remote sensing

MDPI

Article

AgentI2P: Optimizing Image-to-Point Cloud Registration via Behaviour Cloning and Reinforcement Learning

Shen Yan, Maojun Zhang *, Yang Peng, Yu Liu and Hanlin Tan

Department of System Engineering, National University of Defense Technology, Deya Road 107, Changsha 410000, China

* Correspondence: mjzhang@nudt.edu.cn; Tel.: +86-1331-950-6071

Abstract: Image-to-point cloud registration refers to finding relative transformation between the camera and the reference frame of the 3D point cloud, which is critical for autonomous driving. Recently, a two-stage "frustum point cloud classification + camera pose optimization" pipeline has shown impressive results on this task. This paper focuses on the second stage and reformulates the optimization procedure as a Markov decision process. An initial pose is modified incrementally, sequentially aligning a virtual 3D point observation towards a previous classification solution. We consider such an iterative update process as a reinforcement learning task and, to this end, propose a novel agent (AgentI2P) to conduct decision making. To guide AgentI2P, we employ behaviour cloning (BC) and reinforcement learning (RL) techniques: cloning an expert to learn accurate pose movement and reinforcing an alignment reward to improve the policy further. [We demonstrate the effectiveness and efficiency of our approach on Oxford Robotcar and KITTI datasets. The (RTE, RRE) metrics are $(1.34\,\text{m}, 1.46°)$ on Oxford Robotcar and $(3.90\,\text{m}, 5.94°)$ on KITTI, and the inference time is 60 ms, both achieving state-of-the-art performance.] The source code will be publicly available upon publication of the paper.

Keywords: multi-modal registration; re-localization; reinforcement learning

Citation: Yan, S.; Zhang, M.; Peng, Y; Liu, Y.; Tan, H. AgentI2P: Optimizing Image-to-Point Cloud Registration via Behaviour Cloning and Reinforcement Learning. *Remote Sens.* **2022**, *14*, 6301. https://doi.org/10.3390/rs14246301

Academic Editor: Soe Myint

Received: 12 October 2022
Accepted: 3 December 2022
Published: 12 December 2022

1. Introduction

Image-to-point cloud registration aims at predicting the orientation and position of an imaging device relative to a general point cloud [1,2]. It is a fundamental block for many applications in computer vision, such as autonomous driving [3–5], self-navigation [6–8], and augmented/mixed reality (AR/MR) systems [9,10].

A typical pipeline of most existing methods [11,12] is first to reconstruct a 3D model by multiple views and subsequently find correspondences between 2D pixel positions and relevant 3D scene coordinates based on the 2D to 3D back-projection features. Camera poses are then estimated using a standard perspective-n-point(PnP) solver inside a robust RANSAC loop. While offering high precision, such approaches require a structure-from-motion (SfM) technique [13] to pre-build the reference 3D map. However, accurate SfM reconstruction is not always achievable, such as under textureless or snowy road conditions. More importantly, this work line treats RGB images as the first principle and discounts active sensors capable of directly capturing 3D point clouds in harsh environments, such as Lidar.

The problem above poses an open question: is it possible to directly estimate the extrinsic parameters of a vehicle to a 3D point cloud without any extra steps (e.g., image-based reconstruction). It is quite an arduous task because, unlike image-to-image feature matching, building common keypoint representations cross image to point cloud is inherently troublesome as images capture scene appearance, whereas point clouds encode structure. Recently, DeepI2P [14] cleverly circumvents the difficulty by casting the cross-modality registration problem into a two-stage classification and

optimization framework, which acquires promising results. The first stage employs a dual-branch neural network (taking the image and point cloud as input) to predict each point cloud within or beyond the camera frustum. In the second stage, the objective is to find the optimal camera pose, such that 3D points predicted inside the camera frustum are correctly projected into the image plane. By formulating as an unconstrained continuous optimization problem, DeepI2P uses standard solvers such as the Gauss–Newton algorithm [15] to provide a solution.

However, selecting a proper initial pose for the Gauss–Newton solver during the optimization phase is not easy. To release the problem, DeepI2P simply performs the optimization 60 times with randomly generated initialization and finally chooses the solution with the lowest cost. We have to say this naive repetitive operation is highly time-consuming but also cannot guarantee convergence to the global minima. In addition, to improve registration accuracy, DeepI2P lowers the full 6-DoF search space to 3-DoF, leveraging on the fact that the testing datasets are from ground vehicles. This trick will lose the generalization when adapting to other different scenarios.

This paper instead frames the pose optimization procedure as an iterative Markov decision process. Given an arbitrary initial pose, our core idea is to learn to move virtual cameras sequentially until the camera-observed point cloud (Source) aligns with the network-classification point cloud (Target). The insight of our work is illustrated in Figure 1. Specifically, the Source and Target pairs are first randomly sampled and fed to a lightweight policy network to embed the current state and output 6-DoF transformation. After that, the former pose is updated according to the network output, and a novel-viewpoint observation is fed back into the pipeline again for the sake of gradually advancing the Source points towards the Target. At last, this process ends when the maximum iteration is reached or a stop signal triggers from the policy network. In other words, our approach can be viewed as learning to optimize. The network uses [several] independent update blocks to emulate the steps of traditional first-order or second-order optimization [algorithms].

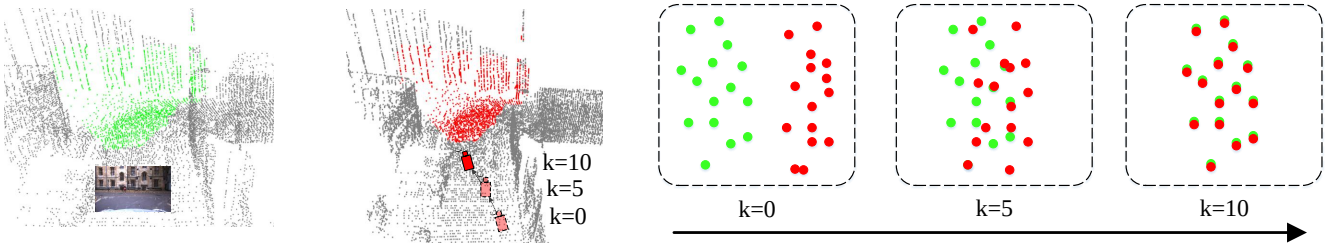

Figure 1. The basic insight of our method. The left figure shows the classification point cloud (green) from a pre-trained network. The middle figure shows the trajectory of estimated poses and the observation point cloud (red) in AgentI2P's iterative pose estimation procedure. The right figure shows the observation point cloud (red) is aligned to the classification point cloud (green) along with several camera pose adjustments.

To accomplish the pipeline mentioned above, we propose a novel agent named AgentI2P, which divides the pose optimization procedure into a sequence of multi-categorical classification sub-steps, as shown in Figure 1. To train the AgentI2P, we combine a behaviour cloning (BC) [16] technique with reinforcement learning (RL) [17]. BC minimizes a cross-entropy loss between expert annotation and network classification for each camera restoration trajectory. At the same time, RL maximizes accumulated rewards regarding the distance between the Source and the Target. The joint training improves the convergence of AgentI2P compared to simply selecting one mode. Advanced RL algorithms not only explore the action space but also exploit the knowledge

at a given state. The wide exploration implies that selecting a proper initial pose is not that important, which leads to a high robustness, and the exploitation means our agent retrieves features from the network to propose the descent direction, where the local [minimum] problem can be effectively reduced, which contributes to high accuracy. Moreover, adopting a deep neural network to infer camera pose refinement [iteratively] brings significant acceleration compared to the traditional solver. In Figure 2, we illuminate the detailed difference between prior optimization work and our RL-based method. Through extensive and detailed experiments on Oxford Robotcar and KITTI datasets, AgentI2P demonstrates an overwhelming advantage over its Gauss–Newton counterpart concerning accuracy, robustness, and running time.

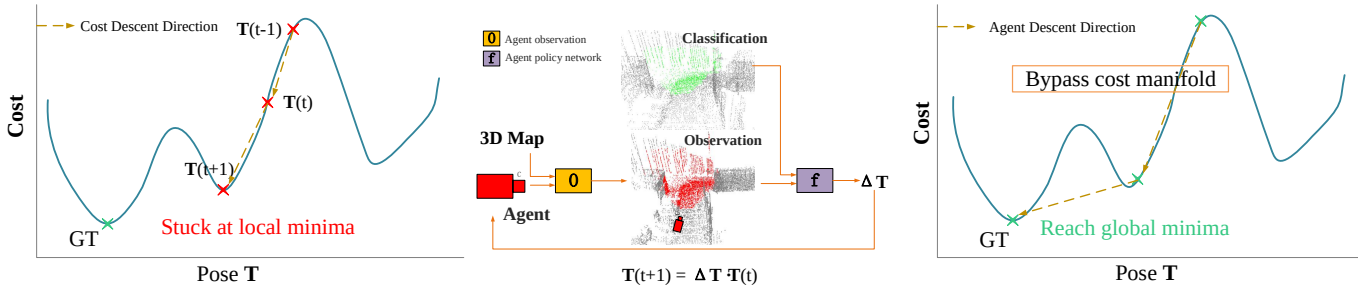

Figure 2. The difference between our approach and classic optimization approaches: (**a**) Prior structured methods rely on the cost function and are sensitive to initializations which makes them easy to become stuck in local optima. (**b**,**c**) Our deep agent iteratively updates the 6-DoF action based on current observation in a 3D environment, which not only eliminates local extrema but also significantly reduces the number of iterations.

The main contributions of this paper are summarized as follows:

- We are the first to reformulate image-to-point registration as an iterative action decision process and propose a novel agent for this task.
- We adopt a lightweight PointNet-like architecture for the agent and utilize behaviour cloning and reinforcement learning to train the model.
- We improve the registration accuracy and reduce the inference time by a large margin compared with related approaches, which adds value to the autonomous vehicle research field.

2. Related Work

2.1. Image-to-SfM Model Registration

Conventionally, most existing methods [11,12,18–20] of visual camera registration depend on local feature matching in the 2D domain. First, SfM systems [13] are employed to generate a sparse point cloud of a scene, where each 3D point is attached with feature descriptors inherited from the corresponding pixel of the image sequence. Afterward, 2D-3D correspondences are established between pixels and points based on the 2D to 3D back-projection features. Finally, a RANSAC-based PnP solver is used to compute relative rotation and translation. Such techniques have already been applied in visual re-localization [21] and modern SLAM systems [22]. An alternative is to directly train a convolutional neural network (CNN) to regress geometric quantities, such as camera absolute pose [23–29] or the 3D coordinate regression, upon each pixel [30–37]. Unfortunately, such methods are scene-specific and require to be trained for new scenes [34].

While sparse feature matching can achieve high accuracy, SfM reconstructions tend to fail in environments dominated by ambiguous, textureless, and repeating structures. Fortunately, in such cases, 3D scanner devices (e.g., Lidar) can still provide a high-precision point cloud. As a result, our work centers on the general image-to-point cloud registration task without any additional image-based features.

2.2. Image-to-Point Cloud Registration

Unlike numerous studies focusing on image-to-SfM registration, little attention has been paid to predicting the relative pose between image and point cloud coordinates. Among them, several methods [38,39] prefer to directly establish pixel-to-point correspondences, motivated by the well-researched area of same-modality feature matching. However, as points in 3D space usually share tiny appearance and geometric correlations with the image in 2D space, the experimental settings of those methods have experienced significant simplifications. Specifically, 2D3D-MatchNet [38] samples images and point cloud pairs from a driving platform at nearby timestamps with almost no relative rotation, while P2-Net [39] is trained and tested on a small-scale dataset [36], and its 3D maps are of high density. Another work [40] exploits line correspondences between images and Lidar maps to refine 6-DoF camera pose, but they require precise initialization from a SLAM system. Some other works [41,42] utilize 2D-to-3D matching information to implement place recognition rather than determining the orientation and position of a camera.

Recently, DeepI2P [14] provides a two-stage classification and optimization solution, which circumvents the challenging need to learn cross-modal detectors and descriptors. However, the second stage of DeepI2P is sensitive to initial pose guess and may converge to a local and incorrect minimum. Therefore, this paper investigates how to make the optimization process more robust, accurate, and efficient.

2.3. Reinforcement Learning for Vision Tasks

Over the past few years, deep reinforcement learning [17] has been applied to a variety of computer vision tasks, ranging from semantic segmentation [43], object detection [44], to point cloud registration [45], etc. Several methods to solve object pose estimation-related tasks via RL are most relevant to our work. For instance, refs. [46,47] train policies to determine incremental movements to bring the object rendering closer to the real observation. Those agents treat rendered, observed, and 2D object masks together as input and learn to output discrete refinement actions.

We declare that such an iterative update formulation suits our cross-modality registration, although they belong to two separate fields of studies. The major distinction is that the input to our policy network is no longer dense images but irregular and orderless point clouds. Moreover, different from [46,47] who learn 6-DoF pose refinement in a pose-free fashion, we additionally supervise the pose renovation with ground-truth expert annotations, thus significantly enhancing the registration performance.

Recently, ref. [45] tried to estimate the relative pose of the same 3D object with point cloud as input tensor, which has something in common with our method. However, AgentI2P shares notable differences with [45]. For instance, (1) Goal: AgentI2P engages in finding a camera orientation and position relative to a point cloud map, while [45] concentrates on estimating a relative rigid motion between two point clouds. Even though only considering Target and observed point of AgentI2P, our work is not to compute their relative rigid transformation. (2) Agent: AgentI2P treats the virtual camera as the agent to take action, while [45] chooses to control the Source point cloud. (3) Environment: during each trajectory, AgentI2P non-differentially interacts with the Lidar map (environment) to obtain dynamic-changed observations, while point cloud shape in [45] remains identity. (4) Coordinate system: AgentI2P infers camera movement after converting the Target and observed point into current camera coordinate (necessary for camera pose refinement), while [45] simply handles point cloud in world coordinate.

3. Methodology

3.1. Overview

We denote an image as $I \in \mathbb{R}^{3\times W\times H}$, where W and H are image width and height, and point cloud as $P = \{\mathbf{P}_1, \cdots, \mathbf{P}_i, \cdots, \mathbf{P}_N | \mathbf{P}_i \in \mathbb{R}^3\}$, where N is the total number. The objective of image-to-point cloud registration is to solve for the relative rotation matrix

$\mathbf{R} \in \mathbf{SO}(3)$ and translation vector $\mathbf{t} \in \mathbb{R}^3$ between the image I and the reference frame of the 3D point cloud P.

To find the rigid transformation, a two-stage "frustum point cloud classification + camera pose optimization" pipeline is proposed by DeepI2P [14]. The first stage classifies each point $\mathbf{P}_i \in P$ inside or outside the camera frustum using a double-branch deep network. The second stage optimizes the camera pose $\{\mathbf{R}, \mathbf{t}\}$ until [the] observed point cloud aligns with frustum classification results.

This paper reformulates the later pose optimization phase as a Markov decision process (MDP) and proposes a novel reinforcement learning-based agent called AgentI2P to tackle this problem. In the following sections, we first briefly introduce the background knowledge related to our work in Section 3.2. We then describe the lightweight and interpretable network architecture of AgentI2P in Section 3.3. Finally, Section 3.4 presents the learning procedure that enables accurate and robust registration.

3.2. Background Knowledge

Pose refinement process. The pose refinement process refers to modifying the camera extrinsic matrix step-by-step, which could be abstracted as an operations chain regarding camera parameters. Without loss of generality, we denote an update from step k to step $k+1$ as:

$$\begin{aligned} \mathbf{G}_{k+1} &= \begin{bmatrix} \mathbf{R}_{k+1} & \mathbf{t}_{k+1} \\ \mathbf{0} & 1 \end{bmatrix} = \Delta\mathbf{T}_k \otimes \mathbf{G}_k \\ &= \begin{bmatrix} \Delta\mathbf{R}_k & \Delta\mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{1}$$

where $\mathbf{G}_k \in \mathbf{SE}(3)$ and $\mathbf{G}_{k+1} \in \mathbf{SE}(3)$ represent camera pose at step k and $k+1$, and $\Delta\mathbf{T}_k \in \mathbf{SE}(3)$ means relative transformation from step k to step $k+1$. The whole pose refinement process can be signified as:

$$\mathbf{G}_n = \Delta\mathbf{T}_{n-1} \otimes \cdots \otimes \Delta\mathbf{T}_1 \otimes \mathbf{G}_1, \tag{2}$$

where $\mathbf{G}_1$ and $\mathbf{G}_n$ indicate initial and final pose, respectively. The operator $\otimes$ may follow different conventions, such as the basic matrix product. In this work, we follow [45,47] to disentangle rotation and translation, and rewrite Equations (1) and (2) as:

$$\begin{aligned} \mathbf{G}_{k+1} &= \begin{bmatrix} \Delta\mathbf{R}_k & \Delta\mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \Delta\mathbf{R}_k\mathbf{R}_k & \Delta\mathbf{t}_k + \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{3}$$

$$\mathbf{G}_n = \begin{bmatrix} (\prod_{k=1}^{n-1} \Delta\mathbf{R}_k)\mathbf{R}_1 & (\sum_{k=1}^{n-1})\Delta\mathbf{t}_k + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{bmatrix}. \tag{4}$$

Such operation does not need to account for the rotation-induced translation, which is beneficial for [the] training of the agent.

Point cloud observation and classification. Point cloud observation aims at figuring out what points can be noticed by the camera, which is a pinhole projection problem. Given a point $\mathbf{P}_i \in \mathbb{R}^3$, camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$, and camera extrinsic parameters $\{\mathbf{R}, \mathbf{t}\}$, we determine the point $\mathbf{P}_i$ observed or not as follows:

First, we transfer $\mathbf{P}_i$ from the point cloud coordinate to the image homogeneous coordinate $\tilde{\mathbf{p}}_i$ by

$$\tilde{\mathbf{p}}_i = \begin{bmatrix} \tilde{p_{x_i}} \\ \tilde{p_{y_i}} \\ \tilde{p_{z_i}} \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} \mathbf{P}_i \\ 1 \end{bmatrix}. \tag{5}$$

Then, we recover the inhomogeneous coordinate $\mathbf{p}_i$ of the image point by

$$\mathbf{p}_i = [p_{x_i}, p_{y_i}]^T = [\tilde{p_{x_i}}/\tilde{p_{z_i}}, \tilde{p_{y_i}}/\tilde{p_{z_i}}]^T. \tag{6}$$

Finally, we define a discriminative function:

$$f_o(\mathbf{P}_i; \mathbf{K}, \mathbf{R}, \mathbf{t}, W, H) = \begin{cases} 1 & \text{if } 0 \le p_{x_i} \le W-1, 0 \le p_{y_i} \le H-1, \tilde{p_{z_i}} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

which assigns a label of 1 to the point $\mathbf{P}_i$ that is observed by the camera, and 0 otherwise. For any camera pose $\{\mathbf{R}_k, \mathbf{t}_k\}$, we could label the observed point cloud as $\hat{L}_k^o = \{\hat{l}_{k_1}^o, \cdots, \hat{l}_{k_i}^o, \cdots, \hat{l}_{k_N}^o\} = f_o(\{\mathbf{P}_i\}; \mathbf{K}, \mathbf{R}_k, \mathbf{t}_k, W, H)$, and obtain observation points $P_k^o = \{\mathbf{P}_i | \hat{l}_{k_i}^o = 1\}$.

As for point cloud classification from DeepI2P, the dual-branch network assigns a binary label to each point $\hat{L}^c = \{\hat{l}_1^c, \cdots, \hat{l}_i^c, \cdots, \hat{l}_N^c\}, \hat{l}_i^c \in \{0, 1\}$, where $\hat{l}_i^c = 1$ indicates the projection of point $\mathbf{P}_i$ is predicted within image plane, and vice versa. Based on the network output, the classification points can be symbolized as $P^c = \{\mathbf{P}_i | \hat{l}_i^c = 1\}$.

It is worth mentioning that before importing to our agent, point clouds are transferred from world coordinates to camera coordinates for normalization, which is necessary for camera pose refinement. Assume that at time k, the camera has been moved to $\{\mathbf{R}_k, \mathbf{t}_k\}$, and the 3D transformation of a point $\mathbf{P}_i$ is given by

$$\mathbf{P}_i' = \mathbf{R}_k \mathbf{P}_i + \mathbf{t}_k, \tag{8}$$

we remark coordinate-altered point cloud as $P_k^{o\prime} = \{\mathbf{P}_i' | \hat{l}_{k_i}^o = 1\}$ for observation (denoted as Source) and $P_k^{c\prime} = \{\mathbf{P}_i' | \hat{l}_i^c = 1\}$ for classification (denoted as Target). The purpose of AgentI2P is to modify camera pose $\{\mathbf{R}_k, \mathbf{t}_k\}$ incrementally as described in [the] previous subsection, until Source points $P_k^{o\prime}$ overlap with the Target points $P_k^{c\prime}$.

3.3. Agent Architecture

The overall architecture for our proposed agent is demonstrated in Figure 3, which has two key modules for (1) state representation and (2) action estimation. To begin with, to encode the agent's information about the current state, the state representation stage manages feature embedding that transfers and concatenates [the] original point cloud (namely Source $P_k^{o\prime}$ and Target $P_k^{c\prime}$) into global feature vector S_k. Then, the action estimation stage predicts two action vectors, one for rotation $a_R(S_k) \sim \pi_R(S_k)$ and one for translation $a_t(S_k) \sim \pi_t(S_k)$, by following a certain policy $\{\pi_R, \pi_t\}$ conditioned on state S_k. Lastly, the predicted action is applied to camera pose $\{\mathbf{R}_k, \mathbf{t}_k\}$ again, iteratively moving it towards the final object $\{\mathbf{R}^*, \mathbf{t}^*\}$. In each refinement step, the agent receives a guidance $\{a_R^*(S_k), a_t^*(S_k)\}$ from an expert, and an immediate reward r_k that judges how well it performs its task. The following subsections describe individual modules with their associating layers in detail.

State representation. Since $P_k^{o\prime}$ and $P_k^{c\prime}$ are a set of points that are irregular and orderless, we follow a PointNet-like structure [48] $\Phi(\cdot)$ to extract robust and powerful representations $\Phi(P_k^{o\prime}), \Phi(P_k^{c\prime})$ for them. The embedding module $\Phi(\cdot)$ comprises shared multilayer perceptron (MLP) layers and max-pooling functions, mapping raw point cloud from $N \times 3$ to $1 \times M$ dimensional state space. The concatenation of both global feature vectors is used to document the current state $S_k(\Phi(P_k^{o\prime}), \Phi(P_k^{c\prime}))$.

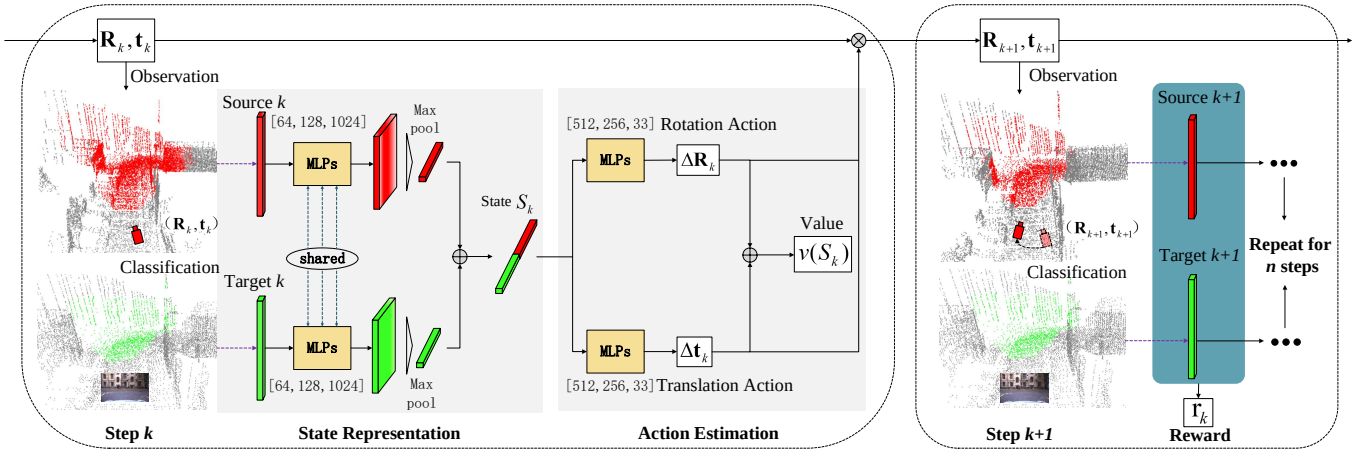

Figure 3. The overall architecture for one iteration of AgentI2P. At each step k, given the present camera pose $\mathbf{R}_k, \mathbf{t}_k$ and dual-classification result, two frames of point clouds (Observation Source and Classification Target) are [treated] as the input. The policy network repetitively learns to embed the current state and predict the update action along rotation and translation axes, which promotes the virtual camera to the next pose. Please see Section 3.3 for more details on the network modules.

Action estimation. Action is a 6-DoF transformation the agent performs at each step k, which can be decomposed into a vector of 6 independent sub-actions along rotation and translation axes. In order to avoid divergence and robustify update steps, we follow [45] to use discrete and limited step sizes in each iteration. In such a case, we handcraft 11 symmetrical operations for every sub-action, where each of them includes 5 positive and 5 negative movements from small to large degrees and 1 "cease" instruction. The differing scale of step sizes contributes to quickly recovering during initial optimization while allowing fine-grained adjustments in later steps.

Given a state S_k, the agent's policy $\{\pi_R(S_k), \pi_t(S_k)\}$ gives the probability of selecting action $\{a_R(S_k), a_t(S_k)\}$. The policy $\{\pi_R(S_k), \pi_t(S_k)\}$ comes from the agent's action head with a format of the multi-categorical distribution. Moreover, to promote reinforcement learning, a value score $v(S_k)$ is estimated from the agent's value head.

3.4. Learning Procedure

Through practical experiments, we notice that only using RL to align Source and Target leads to having a suboptimal policy. This may be because the given reward function (described in the following) only focuses on step-wise improvement, and it probably biases the agent to find the solution with the most small improvements in MCD than the lowest MCD itself. To solve this problem, we additionally introduce BC to supervise the backpropagation of the agent network. In this part, we present our joint learning procedure with a combination of BC and RL techniques. We believe the objective of BC is to obtain the lowest MCD, as a correct camera pose corresponds to a low MCD, and the objective of RL is designed to make the whole policy optimization more stable.

Behaviour Cloning (BC). In BC, the goal is to imitate the behavior of some domain experts. In particular, for the image-to-point cloud registration task, the expert annotation refers to the true $\mathbf{SE}(3)$ transformation at each step during [the] training phase. We obtain the true transformation at each step by the following disentangle relative pose computation:

$$\Delta\mathbf{R}_k^* = \mathbf{R}^*\mathbf{R}_k^{-1}, \Delta\mathbf{t}_k^* = \mathbf{t}^*\mathbf{t}_k^{-1}, \tag{9}$$

where $\Delta\mathbf{R}_k^*, \Delta\mathbf{t}_k^*$ imply accurate $\mathbf{SE}(3)$ transformations. The expert policy selects the most possible action $a_R^*(S_k), a_t^*(S_k)$ from 11 manually defined misalignment bins that largely reduces the absolute error concerning $\Delta\mathbf{R}_k^*, \Delta\mathbf{t}_k^*$. Through the automatic annotation from a so-called expert, we can train the agent using a 6-dimensional cross-entropy loss.

Reinforcement Learning (RL). In RL, the goal is to maximize the expected sum of future discounted rewards. In the ideal course, the overlap rate between Source and Target should experience a monotonous increase along with ongoing iteration. Based on this motivation, we define a step-wise reward function as:

$$r_k = \begin{cases} -\sigma^- & \text{if } MCD(P_k^{o\prime}, P_k^{c\prime}) < MCD({P_{k+1}^o}', {P_{k+1}^c}') \\ -\sigma^0 & \text{if } MCD(P_k^{o\prime}, P_k^{c\prime}) = MCD({P_{k+1}^o}', {P_{k+1}^c}'), \\ +\sigma^+ & \text{if } MCD(P_k^{o\prime}, P_k^{c\prime}) > MCD({P_{k+1}^o}', {P_{k+1}^c}') \end{cases} \tag{10}$$

where we employ Mean Chamfer Distance MCD to measure the degree of the point cloud overlap. Actions that reduce MCD are rewarded by $+\sigma^+$, while steps that raise MCD are penalized by $-\sigma^-$, and "cease" receives a negative punishment $-\sigma^0$ to discourage suspending. We believe unlike directly minimizing the MCD, the designed objective is easier for the agent to handle.

Policy optimization. Generally, deep RL agents incrementally update their network parameters when observing a stream of experience in the simplest form. However, the temporal correlations break the independent and identically distributed assumption of our stochastic gradient-based algorithm. To address this issue, we set up a replay buffer to store offline trajectories. Using the replay buffer could also fully explore the training samples and reduce the amount of experience required to learn. Furthermore, to make the policy learning perceive the reliability of the trust-region, we use the well-known proximal policy optimization (PPO) [49] algorithm. Algorithm 1 summarizes the main procedure of our method.

Algorithm 1 The Joint Behaviour Cloning and Reinforcement Learning using a Replay Buffer.

for for all observation **do**
  % Gather replay buffer
  for each trajectory l in L **do**
    set $k = 0$;
    while $k < n$ **do**
      agent outputs policy $\pi(S_k^{(l)})$ and value $v(S_k^{(l)})$;
      action $a(S_k^{(l)})$ is sampled from policy $\pi(S_k^{(l)})$
      take action $a(S_k^{(l)})$, obtain reward $r_k^{(l)}$ and next state S_{k+1};
      add sample to buffer b and set $S_k = S_{k+1}$;
      set $k = k + 1$;
    end while
  end for
  % Process replay buffer
  compute return R and advantage A, shuffle buffer b;
  for each sample i in buffer b **do**
    agent outputs new policy $\pi^{new}(S_i)$ and new value score $v^{new}(S_i)$;
    % Behaviour Cloning
    expert annotates action $a^*(S_i)$;
    compute cross-entropy loss L_{bc} from $\pi^{new}(S_i)$ and $a^*(S_i)$;
    % Reinforcement Learning
    compute PPO loss L_{ppo} from $\pi^{new}(S_i), v^{new}(S_i)$ and $\pi(S_k^{(l)}), v(S_k^{(l)})$;
    % Update Network
    $L_{all} = L_{bc} + \lambda \cdot L_{ppo}$;
    backpropagate overall loss L_{all};
  end for
  clear buffer b;
end for

Please note that the main claim of the paper is not to propose a novel RL and BC framework to train the agent. AgentI2P employs the standard and commonly used PPO [49] and BC [50] techniques with a replay buffer for our agent training.

3.5. Abbreviated Proper Nouns

In order to promote easy reading, we construct a separate noun list in Table 1.

Table 1. This table provides a separate abbreviated proper nouns list for AgentI2P.

Nouns	Section	Description
MDP	Section 3.1	Markov Decision Process: a model of predicting outcomes, in which an outcome is given only information provided by the current state
SO(3)	Section 3.3	The group of all rotations about the origin of three-dimensional Euclidean space $\mathbb{R}^3$
SE(3)	Section 3.2	The group of simultaneous rotations and translations for a vector of three-dimensional Euclidean space $\mathbb{R}^3$
BC	Section 3.4	Behaviour Cloning: a method by which expert annotations can be captured and reproduced by a computer program
RL	Section 3.4	Reinforcement Learning: intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward
PPO	Section 3.4	Proximal Policy Optimization: a model-free reinforcement learning algorithms, which searches the space of policies by policy gradient
MCD	Section 3.4	Mean Chanfer Distance: for each point in each cloud, MCD finds the nearest point in the other point set and compute the average

4. Experiments

To demonstrate the effectiveness of the proposed approach, we conduct experiments on Oxford Robotcar [51] and KITTI [52] datasets. As our method works on improving DeepI2P [14], we follow the same experimental setting for a fair comparison. For further details about the image-to-point pair construction, we refer readers to DeepI2P [14].

In total, for the Oxford Robotcar dataset, 35 traversals are used for training, while 5 traversals are for testing. Specifically, there are 130,078 pairs for training and 19,156 for testing. By contrast, for the KITTI dataset, 0–8 sequences are used for training, and 9–10 are used for testing, and there are 20,409 pairs for training and 2792 for testing.

4.1. Implement Details

Agent parameters. The embedding network $\Phi(\cdot)$ consists of $[64, 128, 1024]$ MLPs, which results in a 2048 dimensional state vector, and we implement the action and value heads using fully connected layers of size $[512, 256, D]$, where D equals 33 for orientation and position and 1 for value score.

For each sub-action, there are 11 misalignment bins $[\pm 0.27, \pm 0.09, \pm 0.03, \pm 0.01, \pm 0.0033, 0]$ in common, except translation on the z axis. As the ground vehicle dataset shows significant movement along the road, we set step sizes $[\pm 2.7, \pm 0.9, \pm 0.3, \pm 0.1, \pm 0.033, 0]$ for z translation. For rotation, step sizes are interpreted in radians.

Training parameters. The total number N of the Source and Target point cloud is sampled and fixed to 4096. For the reward function, we experimentally determine $(\sigma^+, \sigma^0, \sigma^-) = (0.5, 0.1, 0.6)$. Additional parameters are the rollout trajectories per buffer $L = 4$, the maximum count of pose refinement $n = 10$, the discount factor $\gamma = 0.99$, and the loss trade-off term $\lambda = 1.0$.

The agent is trained using Adam with an initial learning rate of 0.001 and a batch size of 512. Moreover, the model ultimately converges after 30 epochs on a single RTX 3090 with randomly initialized weights.

4.2. Baseline Methods

We compare our AgentI2P with four categories of approaches:

- Regression methods, including direct relative pose regression and grid classification + PnP. For the former, a deep network is used to directly regress the relative poses. For the latter, after labeling an inside camera frustum 3D point to image grid, the RANSAC PnP algorithm optimizes the full 6-DoF pose without any constraints.
- The 3D registration methods, including Monodepth2 [53] + ICP [54] and Monodepth2 [53] + USIP [55]. Note that the scale of the monocular depth is perfectly calibrated using the Lidar point cloud. For the ICP algorithm, we randomly select 60 initial seeds and choose the solution with the lowest cost.
- Feature matching method, including 2D3D-MatchNet [38]. However, we have to say, their experimental setting is much easier as the rotation between the camera and Lidar is almost zero.
- DeepI2P [14], which is closely related to our method, as we are under the same "classification + optimization" framework, and our AgentI2P only replaces its Gauss–Newton (GN) counterpart.

For a fair comparison between AgentI2P and DeepI2P, we conducted controlled experiments with the same frustum classification input. Specifically, for the Oxford dataset, we gain the classification results from DeepI2P's pre-trained model in the official repository (https://github.com/lijx10/DeepI2P, accessed on 1 October 2022). For the KITTI dataset, as the pre-trained model is not available, we retrained the classification network under DeepI2P's official document instrument. Unfortunately, the reproduction did not reach the performance as reported in the original paper [14].

4.3. Evaluation Protocol

The evaluation protocol considers two aspects, namely accuracy, and efficiency. Following [14], we report registration accuracy by two criteria: average relative translational error (RTE) and average relative rotation error (RRE). The mean and standard deviation of those two statistics are presented. Furthermore, to exhibit the testing efficiency, we carefully report the pose optimization duration of our AgentI2P and the DeepI2P's Gauss–Newton (GN) solver.

4.4. Results

Accuracy. Table 2 demonstrates the evaluation results of AgentI2P in terms of accuracy. Our approach achieved the best performance in the Oxford dataset compared to all competitors. We also outperformed other baselines in the KITTI dataset, except that Grid. Cls. + PnP slightly led us in RTE. We attribute the top performance to the robustness to initial guess and the dynamic coarse-to-fine pose update.

In particular, the performance gap was more evident on RRE, where our AgentI2P was ahead of other methods exponentially. The reason why our advantage in the rotation metric was more obvious may be that the observed point cloud was sensitive to pose orientation. As the agent optimization progressed, the high coincidence of the Source and Target point clouds resulted in a more identical rotation.

In addition, the last three rows in Table 2 indicate that our reinforcement learning-based scheme possessed an overwhelming advantage over the traditional solver GN, even though the latter's configuration has unfair advantages, such as 60-fold initialization and simplified 3D space searching. This phenomenon fully shows that our agent is less susceptible to becoming stuck in local optimum.

Table 2. Registration accuracy evaluation on both Oxford and KITTI datasets. The average value and standard deviation of RTE and RRE are reported. The *Frus. Cls.* represents the result of "frustum classification" provided by DeepI2P [14]. The *GN. 6D* and *GN. 3D* indicate the Gauss–Newton solver with 6-DOF and 3-DOF, respectively. The ×60 corresponds to 60 times with randomly generated initialization. Ours refers to the provided BC + RL algorithm. The best results are shown in bold.

Method	Oxford		KITTI	
	RTE (m)	**RRE (°)**	**RTE (m)**	**RRE (°)**
Direct Regression	5.02 ± 2.89	10.45 ± 16.03	4.94 ± 2.87	21.98 ± 31.97
Grid. Cls. + PnP	1.91 ± 1.56	5.94 ± 10.72	**3.22** ± **3.58**	10.15 ± 13.74
MonoDepth2 + USIP	33.2 ± 46.1	142.5 ± 139.5	30.4 ± 42.9	140.6 ± 157.8
MonoDepth2 + ICP (×60)	7.94 ± 4.54	76.96 ± 76.38	9.80 ± 5.86	74.80 ± 73.93
2D3D-MatchNet (No Rot)	1.41	6.40	NA	NA
Frus. Cls. + GN. 6D (×60)	2.12 ± 1.72	14.10 ± 15.24	4.51 ± 3.86	20.89 ± 21.29
Frus. Cls. + GN. 3D (×60)	1.96 ± 1.56	4.51 ± 4.15	4.30 ± 3.76	9.50 ± 8.75
Frus. Cls. + (**ours**)	**1.34** ± **1.47**	**1.46** ± **6.65**	3.90 ± 3.79	**5.94** ± **14.06**

The distributions of the registration RTE (m) and RRE (°) on the Oxford and KITTI dataset are shown in Figures 4 and 5. Specifically, the the translational (m)/rotational (°) errors are ∼1.34 ± 1.47/1.46 ± 6.65 on Oxford and ∼3.90 ± 3.79/5.94 ± 14.06 on KITTI, where Oxford shows better performance. The relative worse performance on the KITTI dataset comes from the less accurate "frustum classification" result of DeepI2P [14] in the first stage. As explained in the original paper [14], the KITTI dataset is more difficult for classifying target point cloud, owing to a severe occlusion effect, limited vertical field-of-view, and small point clouds for training. In the ablation studies Section 4.5, we show that if provided a precise target point cloud, our agent is able to recover a proper cross-domain registration on both datasets.

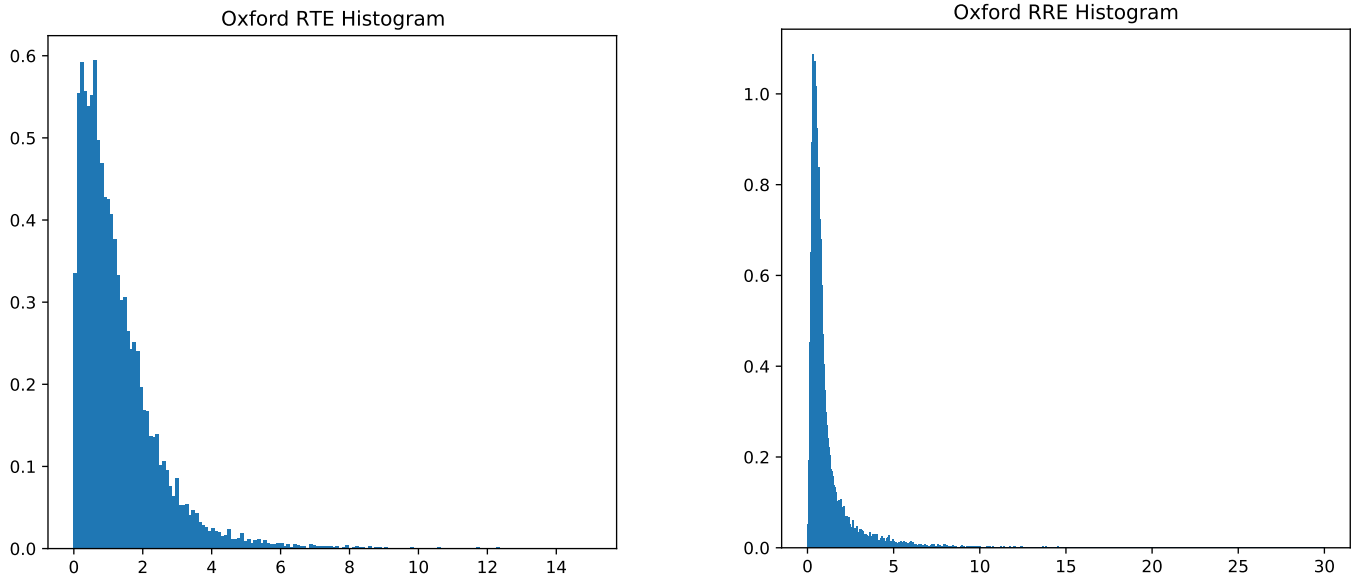

Figure 4. Histograms of image-point cloud registration RTE and RRE on the Oxford datasets. The x-axis is RTE (m) and RRE (°), and the y-axis is the percentage.

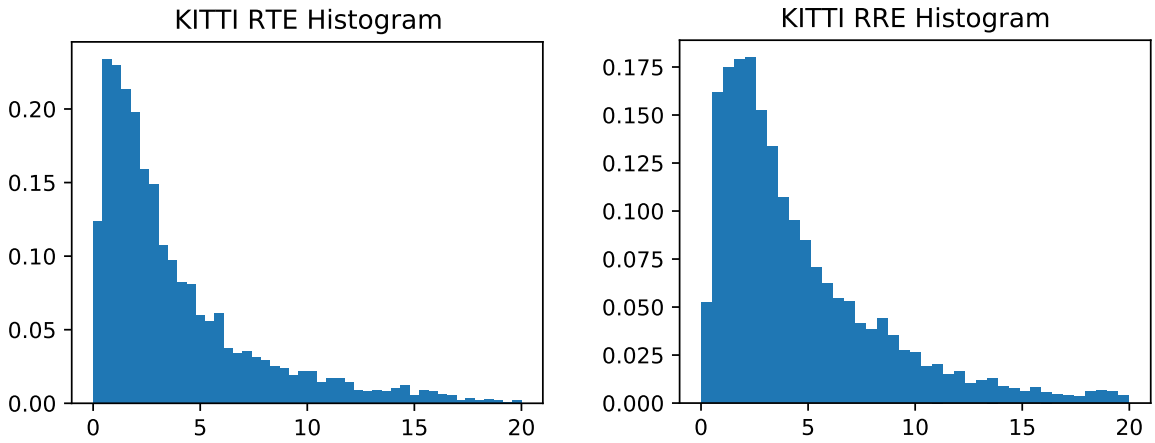

Figure 5. Histograms of image-point cloud registration RTE and RRE on the KITTI datasets. The x-axis is RTE (m) and RRE (°), and the y-axis is the percentage.

Efficiency. Although DeepI2P's specific setting contributes to higher accuracy, it also produces a lower efficiency and less versatility. In the following experiments, we discuss different Gauss–Newton (GN) solver scenarios by determining two variables, including [random] initial times and searching space [dimension]. More details are shown in Table 3. Note that to accelerate the inference duration of the multi-start solution, parallel computation with eight threads was adopted. However, no matter how GN solver balances performance between accuracy and efficiency, all of their evaluation indicators fall behind our BC + RL algorithm by a considerable margin.

Table 3. Registration efficiency evaluation on both Oxford and KITTI datasets. In addition to the average value and standard deviation of RTE and RRE, the mean inference time during camera pose optimization is reported. The ×230-like numbers in the time column show the multiplier of large speed-up. For the Gauss–Newton solver, different initial times and various optimization space are compared. The best results are shown in bold.

Method	Oxford			KITTI		
	RTE (m)	RRE (°)	Time (s)	RTE (m)	RRE (°)	Time (s)
GN. 6D (×60)	2.12 ± 1.72	14.10 ± 15.24	18.46 (×230)	4.51 ± 3.86	20.89 ± 21.29	20.13 (×335)
GN. 3D (×60)	1.96 ± 1.56	4.51 ± 4.15	10.82 (×135)	4.30 ± 3.76	9.50 ± 8.75	8.98 (×150)
GN. 6D (×1)	3.89 ± 2.78	15.04 ± 11.97	0.86 (×10)	5.26 ± 3.20	19.63 ± 13.94	1.25 (×20)
GN. 3D (×1)	3.61 ± 2.70	4.87 ± 4.50	0.44 (×5)	4.73 ± 3.14	10.64 ± 9.70	0.50 (×8)
ours	**1.34 ± 1.47**	**1.46 ± 6.65**	**0.08**	**3.90 ± 3.79**	**5.94 ± 14.06**	**0.06**

Specifically, considering the fairest case, we list a detailed comparison between our RL-based approach and traditional GN. 6D (×1), as they share the same initialization and 6-DoF searching space. In the Oxford dataset, AgentI2P outperforms Gauss–Newton by 190% at RTE, 954% at RRE and 975% at running time. In KITTI dataset, AgentI2P leads Gauss–Newton by 34% at RTE, 232% at RRE, and 1983% at running time. As we can see, the performance gap is quite notable, especially in RRE, and time-consuming.

4.5. Visualizations

A 3D visualization of our iterative optimization procedure is presented in Figure 6. It illuminates the intuition that a camera pose can be recovered by sequentially aligning the camera frustum to the classification result.

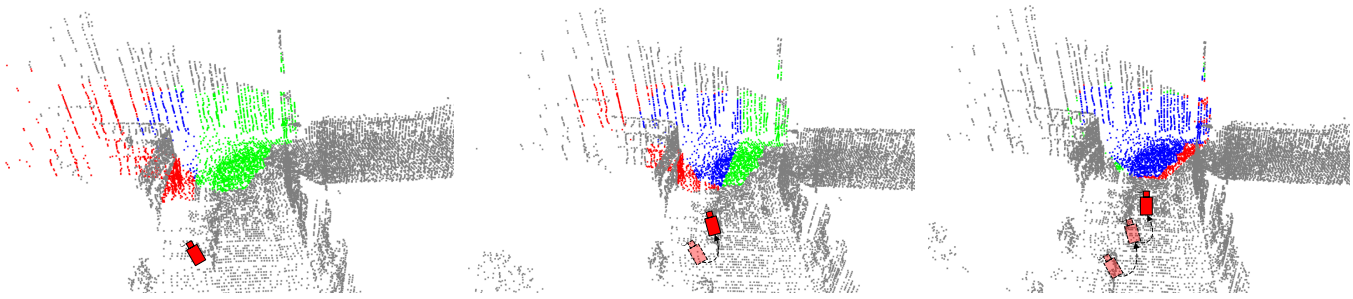

Figure 6. Visualizations of the AgentI2P at iteration 0/5/10 from left to right. Along with the continuous decision making of our agent, the proportion of overlapping points is increasing, and the camera is converging to the correct pose. Grey/green/red/blue indicate Map/Target/Source/overlaid point cloud respectively.

4.6. Ablation Studies

Ultimate ability. Recall that image-to-point cloud registration experiences a two-stage serial workflow, in which the latter "camera pose optimization" relies heavily on the prior "frustum point cloud classification". In this section, we carry out controlled experiments, assuming that the **True** or **Oracle** frustum classification is provided in the first stage. Such experimental setting not only contributes to directly exploiting the performance boost between AgentI2P and its Gauss–Newton counterpart, but also can explore the limited capability of AgentI2P.

The experimental results are shown in Table 4. In both datasets, our approach is significantly ahead of the GN solver with extremely small RTE and RRE, which is sufficient for localization of mobile devices in both indoor and outdoor environments. Furthermore, as depicted in last two columns of Table 4, we notice that the "camera pose optimization" problem has been successfully solved by AgentI2P, while a better frustum classification becomes a bottleneck. Although a better classification result is not our focus in this work, we will consider an appropriate solution in our future work.

Table 4. Ablation studies. Top performance with correctly provided frustum classification on both Oxford and KITTI datasets. The average value and standard deviation of RTE and RRE as well as the inference time are reported. The notation "Oracle" means that the frustum classification is substituted by ground-truth label in the first classification stage. In contrast, the notation "Frus. Cls." refers to deep network classification output of DeepI2P.

Method	Oxford			KITTI		
	RTE (m)	RRE (°)	Time (s)	RTE (m)	RRE (°)	Time (s)
GN. 6D (×60) (Oracle)	0.30 ± 0.22	2.23 ± 3.39	21.94 (×274)	0.31 ± 0.52	3.50 ± 3.29	22.79 (×325)
GN. 3D (×60) (Oracle)	0.28 ± 0.25	1.23 ± 1.33	10.64 (×133)	0.22 ± 0.44	1.81 ± 1.79	10.09 (×144)
GN. 6D (×1) (Oracle)	3.13 ± 3.21	11.34 ± 11.86	1.02 (×12)	3.59 ± 3.43	13.54 ± 8.53	1.25 (×18)
GN. 3D (×1) (Oracle)	2.70 ± 3.18	3.15 ± 4.14	0.44 (×5)	2.15 ± 2.83	5.72 ± 7.01	0.61 (×9)
ours (Frus. Cls.)	1.34 ± 1.47	1.46 ± 6.65	0.08	3.90 ± 3.79	5.94 ± 14.06	**0.06**
ours (Oracle)	**0.08 ± 0.33**	**0.48 ± 3.30**	**0.08**	**0.10 ± 0.26**	**1.06 ± 7.50**	0.07

Learning fashion. To fully understand the effect of different learning fashions, we evaluated three variants with results shown in Table 5, including: (1) BC: only directly supervised the agent action without RL-based reward feedback; (2) RL: only implicitly guided the agent based on greedily acquiring higher accumulation RL scores. (3) BC + RL: combined behaviour cloning and reinforcement learning in a unified manner.

Table 5. Ablation studies. Three variants of AgentI2P were trained and evaluated on both Oxford and KITTI datasets, including RL, BC ,and BC + RL. The average value and standard deviation of RTE and RRE are reported. The notation "Oracle" refers to the frustum classification being substituted by the ground-truth label in the first classification stage. The success rate refers to the percentage of predicted poses falling within the threshold of $(2\,\text{m}, 5^\circ)$.

Method	Oxford			KITTI		
	RTE (m)	RRE (°)	Success Rate (%)	RTE (m)	RRE (°)	Success Rate (%)
RL	5.10 ± 2.94	28.44 ± 12.60	0.00	6.04 ± 2.32	18.38 ± 14.71	0.00
BC	1.37 ± 1.43	1.47 ± 6.68	77.64	3.94 ± 3.86	6.19 ± 13.20	28.44
BC + RL	$\mathbf{1.34 \pm 1.47}$	$\mathbf{1.46 \pm 6.65}$	**78.46**	$\mathbf{3.90 \pm 3.79}$	$\mathbf{5.94 \pm 14.06}$	**30.53**
RL (Oracle)	5.03 ± 2.89	17.22 ± 9.44	0.00	6.05 ± 2.32	14.41 ± 14.95	0.00
BC (Oracle)	$\mathbf{0.07 \pm 0.29}$	0.50 ± 4.02	**99.74**	0.12 ± 0.40	1.13 ± 5.53	98.99
BC + RL (Oracle)	0.08 ± 0.33	$\mathbf{0.48 \pm 3.30}$	**99.74**	$\mathbf{0.10 \pm 0.26}$	$\mathbf{1.06 \pm 7.50}$	**99.16**

It is clear that the agent fails to converge with only RL loss, where 0.00% of estimated poses fall within the threshold of (2 m, 5°). This may be because the designed smooth reward function (i.e., Equation (10)) only focuses on step-wise improvement, and it probably biases the agent to find the solution with the most small improvements in MCD than the lowest MCD itself. Moreover, the BC + RL joint training leads to better results than simply employing BC alone, especially in noisy cases where the ground-truth (noted as Oracle in Table 5) target is not available. It proves that the proposed joint training strategy not only ensures the convergence to the global optimum, but also promotes the stability of the policy.

Robust to initialization. For a fair comparison, our work follows DeepI2P [14] to include three unknown parameters—rotation yaw (θ), translation t_x, t_y in the initialization course, taking the intrinsic properties of vehicle driving into account. The initial yaw (θ) is obtained by simply aligning the average angle of the predicted in-frustum points with the camera principle axis, while the other 2-degree search for position t_x, t_y are randomly spawned.

In this section, we examine to what extent our proposed agent is influenced by the initial pose selection. Particularly, we execute three groups of experiments with the seed position 1-to-2, 2-to-5, and 5-to-10 meters away from the true localization along the road. As shown in Table 6, our cross-domain registration agent is typically insensitive to the initialization, where the accuracy remains almost the same with increasing the seed error.

Table 6. Ablation studies. Three levels (i.e., 1–2 m, 2–5 m, 5–10 m) of initialization were adopted to test the robustness of our "BC + RL" agent. The average value and standard deviation of RTE and RRE are reported. The notation "Oracle" refers to the frustum classification being substituted by the ground-truth label in the first classification stage. The success rate refers to the percentage of predicted poses falling within the threshold of $(2\,\text{m}, 5^\circ)$.

Initial Distance	Oxford			KITTI		
	RTE (m)	RRE (°)	Success Rate (%)	RTE (m)	RRE (°)	Success Rate (%)
1–2 m	1.34 ± 1.51	1.49 ± 6.11	79.22	3.87 ± 3.78	5.94 ± 14.38	30.85
2–5 m	1.35 ± 1.50	1.57 ± 6.96	78.16	3.88 ± 3.81	6.00 ± 13.71	30.34
5–10 m	1.50 ± 2.21	2.51 ± 10.37	75.97	3.87 ± 3.76	6.05 ± 14.66	30.55
1–2 m (Oracle)	0.09 ± 0.45	0.71 ± 6.70	99.24	0.11 ± 0.28	1.18 ± 9.27	98.76
2–5 m (Oracle)	0.10 ± 0.53	0.82 ± 7.74	99.00	0.11 ± 0.46	1.13 ± 7.15	98.86
5–10 m (Oracle)	0.20 ± 1.11	2.04 ± 15.98	97.08	0.14 ± 0.70	1.25 ± 7.36	98.4

5. Discussions

In this paper, we propose a novel framework, AgentI2P, for image-to-point cloud registration via behaviour cloning and reinforcement learning. The experimental results show that the proposed agenti2p improves the previous localization by a large margin in terms of accuracy, robustness, and efficiency on widely used KITTI and Oxford datasets. We attribute the success to three aspects:

- Instead of adopting traditional optimization algorithms for point cloud alignment, we frame the pose optimization as an iterative Markov decision process and design a trainable agent to move the virtual camera sequentially until the camera-observed point cloud aligns with the network-classification point cloud by DeepI2P [14]. The prior classic methods are sensitive to initializations. Even if we randomly generate 60 initial positions, there is no guarantee that such algorithms could eliminate local optima. In contrast, our agent conducts 6-DoF action based on current observation in each iteration. Only provided a randomly initial position (whether noisy or not), our method can robustly optimize to a satisfying optima.
- We design a lightweight yet effective neural network for our agent to extract state representation and conduct decision making. This is because the pose optimization consists of a series of refinements, and we need to limit the time consumption during each iteration. The experimental results show that our method only requires 0.06 s for registration, which improves traditional optimization counterparts by $\times$18 times.
- We utilize behaviour cloning and reinforcement learning to train the model. Experiments demonstrate that the joint training contributes to a better convergence of AgentI2P compared to simply selecting one mode.

As illustrated in ablation studies, if provided with **True** or **Oracle** frustum classification in the first stage, our approach can output very accurate localization in both Oxford and KITTI datasets. As a result, image-to-point cloud classification becomes a bottleneck for cross-domain camera localization. Therefore, future research directions are highlighted as follows:

- The backbone of DeepI2P [14] may be replaced by more advanced embedding networks, such as KpConv [56], Randla-net [57], MinkowskiNet [58], BPNet [59] for scene point cloud, and DRAN [60], HANet [61], Lawin+ [62] for camera image.
- The feature fusion between point cloud and image in DeepI2P [14] may be replaced by Transformers [63–66]. As demonstrated in recent research, Transformers have brought about significant performance boosts for image-to-image matching [18,19] and point-to-point correspondence [67].

6. Conclusions

This paper presents a novel agent, named AgentI2P, that can achieve accurate, efficient, and robust registration across image and point cloud. The proposed AgentI2P uses lightweight PointNet-like architecture and leverages behaviour cloning and reinforcement learning to guide the deep policy network. The feasibility of our approach is evaluated in experiments on Oxford and KITTI datasets, and our approach achieves state of the art for all comparable methods. We believe that AgentI2P provides a new efficient strategy for autonomous vehicle self-localization in large 3D map under textureless or illumination-changing conditions and thus increases the safety of autonomous driving. In future work, we would like to improve the "frustum classification" result as it becomes the bottleneck for image-to-point cloud registration.

Author Contributions: Conceptualization, S.Y. and M.Z.; methodology, S.Y.; software, S.Y. and H.T.; validation, S.Y., Y.L. and M.Z.; formal analysis, Y.L.; investigation, S.Y.; resources, S.Y.; data curation, H.T.; writing—original draft preparation, S.Y.; writing—review and editing, Y.L.; visualization, Y.P.; supervision, M.Z.; project administration, Y.L.; funding acquisition, Y.L. and H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (62101576, 62171451, 61906206, 62071478), the Natural Science Foundation of Hunan Province of China (2020JJ5671), the Key Basic Research Project of the China Basic Strengthening Program (2019jcjqzd00600).

Data Availability Statement: The datasets generated during and/or analysed during the current study are available in the https://www.cvlibs.net/datasets/kitti/eval_odometry.php and https://robotcar-dataset.robots.ox.ac.uk/datasets/ repositories (accessed on 1 October 2022).

Acknowledgments: This work is partially supported by the National Natural Science Foundation of China (62101576, 62171451, 61906206, 62071478), the Natural Science Foundation of Hunan Province of China (2020JJ5671), the Key Basic Research Project of the China Basic Strengthening Program (2019jcjqzd00600).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Toft, C.; Maddern, W.; Torii, A.; Hammarstrand, L.; Stenborg, E.; Safari, D.; Okutomi, M.; Pollefeys, M.; Sivic, J.; Pajdla, T.; et al. Long-term visual localization revisited. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 2074–2088. [CrossRef]
2. Fan, B.; Yang, Y.; Feng, W.; Wu, F.; Lu, J.; Liu, H. Seeing through Darkness: Visual Localization at Night via Weakly Supervised Learning of Domain Invariant Features. *IEEE Trans. Multimed.* **2022**. [CrossRef]
3. Lim, H.; Sinha, S.N.; Cohen, M.F.; Uyttendaele, M.; Kim, H.J. Real-time monocular image-based 6-DoF localization. *Int. J. Robot. Res.* **2015**, *34*, 476–492. [CrossRef]
4. Gong, X.; Liu, Y.; Wu, Q.; Huang, J.; Zong, H.; Wang, J. An Accurate, Robust Visual Odometry and Detail-Preserving Reconstruction System. *IEEE Trans. Multimed.* **2020**, *23*, 2820–2832. [CrossRef]
5. Spera, E.; Furnari, A.; Battiato, S.; Farinella, G.M. EgoCart: A benchmark dataset for large-scale indoor image-based localization in retail stores. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *31*, 1253–1267. [CrossRef]
6. Heng, L.; Choi, B.; Cui, Z.; Geppert, M.; Hu, S.; Kuan, B.; Liu, P.; Nguyen, R.; Yeo, Y.C.; Geiger, A.; et al. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4695–4702.
7. Tian, X.; Shao, J.; Ouyang, D.; Shen, H.T. Uav-satellite view synthesis for cross-view geo-localization. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 4804–4815. [CrossRef]
8. Wang, T.; Zheng, Z.; Yan, C.; Zhang, J.; Sun, Y.; Zheng, B.; Yang, Y. Each part matters: Local patterns facilitate cross-view geo-localization. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 867–879. [CrossRef]
9. Mann, S.; Furness, T.; Yuan, Y.; Iorio, J.; Wang, Z. All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality. *arXiv* **2018**, arXiv:1804.08386.
10. Piao, J.C.; Kim, S.D. Real-time visual–inertial SLAM based on adaptive keyframe selection for mobile AR applications. *IEEE Trans. Multimed.* **2019**, *21*, 2827–2836. [CrossRef]
11. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
12. Sarlin, P.E.; Cadena, C.; Siegwart, R.; Dymczyk, M. From coarse to fine: Robust hierarchical localization at large scale. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12716–12725.
13. Schonberger, J.L.; Frahm, J.M. Structure-from-motion revisited. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4104–4113.
14. Li, J.; Lee, G.H. DeepI2P: Image-to-Point Cloud Registration via Deep Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 15960–15969.
15. Wedderburn, R.W. Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method. *Biometrika* **1974**, *61*, 439–447.
16. Torabi, F.; Warnell, G.; Stone, P. Behavioral cloning from observation. *arXiv* **2018**, arXiv:1805.01954.
17. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *arXiv* **2017**, arXiv:1708.05866.
18. Sarlin, P.E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4938–4947.
19. Sun, J.; Shen, Z.; Wang, Y.; Bao, H.; Zhou, X. LoFTR: Detector-free local feature matching with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8922–8931.

20. Fan, B.; Liu, H.; Zeng, H.; Zhang, J.; Liu, X.; Han, J. Deep unsupervised binary descriptor learning through locality consistency and self distinctiveness. *IEEE Trans. Multimed.* **2020**, *23*, 2770–2781. [CrossRef]
21. Sattler, T.; Leibe, B.; Kobbelt, L. Fast image-based localization using direct 2d-to-3d matching. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 667–674.
22. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
23. Balntas, V.; Li, S.; Prisacariu, V. Relocnet: Continuous metric learning relocalisation using neural nets. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 751–767.
24. Ding, M.; Wang, Z.; Sun, J.; Shi, J.; Luo, P. CamNet: Coarse-to-fine retrieval for camera re-localization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 2871–2880.
25. Kendall, A.; Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5974–5983.
26. Kendall, A.; Grimes, M.; Cipolla, R. Posenet: A convolutional network for real-time 6-dof camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, USA, 7–13 December 2015; pp. 2938–2946.
27. Laskar, Z.; Melekhov, I.; Kalia, S.; Kannala, J. Camera relocalization by computing pairwise relative poses using convolutional neural network. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 929–938.
28. Walch, F.; Hazirbas, C.; Leal-Taixe, L.; Sattler, T.; Hilsenbeck, S.; Cremers, D. Image-based localization using lstms for structured feature correlation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 627–637.
29. Zhou, Q.; Sattler, T.; Pollefeys, M.; Leal-Taixe, L. To learn or not to learn: Visual localization from essential matrices. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 3319–3326.
30. Brachmann, E.; Krull, A.; Nowozin, S.; Shotton, J.; Michel, F.; Gumhold, S.; Rother, C. Dsac-differentiable ransac for camera localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6684–6692.
31. Brachmann, E.; Rother, C. Learning less is more-6d camera localization via 3d surface regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4654–4662.
32. Brachmann, E.; Rother, C. Expert sample consensus applied to camera re-localization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 7525–7534.
33. Brachmann, E.; Rother, C. Visual camera re-localization from RGB and RGB-D images using DSAC. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5847–5865. [CrossRef]
34. Cavallari, T.; Golodetz, S.; Lord, N.A.; Valentin, J.; Di Stefano, L.; Torr, P.H. On-the-fly adaptation of regression forests for online camera relocalisation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4457–4466.
35. Li, X.; Wang, S.; Zhao, Y.; Verbeek, J.; Kannala, J. Hierarchical scene coordinate classification and regression for visual localization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11983–11992.
36. Shotton, J.; Glocker, B.; Zach, C.; Izadi, S.; Criminisi, A.; Fitzgibbon, A. Scene coordinate regression forests for camera relocalization in RGB-D images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2930–2937.
37. Yang, L.; Bai, Z.; Tang, C.; Li, H.; Furukawa, Y.; Tan, P. Sanet: Scene agnostic network for camera localization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 42–51.
38. Feng, M.; Hu, S.; Ang, M.H.; Lee, G.H. 2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4790–4796.
39. Wang, B.; Chen, C.; Cui, Z.; Qin, J.; Lu, C.X.; Yu, Z.; Zhao, P.; Dong, Z.; Zhu, F.; Trigoni, N.; et al. P2-Net: Joint Description and Detection of Local Features for Pixel and Point Matching. *arXiv* **2021**, arXiv:2103.01055.
40. Yu, H.; Zhen, W.; Yang, W.; Zhang, J.; Scherer, S. Monocular camera localization in prior lidar maps with 2d-3d line correspondences. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 4588–4594.
41. Cattaneo, D.; Vaghi, M.; Fontana, S.; Ballardini, A.L.; Sorrenti, D.G. Global visual localization in LiDAR-maps through shared 2D-3D embedding space. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 4365–4371.
42. Pham, Q.H.; Uy, M.A.; Hua, B.S.; Nguyen, D.T.; Roig, G.; Yeung, S.K. Lcd: Learned cross-domain descriptors for 2d-3d matching. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11856–11864.
43. Zhou, Y.; Sun, X.; Zha, Z.J.; Zeng, W. Context-reinforced semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4046–4055.

44. Mathe, S.; Pirinen, A.; Sminchisescu, C. Reinforcement learning for visual object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2894–2902.
45. Bauer, D.; Patten, T.; Vincze, M. ReAgent: Point Cloud Registration using Imitation and Reinforcement Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14586–14594.
46. Busam, B.; Jung, H.J.; Navab, N. I like to move it: 6d pose estimation as an action decision process. *arXiv* **2020**, arXiv:2009.12678.
47. Shao, J.; Jiang, Y.; Wang, G.; Li, Z.; Ji, X. Pfrl: Pose-free reinforcement learning for 6d pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11454–11463.
48. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
49. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
50. Bojarski, M.; Testa, D.D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J. End to End Learning for Self-Driving Cars. *arXiv* **2016**, arXiv:1604.07316.
51. Maddern, W.; Pascoe, G.; Linegar, C.; Newman, P. 1 year, 1000 km: The oxford robotcar dataset. *Int. J. Robot. Res.* **2017**, *36*, 3–15. [CrossRef]
52. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
53. Godard, C.; Mac Aodha, O.; Firman, M.; Brostow, G.J. Digging into self-supervised monocular depth estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3828–3838.
54. Besl, P.J.; McKay, N.D. Method for registration of 3-D shapes. In *Proceedings of the Sensor Fusion IV: Control Paradigms and Data Structures*; International Society for Optics and Photonics: Bellingham, WA, USA, 1992; Volume 1611, pp. 586–606.
55. Li, J.; Lee, G.H. Usip: Unsupervised stable interest point detection from 3d point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 361–370.
56. Thomas, H.; Qi, C.R.; Deschaud, J.E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. Kpconv: Flexible and deformable convolution for point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6411–6420.
57. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. Randla-net: Efficient semantic segmentation of large-scale point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11108–11117.
58. Choy, C.; Gwak, J.; Savarese, S. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3075–3084.
59. Hu, W.; Zhao, H.; Jiang, L.; Jia, J.; Wong, T.T. Bidirectional projection network for cross dimension scene understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14373–14382.
60. Fu, J.; Liu, J.; Jiang, J.; Li, Y.; Bao, Y.; Lu, H. Scene segmentation with dual relation-aware attention network. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 2547–2560. [CrossRef] [PubMed]
61. Choi, S.; Kim, J.T.; Choo, J. Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9373–9383.
62. Yan, H.; Zhang, C.; Wu, M. Lawin transformer: Improving semantic segmentation transformer with multi-scale representations via large window attention. *arXiv* **2022**, arXiv:2201.01615.
63. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
64. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 5156–5165.
65. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
66. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
67. Qin, Z.; Yu, H.; Wang, C.; Guo, Y.; Peng, Y.; Xu, K. Geometric transformer for fast and robust point cloud registration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 11143–11152.