.

[22] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3856–3861.

[23] J. Wang and M. Q.-H. Meng, "Optimal path planning using generalized voronoi graph and multiple potential functions," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 12, pp. 10 621–10 630, 2020.

[24] S. R. Martin, S. E. Wright, and J. W. Sheppard, "Offline and online evolutionary bi-directional rrt algorithms for efficient re-planning in dynamic environments," in *IEEE International Conference on Automation Science and Engineering*. IEEE, 2007, pp. 1131–1136.

[25] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5539–5544.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[27] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," in *Conference on Robot Learning*. PMLR, 2021, pp. 355–364.

[28] K. Sugiura and H. Matsutani, "P3net: Pointnet-based path planning on fpga," in *International Conference on Field-Programmable Technology*. IEEE, 2022, pp. 1–9.

[29] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[30] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao, "Learning-based motion planning in dynamic environments using gnns and temporal encoding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 003–30 015, 2022.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, vol. 1412, 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[33] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[34] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9902–9915.