**Software Engineering and Applications** 软件工程与应用, 2023, 12(6), 763-773
Published Online December 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.126074

Hans汉斯

# 基于数据增强的神经路径规划器

**寇国钊[1,2]，孙　涵[1,2]，白云翔[1,2]**

[1]上海理工大学机器智能研究院，上海
[2]上海理工大学健康科学与工程学院，上海

收稿日期：2023年10月8日；录用日期：2023年12月1日；发布日期：2023年12月13日

## 摘　要

**传统的路径规划算法主要是基于搜索或采样的方法，如D*、A*、RRT、RRT*。但随真实应用场景日趋复杂，尤其是在3D复杂场景下，传统算法的计算量大幅增加，难以实现实时计算。近年来，鉴于深度神经网络在的快速推理能力以及非线性建模方面的优势，基于神经网络的路径规划算法成为目前路径规划研究的热点。本文针对神经路径规划器训练数据不易收集，数据不足的问题，提出了两种数据增强方法，分别在环境编码器和神经规划器训练阶段应用，以此达到丰富训练样本的目的，令模型学习到更多的知识。实验表明，本文所提方法较于之前的方法，在路径规划成功率以及时间两个评估指标上都取得优势，证明了本方法的优越性能，满足路径规划任务的需求。**

## 关键词

**路径规划，神经网络，环境编码器，数据增强**

# Neural Path Planner Using Data Augmentation

**Guozhao Kou[1,2], Han Sun[1,2], Yunxiang Bai[1,2]**

[1]Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai
[2]School of Health Science and Engineering, University of Shanghai for Science and Technology, Shanghai

Received: Oct. 8th, 2023; accepted: Dec. 1st, 2023; published: Dec. 13th, 2023

## Abstract

**Traditional path planning algorithms are mainly based on search or sampling methods, such as D*, A*, RRT, and RRT*. However, with the increasing complexity of real application scenarios, especially in 3D complex scenarios, the computation of traditional algorithms increases dramatically, making it difficult to realize real-time computation. In recent years, given the advantages of deep**

文章引用: 寇国钊, 孙涵, 白云翔. 基于数据增强的神经路径规划器[J]. 软件工程与应用, 2023, 12(6): 763-773.
DOI: 10.12677/sea.2023.126074

**neural networks in terms of their fast reasoning ability and nonlinear modeling, neural network-based path planning algorithms have become the current hotspot in path planning research. This paper introduces two data augmentation methods to address the challenges of limited and difficult-to-collect training data for neural path planners. These methods are implemented during the training phase of the environment encoder and the neural planner, respectively, with the aim of enhancing the training samples and improving the model's knowledge acquisition. Experimental results demonstrate that the proposed method outperforms previous approaches in terms of path planning success rate and time, thereby confirming its superior performance and suitability for path planning tasks.**

## Keywords

**Path Planning, Neural Network, Environment Encoder, Data Augmentation**

Copyright © 2023 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

## 1. 引言

路径规划是移动机器人相关应用研究的一个重要研究方向，它的任务是找到在给定环境中从起始位置到目标位置的无碰撞可行路径。路径规划在许多领域都有广泛的应用，比如无人驾驶、机器人导航、物流管理等。随着移动机器人的及其应用的快速发展，路径规划算法也在不断演进，以应对现实应用中更加复杂的问题。本文将现有一些路径规划算法分为四类，如表 1 所示。

传统的路径规划算法中，最常见的是基于图搜索的方法，如 Dijkstra 算法和 A*算法。Dijkstra 算法通过计算起点到所有其他点的最短路径来寻找目标点的最优路径，但随着计算图的规模增大时，其开销也会与之增加。A*算法则结合了启发式信息，通过计算每个节点到目标点的代价，来优化搜索过程。此外，还有基于采样的方法，如 Rapidly-exploring Random Trees (RRT) [1]算法、RRT*算法以及一些 RRT*算法的变体[2] [3]。除了基于图搜索和采样的方法，还有一类称为演化算法[4] [5]的路径规划方法，这类算法受自然界的生物进化现象启发，通过模拟生物进化的过程来优化路径规划问题。其中，遗传算法是应用最广泛的一种演化算法。它通过模拟基因的交叉和变异来生成新的路径规划解，并根据适应度函数来评估解的优劣，从而逐代演化出较优的路径规划策略。上述这些算法虽然简单高效，但在复杂的实际场景中，尤其是高维度的 3D 场景中可能存在一些局限性。

近年来，随着机器学习和深度学习技术的迅速兴起，基于智能的路径规划方法也成为了研究热点。通过使用大量的训练数据，可以让智能体从已有经验中学习到知识去执行规划路径任务。事实上，早在上个世纪 70、80 年代就有利用神经网络进行路径规划的相关研究[6] [7]，由于受到当时硬件设备运算能力的局限，神经网络的路径规划算法并没有取得特别优秀的性能。目前，硬件设备运算能力的大幅提升，这为神经网络在路径规划中的应用提供了硬件基础。最近几年，基于神经网络的路径规划的相关工作有 MPNet [8]、GrMPN [9]和[10]，它们在完成实验中的路径规划任务时都取得了不错的结果。

然而，由于基于神经网络的路径规划方法需要大量的训练样本，训练数据的收集并不容易，使用少量数据训练的模型可能无法令人满意。因此，本文提出了两种数据增强的方法，分别在训练环境编码器和神经规划器过程中应用。本文的方法旨在丰富训练样本，弥补训练数据不足的问题，让模型能够学习到更多的经验。实验评估显示，采用本文提出的方法后，其训练的模型较于之前的模型，拥有更高的规

划成功率以及较少的规划时间，更适应路径规划任务需求。

**Table 1.** Advantages and disadvantages of four classes of path planning algorithms
**表 1.** 四种类别路径规划算法的优缺点

| 类别 | 缺点 | 优点 |
|---|---|---|
| 搜索算法 | 需要事先建立搜索地图；<br>在高维空间中计算量大 | 搜索效率高；<br>能保证结果最优 |
| 采样算法 | 不能保证结果最优；<br>采样效率受概率响 | 不需搜索地图；<br>应用范围广泛 |
| 演化算法 | 实现复杂；<br>算子参数影响最终结果 | 具有可扩展性可与其他算法结合；<br>具有较好的收敛性 |
| 智能算法 | 泛化能力差；<br>不能保证路径最优 | 不需搜索地图；<br>规划速度快 |

## 2. 基于神经网络的路径规划模型结构

基于神经网络的路径规划算法通常采用端到端的方式。端到端的含义是：网络模型自原始数据输入，到结果输出，从输入端到输出端，整个过程都时在模型中完成，而不是将模型分为多个模块完成。

端到端的神经网络路径规划算法的规划流程如图 1 所示。该算法包含两部分：

1) 环境编码器。环境编码器用于直接编码任务场景。

2) 神经规划器。神经规划器以环境编码信息和任务(当前位置、目标位置)作为输入，输出的结果是下一时刻的位置。

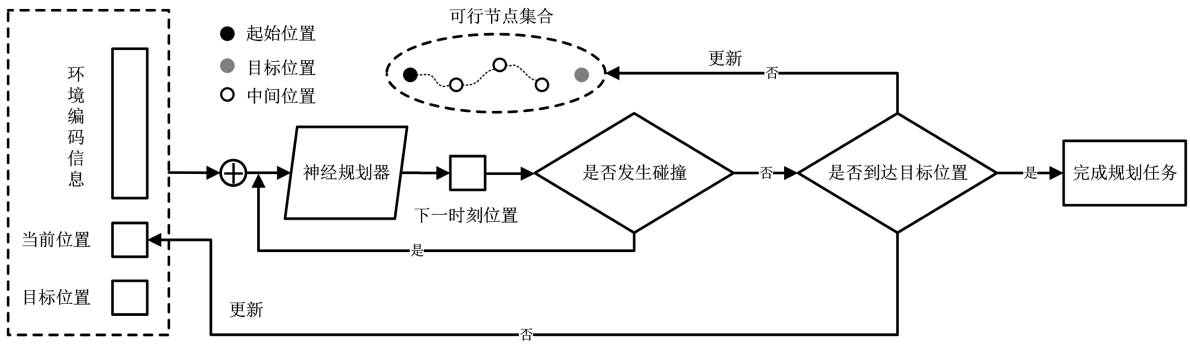



**Figure 1.** Schematic diagram of the end-to-end neural network path planning process
**图 1.** 端到端的神经网络路径规划流程示意图

算法流程如下：

步骤 1：将环境编码信息与当前位置、目标位置拼接成规划任务信息。

步骤 2：将规划任务信息输入神经规划器，输出下一时刻位置。

步骤 3：判断神经规划器预测的位置是否发生碰撞。否，执行步骤 4；是，重复步骤 2。

步骤 4：判断神经网络规划器预测的位置是否到达目标位置。否，执行步骤 5；是执行步骤 6。

步骤 5：更新可行节点集合及当前位置后重复步骤 1。

步骤 6：输出可行节点集合，完成规划任务。

### 2.1. 环境编码器

在深度学习中，编码器常常被用来降低高维数据的维度，它的本质是将高维度的数据映射到一个低

维空间，而后用低维的数据来表示原来的高维数据，这种操作一般被称作特征提取。编码器的实现思路有多种，如常见的图片分类，其使用卷积网络或者多层感知机进行特征提取。

本文中的环境编码器采用自编码器(AutoEncoder) [12]结构，这是一种无监督学习神经网络模型，用于降低数据维度、特征提取或生成类似输入数据的重构数据。自编码器由一个编码器(Encoder)和一个解码器(Decoder)组成，两者都是神经网络。自编码器的训练目标是使解码器能够将编码器产生的隐藏表示转换回原始输入数据，其结构示意图如图 2 所示。

1) 编码器：接受输入数据 $\boldsymbol{X} \in \mathbb{R}^d$，$d$ 表示数据维度，并 $\boldsymbol{X}$ 映射到一个低维的表示空间，得到原始数据 $\boldsymbol{X}$ 在低维空间的隐藏表示 $\boldsymbol{h} \in \mathbb{R}^h$，$h$ 表示低维空间的维度，且 $h < d$。

2) 隐藏表示：编码器的输出一个低维的表示 $\boldsymbol{h}$，其中包含输入数据的关键特征。这种编码操作也可以看作是一个数据压缩。

3) 解码器：输入隐藏表示 $\boldsymbol{h}$，并试着将其还原回原始输入数据的维度，输出简称为 $\boldsymbol{X}^R \in \mathbb{R}^d$。解码器的目标是学习一种映射，使得从隐藏表示到重构数据的转换能够尽可能地还原输入数据的信息。

4) 损失函数：在训练自编码器的过程中，需要一个评估标准来评估原始输入数据与解码器重构数据之间的差异，这个评估标准常称作损失函数。数据重构任务的常见损失函数是均方误差(Mean Squared Error)，其计算方式如式 1 所示。

$$MSE = \frac{1}{n}\sum_{i}^{n}\left\|Y_i - Y_i^{'}\right\|^2 \tag{1}$$

式中 $n$ 表示数据个数，$Y_i, Y_i^{'}$ 分别表示原始数据和重构数据的第 $i$ 个数据。

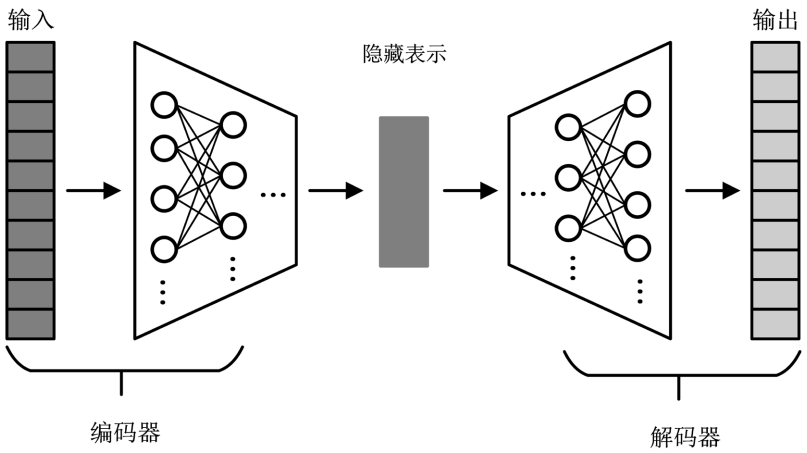



**Figure 2.** Schematic diagram of the autoencoder structure
**图 2.** 自编码器结构示意图

## 2.2. 神经规划器

神经规划器期望智能体从已有经验中学习到在任务场景中如何规划路径能力。定义函数 $G$ 代表规划路径操作，设有函数

$$G\left(\boldsymbol{f}_{task}\right) = \boldsymbol{x} \tag{2}$$

式 2 中 $\boldsymbol{f}_{task} \in \mathbb{R}^{2d+h}$ 表示规划任务，$\boldsymbol{x} \in \mathbb{R}^d$ 为下一时刻的位置。神经网络的目的就是学习到一个模型去拟合函数 $G$。

神经规划器结构如图 3 所示，其输入为规划任务 $\boldsymbol{f}_{task} \in \mathbb{R}^{2d+h}$，$\boldsymbol{f}_{task}$ 由环境编码信息 $\boldsymbol{h}$、当前位置(简

写为 $\boldsymbol{a}_{init} \in \mathbb{R}^d$ )和目标位置(简写为 $\boldsymbol{a}_{goal} \in \mathbb{R}^d$ )构成，环境编码信息由环境编码器对任务场景的点云数据编码后得到，将 $\boldsymbol{f}_{task}$ 输入到一个多层感知机网络，网络输出一个预测结果。

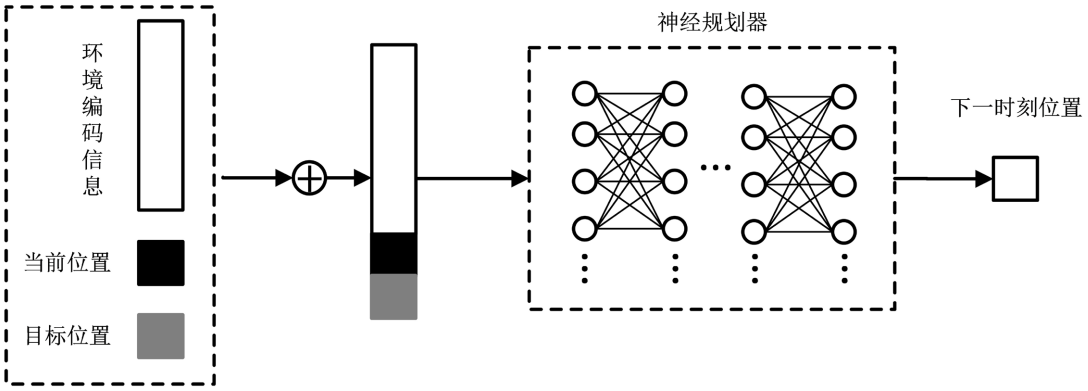



**Figure 3.** Schematic diagram of neural planner structure
**图 3.** 神经规划器结构示意图

## 3. 基于数据增强的神经网络路径规划算法

### 3.1. 基于数据增强的环境编码器训练方法

假设原环境编码器训练数据集由少量任务场景构成，为了丰富训练样本，我们采用平移的方法，将每个场景中的障碍物随机移动，从而生成新的任务场景。具体方法如算法 1 所示。

算法 1：环境编码器训练集构造

输入：任务场景数据集 $x_{obs}$，随机移动阈值 MoveValue

输出：任务场景数据集 $x'_{obs}$

1 $x'_{obs} \leftarrow x_{obs}$
**2 for** $i$←1 to $N$ **do**//$N$ 代表任务场景的个数
3 **for**$j$←1 to $M$ **do**//$M$ 代表第 $i$ 个任务场景的障碍物个数
4 $\boldsymbol{x}$ ←RandomMove(*MoveValue*)
5 $x'_{obs}(i,j) \leftarrow x_{obs}(i,j) + \boldsymbol{x}$ // $x_{obs}(i,j)$代表第 $i$ 个任务场景第 $j$ 个障碍物
6 $x'_{obs}$ ←Contact( $x'_{obs}$， $x_{obs}$ )
**7 return** $x'_{obs}$

环境编码器采用自编码器结构，训练示意图如图 4 所示。编码器输入为任务场景的原始点云表示，输出任务场景的低维表示。解码器输入为任务场景的低维表示，输出为还原的任务场景的原始点云表示。将还原的任务场景点云表示与原始的任务场景点云表示计算重构误差，然后在反向传播后更新网络参数。重构误差可表示为：

$$L_E\left(\theta^e,\theta^d\right)=\frac{1}{N_{pc}}\sum_{i=0}^{N_{pc}}\| \boldsymbol{x}_i-\boldsymbol{x}_i^{'} \|^2+\lambda\sum_{ij}\left(\theta_{ij}^e\right)^2 \tag{3}$$

式 3 中 $\theta^e,\theta^d\partial$ 分别代表编码器和解码器的网络参数； $N_{pc}$ 代表点云的个数，即一个任务场景由多少个点云表示； $x_i,x_i^{'}$ 分别代表重建的点云和真实的点云； $\lambda$ 代表其后正则化项的权重参数。环境编码器训练时的目标是最小化 $L_E\left(\theta^e,\theta^d\right)$，其训练流程如算法 2 所示。

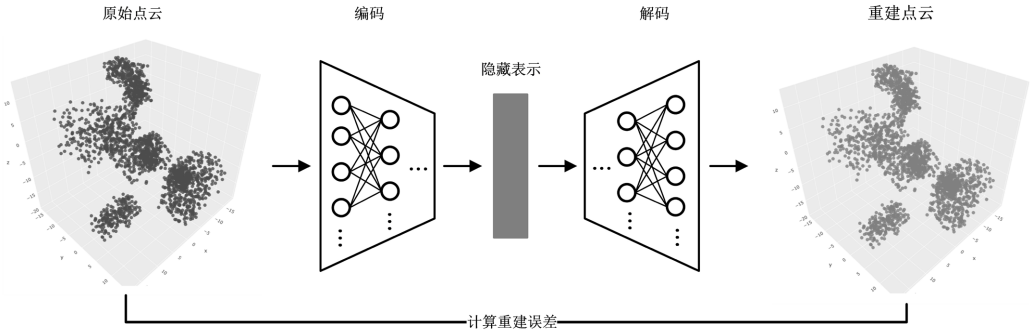



**Figure 4.** Diagram of the practical teaching system of automation major
**图 4.** 环境编码器训练示意图

算法 2：训练环境编码器

输入：训练集 $\mathcal{D}$，正则化权重参数 $\lambda$，学习率 lr, EPOCHS,BATHSIZE
输出：重建点云
1. **for** *ep*←1 to *EPOCHS* **do**
2. **for***i*←1 to *N/ BATHSIZE* **do**
3. Encoder，Decoder 梯度清零
4. 从 $\mathcal{D}$ 中取出 BATHSIZE 个训练样本 $\mathcal{X}_{obs}$
5. $\mathcal{X}_{obs}$ 的低维隐藏表示 $\mathcal{H}$ ←Encoder( $\mathcal{X}_{obs}$ )
6. 重建结果 $\mathcal{D}_{obs}$ ←Decoder( $\mathcal{H}$ )
7. 计算重建误差 $L_E$
8. 反向传播
9. 更新参数 $\theta^e, \theta^d$

## 3.2. 基于数据增强的神经规划器训练方法

我们将一条可行路径看作一个有序列表，那么一条可行路径 $\tau$ 可以表示为由 $t$ 个时刻节点构成的有序列表，即 $\tau = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_t\}$。由于有序列表对列表成员的顺序敏感，有序列表任意成员的顺序发生变化都可视为生成了一个新的列表。于是我们将可行路径 $\tau$ 进行反转操作，结果得到一个新的有序列表 $\tau' = \{\boldsymbol{a}_t, \boldsymbol{a}_{t-1}, ..., \boldsymbol{a}_1\}$。如果只是将 $\tau, \tau'$ 看作两个集合，那么其成员可视为完全等同的，有 $\tau = \tau'$，但事实上针对于路径规划任务而言，$\tau, \tau'$ 应当是两个有序列表，则有 $\tau \neq \tau'$。本节基于上述新的可行路径 $\tau'$ 构造出训练神经规划器的数据增强数据，神经规划器训练集构造的过程如算法 3 所示。

算法 3：神经规划器训练集构造

输入：训练样本集 $\mathcal{D}$，任务场景数据集 $\mathcal{X}_{obs}$，任务场景的隐藏表示集 $\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n\}$，任务场景中可行路径的集 $\mathcal{P} = \{P_i \mid P_i = \{L_1, L_2, ..., L_m\}\}$
输出：下一时刻位置
1. $\mathcal{D} \leftarrow \varnothing$
2. $\mathcal{H}$ ←Encoder( $\mathcal{X}_{obs}$ )
3. $\mathcal{P}$ ←LoadPath()
4. **for** *i*←1 to *N* **do**//*N* 代表任务场景的个数
5. **for***j*←1 to *M* **do**　　//*M* 代表第 *i* 个任务场景的可行路径条数
6. *Length*← $L_j^{P_i}$ .Length()// $L_j^{Pi}$ 代表第 *i* 个任务场景的第 *j* 条可行路径
7. *Reverse*←**False**
8. **if** *Length* > 0 **then**

**Continued**

9. **For** *k*←1 to (*Length*-1)*2 **do**
10. **if** *k* == *Length* **and** *Reverse* == **False then**
11. $L_j^{P_i} \leftarrow L_j^{P_i}$.Reverse()
12. *Reverse*←**True**
13. **end**
14. $\boldsymbol{a}_{goal} \leftarrow L_j^{P_i}$ (*end*)// $L_j^{P_i}$ (*end*)代表第 *i* 个任务场景的第 *j* 条可行路径的终点
15. **if** *Reverse*==**False then**
16. $\boldsymbol{a}_{init} \leftarrow L_j^{P_i}$ (*k*)// $L_j^{P_i}$ (*k*)代表第 *i* 个任务场景的第 *j* 条可行路径的 *k* 时刻节点
17. $\boldsymbol{a}_{next} \leftarrow L_j^{P_i}$ (*k* + 1)
18. **else**
19. $\boldsymbol{a}_{init} \leftarrow L_j^{P_i}$ (*k*- *Length* + 1)
20. $\boldsymbol{a}_{next} \leftarrow L_j^{P_i}$ (*k* + 2- *Length*)
21. **end**
22. $\boldsymbol{f}_{task}$ ←Contact ( $\boldsymbol{h}_i$ , $\boldsymbol{a}_{init}$ , $\boldsymbol{a}_{goal}$ )
23. $\mathcal{D}$ .Append (( $\boldsymbol{f}_{task}$ , $\boldsymbol{a}_{next}$ ))
24. **end**
25. **return** $\mathcal{D}$

神经规划器通过一个多层感知机来拟合路径规划操作函数 $G$，其训练示意图如图 5 所示。将环境编码信息和当前位置、目标位置拼接成一个向量，输入到神经规划器，预测下一时刻位置，而后将预测结果与真实结果计算预测误差，然后在反向传播后更新网络参数。预测误差可表示为：

$$L_P\left(\theta^p\right)=\left\|\boldsymbol{a}_{t+1}-\boldsymbol{a}_{t+1}^{'}\right\|^2 \tag{4}$$

式 4 中 $\theta^p$ 代表神经规划器的网络参数；$\boldsymbol{a}_{t+1}, \boldsymbol{a}_{t+1}^{'}$ 分别代表神经规划器预测的 $t+1$ 时刻位置和真实的 $t+1$ 时刻位置。神经规划器训练时的目标是最小化 $L_P\left(\theta^p\right)$，其训练过程如算法 4 所示。

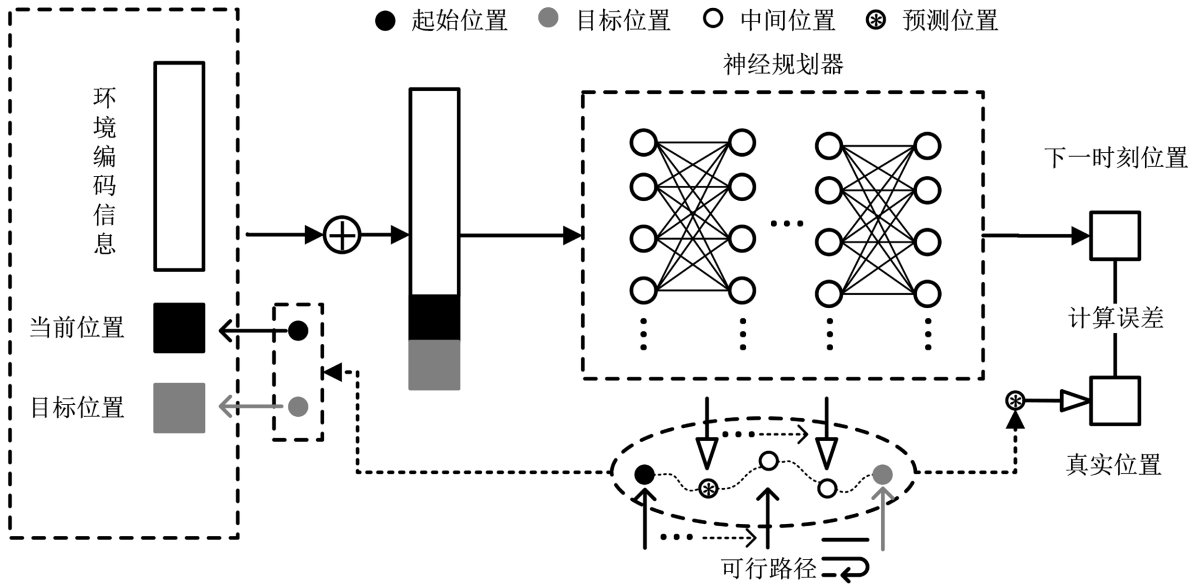



**Figure 5.** Diagram of the practical teaching system of automation major
**图 5.** 神经规划器训练示意图

算法 4：训练神经规划器

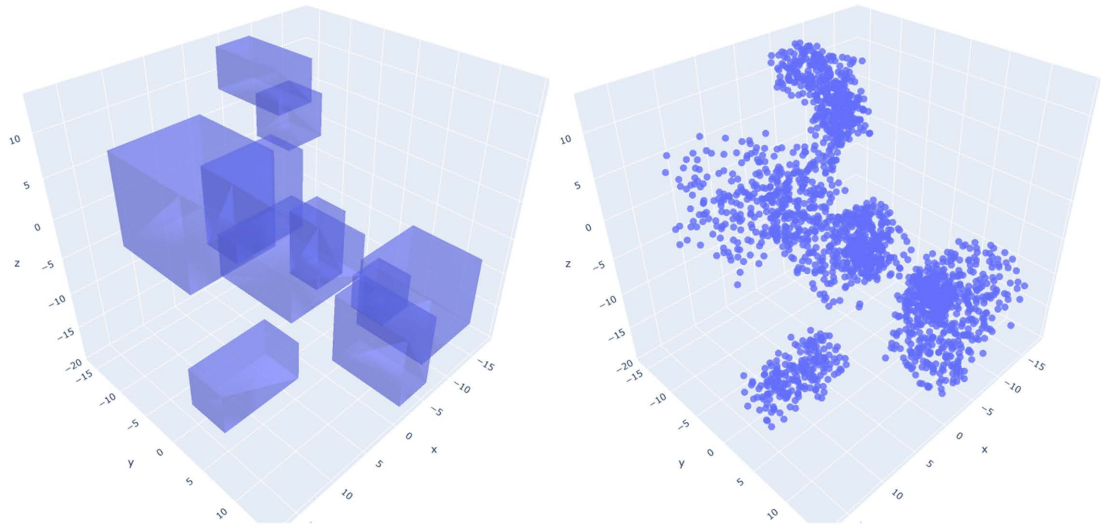
输入：训练集 $\mathcal{D}$，学习率 lr, EPOCHS,BATHSIZE
输出：下一时刻位置
1. **for** *ep*←1 to *EPOCHS***do**
2. **for** *i*←1 to *N/ BATHSIZE* **do**
3. PlannerNet 梯度清零
4. 从 $\mathcal{D}$ 中取出 BATHSIZE 个训练样本 $\mathcal{F}_{task}$
5. 预测结果 $\mathcal{P}_{next}$ ←PlannerNet ( $\mathcal{F}_{task}$ )
6. 计算预测误差 $L_p$
7. 反向传播
8. 更新参数 $\theta^p$

## 4. 实验结果与分析

### 4.1. 数据集准备

本文使用 MPNet 开放的数据集 C3D 作为实验的任务环境，其任务场景如图 6 所示。该数据集包含两部分，一部分是任务场景的点云数据，每个任务场景拥有 10 个障碍物，一共有 30000 个场景的点云数据；另一部分是可行路径数据，数据集提供了 110 个场景的可行路径，其中前 100 个场景，每个场景有 5000 条由 RRT*算法生成的路径，后 10 个场景，每个场景有 2000 条由 RRT*算法生成的路径。在训练环境编码器时，30000 个场景的点云数据都会参与训练；在训练神经规划器时，我们将数据分为见过的场景(Seen)和未见过的场景(Unseen)，Seen 的数据包括可行路径数据中前 100 个场景和与之对应的可行路径，Unseen 的数据包括可行路径数据中后 10 个场景和与之对应的可行路径。

在上述数据集的基础上，我们基于本文提出的两种数据增强方法分别构建了环境编码器和神经规划器的数据增强数据。我们取出 C3D 中 30000 个任务场景的点云数据，然后将场景中的障碍物在其三个维度上进行随机移动，得到了 30000 个新的任务场景点云数据。接下来，我们将 C3D 中提供的每条路径进行反转操作，得到新的可行路径。



（a）C3D中某个任务场景　　（b）C3D中某个任务场景的点云表示

**Figure 6.** Diagram of the practical teaching system of automation major
**图 6.** C3D 场景点云数据集中某一场景的可视化

## 4.2. 实现细节

本文所涉及编程的实验，均采用 Python 语言实现，网络模型由 pytorch 深度学习框架实现。在具体实验中，采用经典的自编码器结构作为环境编码器。编码器的网络结构是一个拥有四层全连接层的多层感知机网络，其大小为 786 × 512 × 256 × 60，输入大小是 6000，输出大小为 60。解码器的网络结构与编码器的网络结构是对称的，其大小为 256 × 512 × 786 × 6000，输入大小是 60，输出大小为 6000。训练时的 batchsize 设置为 100，epoch 设置为 400，使用 Adagrad 作为优化器，学习率设置为 1e-2。计算重建误差时正则化项的权重参数 $\lambda$ 设置为 1。

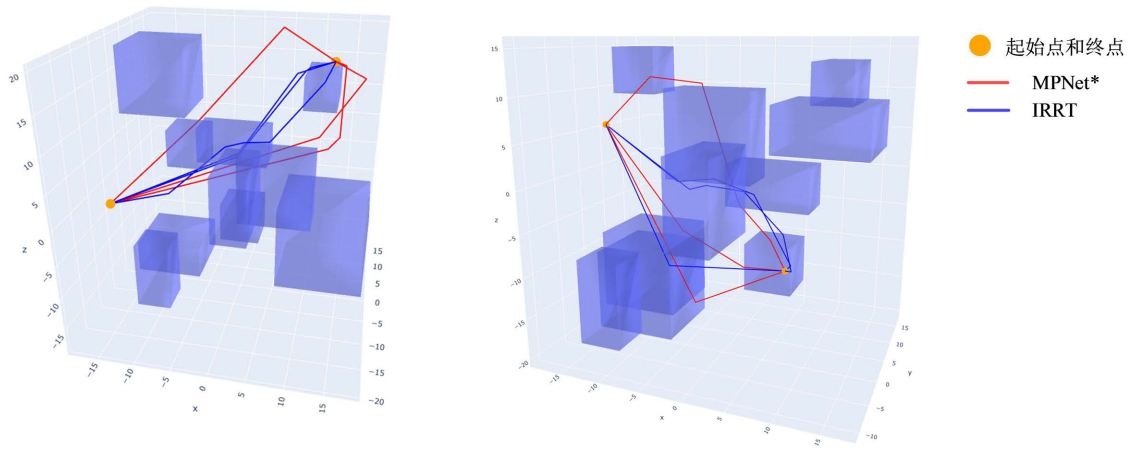
在实现神经规划器时，其网络由 12 层全连接层组成，并对前 11 层的每一层输出进行 PReLU 操作，前 10 层进行 Dropout [11]操作，Dropout 的激活概率设置为 0.5。网络的输入大小是 66，输出大小为 3。训练时的 batchsize 设置为 100，epoch 设置为 500，使用 Adagrad 作为优化器，学习率设置为 1e-2。

为了验证本文提出的方法，我们以 MPNet 为主要模型框架，并在实验部分将 RRT*和 Informed-RRT* (下文简写为 IRRT) [13]两种传统方法以及我们的方法在 Seen 和 Unseen 场景下完成路径规划任务，比较了四种方法的规划时间、规划路径长度。

## 4.3. 结果分析

在下文中，为了方便区分及比较旧模型和新模型，我们将使用仅一种数据增强方法训练的模型称作 MPNet (E*)，MPNet (P*)，分别代表仅在环境编码器训练使用数据增强方法和仅在神经规划器训练使用数据增强方法；将使用两种数据增强方法训练的模型称作 MPNet*。

实验结果表明，我们的方法提高了 MPNet 在 Seen 和 Unseen 场景中的规划成功率，并且在规划时间上也有改善。图 7 展示了 IRRT 算法和 MPNet*在 Seen 和 Unseen 两种场景下规划路径的情况。




（a）Seen场景下规划路径　　（b）Unseen场景下规划路径

**Figure 7.** Diagram of the practical teaching system of automation major
**图 7.** IRRT 和 MPNet*在 C3D 场景中规划的路径

在表 2 中，旧方法训练的 MPNet 在两种场景的规划成功率分别为 97.0%和 93.7%，采用本文提出的方法训练的 MPNet*在 Seen 和 Unseen 场景下的规划成功率分别为 97.7%和 96.5%。MPNet*比 MPNet 在两种场景下分别高出 **0.7%**和 **2.8%**。同时，在仅采用一种数据增强方法的情况下，本文的方法在保证 Seen 场景规划成功率稳定的情况下，对 Unseen 场景的规划成功率相较旧模型也具有一定优势。例如，仅采用数据增强方法的情况下训练出模型 MPNet (P*)，其在 Unseen 场景中的规划成功率为 95.8%，较于旧模型

提高 **2.1%**。

**Table 2.** MPNet and MPNet* planning success in two scenarios
**表 2.** 两种场景下 MPNet 和 MPNet*规划成功率

| 场景 | MPNet | MPNet (E*) | MPNet (P*) | MPNet* |
|---|---|---|---|---|
| Seen | 97.0 ± 0.4 | 97.4 ± 0.4 | 96.5 ± 0.8 | **97.7 ± 0.6** |
| Unseen | 93.7 ± 0.6 | 95.1 ± 0.4 | 95.8 ± 0.2 | **96.5 ± 0.5** |

在表 3 中展示了 MPNet、MPNet*与另外两种传统算法在 Seen、Unseen 场景下的规划时间比较。旧方法训练的 MPNet 在两种场景，结果表明神经网络规划器的规划效率是明显优于两种传统算法的，并且使用本文提出的方法训练的模型相较于旧模型也有一定优势，MPNet*的规划时间较于 MPNet 减少了 0.20 s 左右。

**Table 3.** Planning time of MPNet, MPNet* and traditional algorithms for both scenarios
**表 3.** 两种场景下 MPNet、MPNet*及传统算法的规划时间

| 场景 | MPNet | RRT* | IRRT | MPNet* |
|---|---|---|---|---|
| Seen | 1.43 ± 0.04 | 46.04 ± 1.02 | 8.97 ± 0.70 | **1.21 ± 0.03** |
| Unseen | 1.61 ± 0.09 | 46.15 ± 1.13 | 9.18 ± 0.81 | **1.23 ± 0.06** |

表 4 展示了两种场景下四种方法的规划路径的长度比较，对比结果时以 RRT*算法规划的路径为基线，比较了另外三种算法与其结果的优劣。评估标准如下：在某场景的某个规划任务中，算法 A 与 RRT*各规划出三条可行路径，若算法 A 平均的规划长度小于等于 RRT*的平均的规划长度，则判定算法 A 在此次任务中的规划长度优于 RRT*。若算法 A 和 RRT*比较了 K 个规划任务，并在 N 个任务中优于 RRT*，则算法 A 最后的评估表示为 A 优于 RRT*算法 N/K%。我们在 Seen 和 Unseen 场景分别选取了 100 个规划任务，每个任务规划三次，计算平均长度。实验结果表明，虽然神经网络规划器在规划速度上具有一定优势，但在规划路径长度指标上并不如传统算法。

**Table 4.** Lengths of planned paths for MPNet, MPNet* and traditional algorithms for both scenarios
**表 4.** 两种场景下 MPNet、MPNet*及传统算法的规划路径的长度

| 场景 | RRT* | IRRT | MPNet | MPNet* |
|---|---|---|---|---|
| Seen | - | 53% | 44% | 20% |
| Unseen | - | 46% | 38% | 16% |

## 5. 结束语

路径规划作为机器人学的重要研究方向之一，一直在不断发展。从传统的图搜索算法到基于人工智能的方法和演化算法，各种不同的技术在不同应用场景下展现出强大的优势。随着技术的进步和研究的深入，我们有理由相信神经路径规划算法将在更多领域发挥巨大作用，为人类带来更加智能高效的解决方案。本文提出的两种数据增强的方法，旨在解决训练神经路径规划器所需数据不易收集、数据不足的问题。通过在训练环境编码器以及神经规划器时对训练数据分别采用平移，反转操作，凭此达到丰富训练样本的目的，缓解数据不足的问题。实验对比评估表明，与对比方法相比，本文提出的方法在各个任务场景中都强于对比方法，在规划成功率和时间耗费两个评估指标上都明显优于对比方法。因此，本论文方法满足路径规划任务的需求，具有重要的应用价值。

## 参考文献


[1] La Valle, S.M., Kuffner, J.J. and Donald, B.R. (2001) Rapidly-Exploring Random Trees: Progress and Prospects. *Algorithmic and Computational Robotics*: *New Directions*, **5**, 293-308.

[2] 徐小小, 周启银, 席艳艳, 等. 基于改进 RRT 算法的路径规划[J]. 中国科技信息, 2023(17): 124-127.

[3] 孙海杰, 伞红军, 肖乐, 等. RRT-QSA*: 一种改进的移动机器人路径规划算法[J/OL]. 系统仿真学报, 2023: 1-17. https://doi.org/10.16182/j.issn1004731x.joss.23-0585

[4] 黄丰云, 江仕球, 许建宁. 改进的蚁群算法机器人路径规划研究[J/OL]. 机械设计与制造, 2023: 1-5. https://doi.org/10.19356/j.cnki.1001-3997.20230605.030

[5] 王雷, 王艺璇, 李东东, 等. 基于改进遗传算法的移动机器人路径规划研究[J/OL]. 华中科技大学学报(自然科学版), 2023: 1-13. https://doi.org/10.13245/j.hust.240403

[6] Martin, P. and Del Pobil, A.P. (1970) Application of Artificial Neural Networks to the Robot Path Planning Problem. *WIT Transactions on Information and Communication Technologies*, **6**, 8.

[7] Noguchi, N. and Terao, H. (1997) Path Planning of an Agricultural Mobile Robot by Neural Network and Genetic Algorithm. *Computers and Electronics in Agriculture*, **18**, 187-204. https://doi.org/10.1016/S0168-1699(97)00029-X

[8] Qureshi, A.H., Simeonov, A., Bency, M.J., *et al*. Motion Planning Networks. 2019 *International Conference on Robotics and Automation* (*ICRA*), Montreal, 20-24 May 2019, 2118-2124. https://doi.org/10.1109/ICRA.2019.8793889

[9] Hoang, T. and Vien, N.A. (2021) Graph-Based Motion Planning Networks. In: Hutter, F., Kersting, K., Lijffijt, J. and Valera, I., Eds., *Machine Learning and Knowledge Discovery in Databases*, Springer, Cham, 557-573. https://doi.org/10.1007/978-3-030-67661-2_33

[10] Sung, I., Choi, B. and Nielsen, P. (2021) On the Training of a Neural Network for Online Path Planning with Offline Path Planning Algorithms. *International Journal of Information Management*, **57**, Article ID: 102142. https://doi.org/10.1016/j.ijinfomgt.2020.102142

[11] Trottier, L., Giguere, P. and Chaib-Draa, B. (2017) Parametric Exponential Linear Unit for Deep Convolutional Neural Networks. 2017 16*th IEEE International Conference on Machine Learning and Applications* (*ICMLA*), Cancun, 18-21 December 2017, 207-214. https://doi.org/10.1109/ICMLA.2017.00038

[12] Vincent, P., Larochelle, H., Bengio, Y. and Manzagol, P.-A., *et al*. Extracting and Composing Robust Features with Denoising Autoencoders. *Proceedings of the* 25*th International Conference on Machine Learning* (*ICML*'08), New York, 5-9 July 2008, 1096-1103. https://doi.org/10.1145/1390156.1390294

[13] Gammell, J.D., Srinivasa, S.S. and Barfoot, T.D. (2014) Informed RRT: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. 2014 *IEEE*/*RSJ International Conference on Intelligent Robots and Systems*, Chicago, 14-18 September 2014, 2997-3004. https://doi.org/10.1109/IROS.2014.6942976