drones

MDPI

Article

An Attention-Based Odometry Framework for Multisensory Unmanned Ground Vehicles (UGVs)

Zhiyao Xiao and Guobao Zhang *

School of Automation, Southeast University, No. 2, Sipailou, Nanjing 210018, China; 230198548@seu.edu.cn
* Correspondence: guobaozh@seu.edu.cn

Abstract: Recently, deep learning methods and multisensory fusion have been applied to address odometry challenges in unmanned ground vehicles (UGVs). In this paper, we propose an end-to-end visual-lidar-inertial odometry framework to enhance the accuracy of pose estimation. Grayscale images, 3D point clouds, and inertial data are used as inputs to overcome the limitations of a single sensor. Convolutional neural network (CNN) and recurrent neural network (RNN) are employed as encoders for different sensor modalities. In contrast to previous multisensory odometry methods, our framework introduces a novel attention-based fusion module that remaps feature vectors to adapt to various scenes. Evaluations on the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (KITTI) odometry benchmark demonstrate the effectiveness of our framework.

Keywords: UGV; odometry; deep learning; attention mechanism; KITTI dataset

Citation: Xiao, Z.; Zhang, G. An Attention-Based Odometry Framework for Multisensory Unmanned Ground Vehicles (UGVs). *Drones* **2023**, *7*, 699. https://doi.org/10.3390/drones7120699

Academic Editor: Giordano Teza

Received: 23 October 2023
Revised: 3 December 2023
Accepted: 7 December 2023
Published: 9 December 2023

1. Introduction

With the advancement of computer and sensor technology, UGVs find applications in various domains, including military operations and reconnaissance, security surveillance, search and rescue missions, logistics and delivery, and industrial automation. The odometry task, as a crucial component of the UGV framework, has been studied for decades but still presents challenges such as drift of pose estimation and the fusion of data from different sensors [1]. Our work primarily revolves around enhancing odometry estimation accuracy and optimizing multisensory fusion.

Traditional odometry methods typically rely on hand-crafted features to estimate relative pose and have achieved high accuracy [2]. However, these methods depend on human experience and lack adaptability in challenging scenarios. Recently, odometry methods based on deep neural networks have gained growing interest. Compared to traditional odometry methods, deep odometry methods offer several advantages: (1) High-dimensional features extracted by deep learning demonstrate robustness in the face of challenging scenarios; (2) Deep odometry methods do not require complex parameter calibration and geometric transformations; (3) Deep odometry methods can continuously learn and adapt to new scenes as their datasets expand.

In the face of complex environments, it is necessary for UGV to embrace multisensory fusion to enhance robustness [3]. These sensors include cameras, lidar, and inertial measurement units (IMUs). In recent years, numerous studies have been proposed to deal with the odometry task using these different sensors. Monocular cameras, being small-sized and cost-effective, provide rich pixel information. However, they heavily rely on sufficient visible light and struggle to extract range information. Stereo cameras can derive range information from two images but require precise calibration. Lidar captures abundant 3D structural information about the environment, but its cost depends on resolution and measurement distance, which can impact pose estimation accuracy. Additionally, incorrect 3D point associations between two frames can lead to pose estimation drift due to outliers or inappropriate algorithms. IMUs measure angular velocity and acceleration at

high frequencies, making them valuable for compensating for the inaccuracies of rapidly moving cameras or lidar. However, IMUs also suffer from cumulative errors. Recognizing the limitations of individual sensors in certain scenarios, multimodal sensor fusion has become a research hotspot. Filter-based methods, including the Kalman filter (KF) and extended Kalman filter (EKF), have traditionally been applied in sensor fusion frameworks [4]. However, linearization errors constrain the performance of filter-based methods. Meanwhile, many deep sensor fusion methods simply concatenate feature vectors encoded from different sensors without employing effective algorithms.

For the stated reasoning, we introduce an end-to-end visual-lidar-inertia odometry framework. Initially, we employ CNN to encode grayscale images and 3D point clouds, and RNN for inertial data. Notably, the 3D point clouds are projected into 2D images to align with CNN. Subsequently, the resulting feature vectors are concatenated and remapped to derive a 6-DoF relative pose. Finally, we train our framework in a supervised manner, utilizing the L2 norm. The main contributions of our work are as follows:

1. We utilize grayscale images, 3D point clouds, and inertial data as inputs to compensate for the weakness of each sensor, encoding these data by deep neural networks;
2. We introduce a novel fusion module that reorganizes and remaps feature vectors to adapt to challenging scenes by introducing an attention mechanism and auxiliary vectors;
3. We evaluate the performance of our framework on the KITTI odometry benchmark. A comparison with several variants demonstrates the effectiveness of our algorithm.

The rest of the paper is structured as follows: Section 2 introduces related work on the odometry task, categorized as traditional and deep works. Section 3 provides details about our algorithm. Section 4 presents the experimental results on the KITTI odometry benchmark. Finally, in Section 5, we conclude the paper and discuss our future plans.

2. Related Work

In this section, we provide an overview of traditional and deep methods applied to the odometry problem. Given our primary focus on multisensory UGVs, we emphatically discuss deep sensor fusion methods. As illustrated in Figure 1, odometry methods estimate the relative pose between two scenes using information from various sensors.

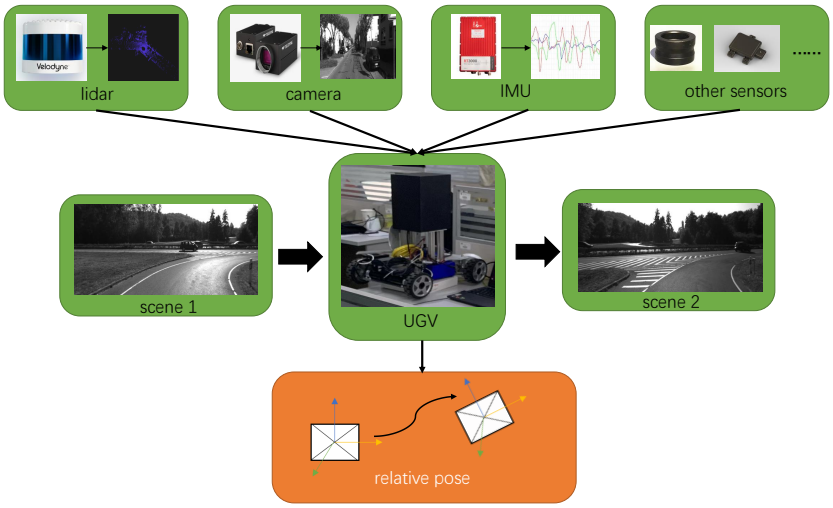


Figure 1. An overview of the odometry method.

2.1. Traditional Odometry

Behley et al. [5] employed a 2D vertex map and a normal map projected from 3D point cloud to estimate odometry efficiently. While suitable for mapping large outdoor scenes, their algorithm needs to exclude the influence of dynamic objects. Mur-Artal et al. [6] utilized oriented FAST and rotated BRIEF (ORB) feature from visual images to achieve

real-time operation. However, the extraction of ORB features resulted in sparse point clouds in the generated map, and the running speed was limited. Moreover, this approach was sensitive to dynamic objects, leading to tracking loss. In contrast to pre-computed features, Engel et al. [7] directly employed visual light intensity for motion tracking and can generate a map of dense point clouds. However, this approach had high requirements for scene illumination and demanded that the sensor maintain stable exposure. Zhen et al. [8] updated inertial and lidar measurements using a Gaussian particle filter. Subsequently, the outputs were fused using an error state Kalman filter to meet the real-time requirements of UAVs. Although it was efficient, this method did not consider the impact of the system state on the measurements, resulting in cumulative errors. Qin et al. [9] proposed a tightly-coupled visual-inertial system comprising a monocular camera and a low-cost IMU, which achieved highly accurate odometry by fusing pre-integrated IMU measurements and feature observations. As the method used the optical flow method to track and match feature points, the images need to be clear and continuous; otherwise, it will lead to the drift of the predicted trajectory. Shan et al. [10] introduced a tightly-coupled lidar-inertial odometry framework that incorporated a factor graph to effectively fuse lidar and IMU measurements. The factor graph framework they proposed demonstrated good compatibility. Graeter et al. [11] combined visual images with the depth information extracted from lidar measurements to estimate ego-motion. This method only used the point clouds in the visual field of the images to provide depth information, and the lidar was not fully utilized. Shan et al. [12] developed a tightly-coupled lidar-visual-inertial odometry framework based on a factor graph. This framework includes a visual-inertial system and a lidar-inertial system, mitigating the risk of single subsystem failure.

2.2. Deep Odometry

Li et al. [13] presented a deep convolutional network for odometry estimation. They projected 3D lidar point clouds into 2D images using cylindrical projection and then converted them into normal images to ensure geometric consistency. In contrast, Adis et al. [14] did not project 3D lidar point clouds but instead utilized mini-PointNet to directly handle point clouds, taking full advantage of point clouds. However, obtaining local point cloud features with PointNet in this method proved challenging, making the analysis of complex scenes difficult. Konda et al. [15] were the first to apply deep learning to visual odometry. They employed a simple CNN structure to predict direction and velocity from stereo images using the softmax function. But as an early attempt in this field, they had a poor accuracy of prediction. Zhou et al. [16] proposed an unsupervised framework for learning ego-motion and depth information from video sequences, with view synthesis serving as a supervisory signal. However, they did not explicitly incorporate dynamic objects and occlusion into the constraints, resulting in reduced robustness of the method in scenes that include both. Chen et al. [17] treated inertial odometry as a sequential learning problem and introduced a deep neural network based on long short-term memory (LSTM) structure to learn location transforms. However, the performance of their method is easily affected by the high bias value of IMU, and the generalization ability of the neural network is not strong.

The combination of multiple sensors can have a complementary effect. Clark et al. [18] introduced the first end-to-end trainable method for visual-inertial odometry. Their experiments demonstrated that the visual-inertial odometry method based on deep learning is more robust than conventional methods, particularly when there are errors in sensor calibration parameters. However, they simply concatenated features extracted from RGB images and inertial data without deploying a sensor fusion strategy. Han et al. [19] used a fully connected layer to fuse the outputs of RGB images and inertial data encoder, allowing for the update of additional bias for the IMU. Despite this, drift problems still exist in their method due to the absence of place recognition and relocalization pipelines. Son et al. [20] employed the softmask-based method to fuse lidar and inertial data, enabling the learning of the weights of each sensor. Their method relied on a multilayer perceptron (MLP), which may not be sufficient to handle challenging situations in the real world. Sun et al. [21]

combined soft mask attention fusion (SMAF) and a transformer for lidar-inertial odometry estimation task, addressing the overfitting problem associated with transformer networks. Moreover, it is able to fuse a mixture of heterogeneous sensor data, such as lidar and IMU. However, their fusion module suffers from high redundancy and is not beneficial for easy training and real-world applications. Aydemir et al. [22] fused the lidar and visual data using Bayesian inference to enhance depth maps, resulting in improved scale recovery. This method is a hybrid framework, and the sub-neural network needs to be trained separately, which can be cumbersome. Li et al. [23] adopted a siamese network architecture with input from both flipped depth and RGB images. They introduced a flip consistency loss to facilitate network learning, but it is susceptible to fall into local optima. Song et al. [24] proposed a feature pyramid network to fuse lidar point clouds and visual images, enhancing odometry accuracy across different levels. However, their method highly relied on photometric consistency; otherwise, errors are amplified through the pyramid structure.

3. Method

In this section, we describe the details of our odometry framework. Given two frames, P and Q, corresponding to time steps t and t + 1, our objective is to estimate the 6-DoF relative pose (translation on the x, y, and z axes, angle of roll, pitch, and yaw) between them. Our system comprises feature networks and a fusion network. The feature networks encode the features from visual images, 3D point clouds, and inertial data. The fusion network combines these feature vectors to calculate the transformation estimation between the two frames. The pipeline of our framework is illustrated in Figure 2.

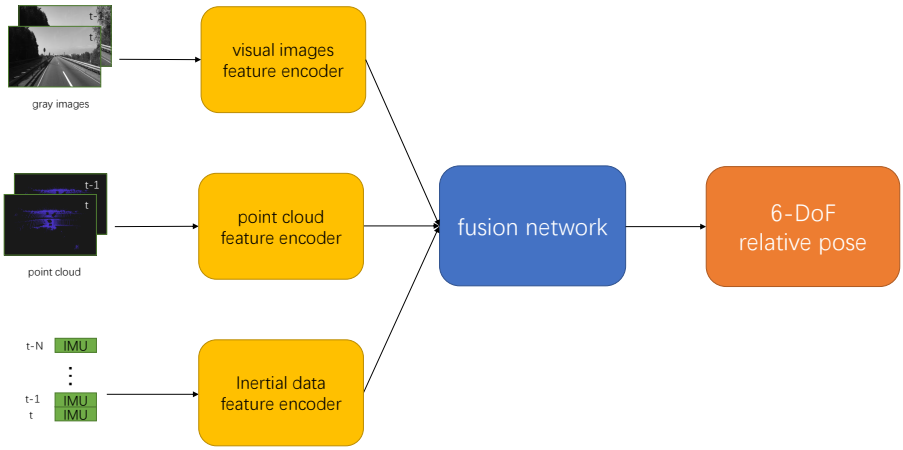


Figure 2. The pipeline of our odometry framework.

3.1. Data Encoding

3.1.1. Visual Images Feature Encoding

The inputs to the visual images feature encoder are two consecutive grayscale images. The encoder starts with two 5 * 5 convolutional layers, followed by two 3 * 3 convolutional layers. These layers are responsible for extracting features from the grayscale images. Each of the four convolutional layers is followed by a max-pooling layer and a rectified linear unit to reduce the size of images and introduce nonlinearity to the network. The remaining structure is a 1 * 1 convolutional layer, which is used to adjust the dimension of the feature vector. The convolutional layers are detailed in Table 1.

Table 1. Convolutional layers of visual images feature encoder.

Layer	Kernel Size	Stride	Padding	Channel Number
conv1	5 * 5	2 * 4	2	64
conv2	5 * 5	2 * 3	2	128
conv3	3 * 3	2 * 2	1	256
conv4	3 * 3	1 * 2	1	512
conv5	1 * 1	1 * 1	0	256

3.1.2. Point Cloud Feature Encoding

We project the 3D point clouds into a 2D image. Since a point can be represented as a 3D coordinate (x, y, z), its 2D coordinate (u, v) should be

$$u = H * \left(\frac{1}{2} - \frac{\varphi}{KVFOV} + \frac{bias\_KVFOV}{KVFOV}\right), \tag{1}$$

$$v = W * \left(\frac{1}{2} + \frac{\theta}{KHFOV} - \frac{bias\_KHFOV}{KHFOV}\right), \tag{2}$$

$$\varphi = \arctan\left(\frac{y}{x}\right), \tag{3}$$

$$\theta = \arcsin\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right), \tag{4}$$

where H is the height of the 2D image, W is the width of 2D image, φ is the azimuth angle, θ is the elevation angle, $KVFOV$ and $KHFOV$ are the vertical and horizontal view angle of the lidar sensor, and $bias\_KVFOV$ and $bias\_KHFOV$ are the mean of the top and bottom limitation of the view angle. The position (u, v) of the 2D image is filled with the range value $r = \sqrt{x^2 + y^2 + z^2}$. If several points have the same 2D coordinate, we keep the point with the smallest range value. The effect of converting a point cloud into a depth map is shown in Figure 3.

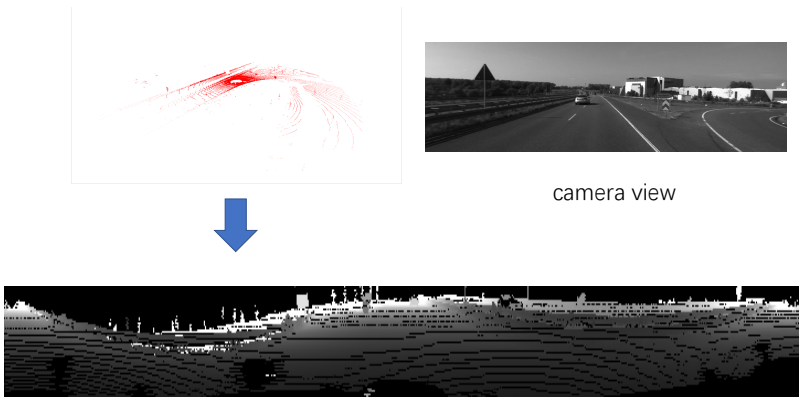


Figure 3. An example of point clouds projection.

The point cloud feature encoder has a similar structure to the visual images feature encoder. The inputs of the encoder are two range images projected by two consecutive frames of point clouds. Note that we abandon the max-pooling layer in this encoder since contrast experiments indicate that it decreases the estimation accuracy. Without max-pooling layers, we adjust the kernel size and stride to reduce the size of the images. The convolutional layers are detailed in Table 2.

Table 2. Convolutional layers of point cloud feature encoder.

Layer	Kernel Size	Stride	Padding	Channel Number
conv1	5 * 5	2 * 4	2	64
conv2	3 * 5	2 * 4	2	128
conv3	3 * 5	2 * 4	2	256
conv4	3 * 5	2 * 4	2	512
conv5	3 * 3	2 * 2	1	1024
conv6	1 * 1	1 * 1	0	256

3.1.3. Inertial Data Feature Encoding

Inertial data are typically available at higher frequency (100 Hz) compared to images (10 Hz). To align the inertial data with visual images and point clouds, we use spherical linear interpolation. We use quaternions to represent inertial data. Given two quaternions q_{t_1} and q_{t_2}, corresponding to time steps t_1 and t_2, the quaternion q_t, corresponding to time step t ($t \in (t_1 , \ t_2)$), is expressed as follows:

$$q_t = \frac{\sin\big((1-t_q)\theta\big)}{\sin(\theta)} q_{t_1} + \frac{\sin\big(t_q\theta\big)}{\sin(\theta)} q_{t_2}, \tag{5}$$

$$\theta = \arccos(q_{t_1} \cdot q_{t_2}), \tag{6}$$

$$t_q = \frac{t - t_1}{t_1 - t_2}, \tag{7}$$

where $\cdot$ denotes dot product.

The input of the inertial data feature encoder is composed of two consecutive inertial measurements. Due to the kinematic connection between inertial data and time, we choose RNN as the encoder. The RNN in our setup is a two-layer Bi-directional LSTM with 128 hidden states. The choice of Bi-directional LSTM is motivated by its ability to extract features from both preceding and succeeding inertial data.

3.2. Fusion Module

Features extracted from different sensors are complementary but not equally important in certain scenes. For instance, when a multitude of dynamic objects populate the scene, the contribution of visual and LiDAR features to pose estimation should be diminished. Inspired by the attention mechanism [25,26], we introduce a novel method for feature vector fusion. Rather than employing a weight function, we utilize a fully connected network to reorganize feature vectors, thereby achieving better fine-grained attention. So, we put feature vectors into the function:

$$f_{fuse}(a_V, \ a_P, \ a_I) = \mathrm{FC}(a_V \oplus a_P \oplus a_I), \tag{8}$$

where a_V is the visual images feature vector, a_P is the point cloud feature vector, a_I is the inertial data feature vector, FC is the fully connected network, and $\oplus$ denotes concatenating two feature vectors together.

Then we remap the feature vector for different targets to further improve the pose estimation accuracy. Given that the output is a 6-dimensional vector, the reorganized feature vector is concatenated with six independent vectors respectively. Each of independent vectors is deterministically parameterized by the network and updated using gradient descent. Thus, these recombined vectors are passed through six separate fully connected networks to obtain the final output. The remapping function can be described as follows:

$$f_{remap}\Big(a_{fuse}\Big) = \mathrm{FC}\Big(a_{fuse} \oplus v_i\Big), \tag{9}$$

where a_{fuse} is the reorganized feature vector calculated by function 8 and v_i is the independent vector ($i = 1, 2, 3, 4, 5, 6$). The detail of the fusion module is shown in Figure 4.

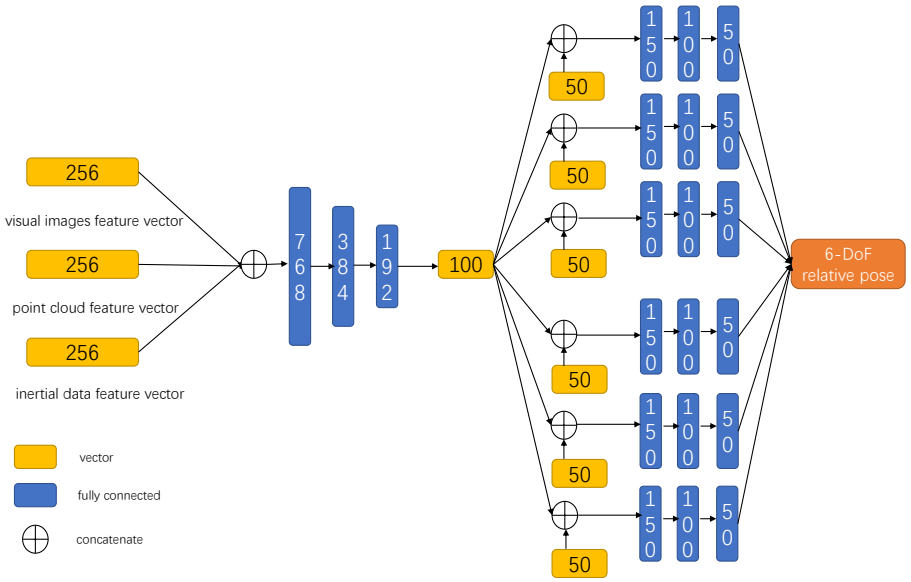


Figure 4. An example of point clouds projection. The number of fully connected layers' input channels is marked on the block. The length of vector is marked on the yellow block.

3.3. Loss Function

The output of our network is a 6-DoF relative pose v:

$$v = [p, r], \tag{10}$$

where p is a 3D translation vector and r is a 3D Euler rotation vector. Our loss function of the network is defined as follows:

$$L = \frac{1}{\lambda + 1} * ||p - \hat{p}||_2 + \frac{\lambda}{\lambda + 1} * ||r - \hat{r}||_2, \tag{11}$$

where $\hat{p}$ is ground truth of translation vector, $\hat{r}$ is ground truth of Euler rotation vector, $|| * ||_2$ means L2 norm, and λ is a scale factor to balance the error of translation and rotation. In our experiment, λ is chosen as 100.

4. Experiment

In this section, we introduce the details of training and experiment. We compare our method with several representative works on the odometry task. We also compare our method with its variants.

4.1. Dataset and Evaluation Metrics

We choose the KITTI odometry benchmark for our experiment, which comprises 22 sequences containing grayscale images, color images, and Velodyne laser data. Among these sequences, 10 (00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10) provide ground truth pose data. Additionally, corresponding inertial data are available in the KITTI raw data for all sequences except Sequence 03. For our training, we utilized Sequence 00, 01, 02, 04, 05, 06, 07, 08, and for testing, we used Sequence 09 and 10. Table 3 shows the details of the KITTI odometry benchmark.

Table 3. Convolutional layers of Visual images feature encoder.

Sequence	Distance (m)	Frames	Average Speed (km/h)	Max Speed (km/h)	Environment
00	3724	4541	29.5	46	Urban
01	2453	1101	80.2	96	Highway
02	5067	4661	39.1	49	Urban
03	561	801	25.2	31	Country
04	394	271	52.3	56	Country
05	2206	2761	28.8	40	Country
06	1233	1101	40.3	51	Urban
07	695	1101	22.7	39	Urban
08	3223	4071	28.5	43	Urban
09	1705	1591	38.6	52	Urban
10	920	1201	27.6	51	Country

To evaluate the pose estimation results, we follow the official criteria outlined in the KITTI benchmark [27]. This metric is calculated by averaging the root mean square errors of the translation and rotation for the involved sequences of lengths (100, 200, 300, 400, 500, 600, 700, 800) meters. Formally, the error metrics are defined as follows:

$$t_{rel} = \frac{1}{|F|} \sum_{(i,j) \in F} \frac{\text{Trans}\big((\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\big)}{\text{length}(i,j)}, \tag{12}$$

$$r_{rel} = \frac{1}{|F|} \sum_{(i,j) \in F} \frac{\text{Rot}\big((\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\big)}{\text{length}(i,j)}, \tag{13}$$

where F is the set of frames (i, j) , i and j represent the subscript of two frames separated by a fixed distance, length(i, j) represents the fixed distance, Trans() and Rot() represent figuring out the rotation and translation respectively, $\ominus$ denotes calculating the relative pose transformation between two frames, and $\hat{p}$ and p represent the estimated and true values of pose. The evaluation metrics of the KITTI dataset incorporates the length of the trajectory into the metric, which gives the final error a clear physical meaning.

4.2. Baselines

Several traditional and deep odometry works are chosen as baselines, including DeepVO [28], DeepLO [29], Au et al. [30], Chen et al. [31], EMA-VIO [32], DeepLIO [33] and LOAM [34]. These deep learning methods share similar encoder structures with our method, comprising simple CNN and RNN. DeepVO [28] proposed a deep recurrent convolutional neural network to estimate pose changes from a sequence of RGB images. DeepLO [29] projected point clouds into a vector map and a normal map, extracting features to train for relative pose. Au et al. [30] employed lidar data projection onto the image plane to generate depth images. They used three depth images from different angles along with one RGB image as input for their CNN. Chen et al. [31] proposed a selective sensor fusion module, akin to attention mechanisms, to weight different features. EMA-VIO [32] introduced an external memory mechanism based on a transformer when fusing visual and inertial data, showing good performance in accuracy and robustness. DeepLIO [33] used the soft mask-based method, which relied on MLP to learn the weights of lidar and inertial data. LOAM [34] extracted edge points and planar points from two scans of lidar and figured out the ego-motion using the Levenberg–Marquardt method.

DeepVO [28] used Sequence 00, 02, 08, 09 for training and Sequence 03, 04, 05, 06, 07, 10 for testing. DeepLO [29] and Au et al. [30] used Sequence 00, 01, 02, 03, 04, 05, 06, 07, 08 for training and Sequence 09, 10 for testing. Chen et al. [31] used Sequence 00, 01, 02, 03, 04, 06, 08, 09 for training and Sequence 05, 07, 10 for testing. EMA-VIO [32] and DeepLIO [33]

used Sequence 00, 01, 02, 04, 05, 06, 07, 08 for training and Sequence 09, 10 for testing. DeepVO [28], Chen et al. [31], EMA-VIO [32], and DeepLIO [33] trained in a supervised manner. DeepLO [29] and Au et al. [30] trained in an unsupervised manner. LOAM [34] is a traditional work that has good performance on the KITTI odometry benchmark.

4.3. Implementation Details

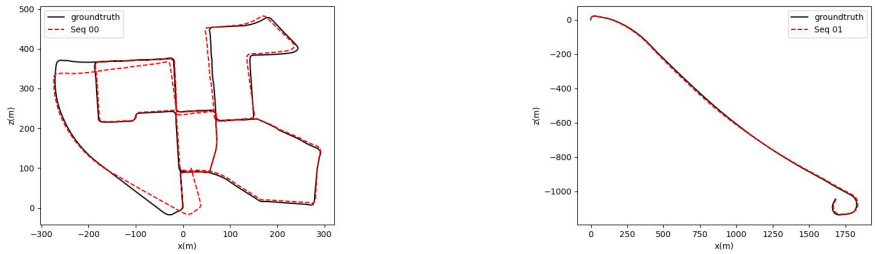
We implement our method based on the PyTorch framework and train our network using a single NVIDIA RTX 2080Ti. Adam optimizer is used in our training with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is 10-4 and controlled by a step scheduler with a patience of 5 and factor of 0.5. Batch size was set to 5. Our model is trained for 500 epochs. Note that we do not use normalization layers in our architecture since our experiments show that adding these layers decreases the accuracy of prediction.

4.4. Result of Comparison

Figure 5 shows the result of our training on the training set. We can see that the motion trails predicted by our method fit the ground truth. As shown in Table 4, we conducted a comparative analysis of our method against baselines on the odometry task. Our method exhibits superior performance compared to most of baselines, especially when compared to DeepLO [29] and Au et al. [30] on the training set (Sequence 00, 01, 02, 04, 05, 06, 07, 08). Notably, sequence 01 includes more dynamic objects than other sequences since it was recorded on a busy highway. Consequently, DeepLO [29], DeepLIO [33] and LOAM [34] display higher translation errors in Sequence 01 due to interference from dynamic objects. In contrast, our method demonstrates remarkable robustness in Sequence 01, attributed to the adoption of multisensory inputs, particularly inertial data, proving to be immune to dynamic objects' interference. On the testing sets, our method outperforms both DeepVO [28] and DeepLO [29], both of which rely on a single sensor. While our method excels in translational accuracy compared to multisensory methods, it does show higher rotational errors in Sequence 10. It is worth noting that both Au et al. [30] and Chen et al. [31] utilized RGB images as inputs, which contain more features but require more memory during training. Additionally, some inertial data are missing for certain timesteps in the KITTI raw data, resulting in errors during normalized interpolation of inertial data. DeepLIO [33] performs better than other methods in rotational accuracy, proving that the combination of point clouds and inertial data can extract more accurate information to predict rotation. Although there is still a performance gap between our method and LOAM [34], it is important to highlight that our method does not require precise calibration matrices.

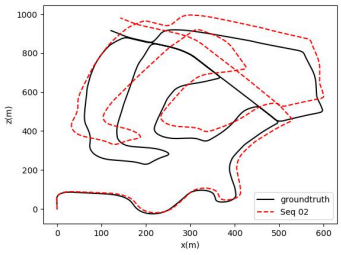
Table 4. Comparison with other works. t_{rel} is the translational error (%) and r_{rel} is the rotational error (deg/100 m). V denotes taking visual images as input. L denotes taking 3D point clouds as input. I denotes taking inertial data as input. V+L denotes taking both visual images and 3D point clouds as input. V+I denotes taking both visual images and inertial data as input. L+I denotes taking both 3D point clouds and inertial data as input. V+L+I denotes taking visual images, 3D point clouds, and inertial data as input. The best model's number will be bolded in the table.

Seq.	Our Method		DeepVO [28]		DeepLO [29]		Au et al. [30]		Chen et al. [31]		Ema-Vio [32]		DeepLIO [33]		LOAM [34]	
	V+L+I		V		L		V+L		V+L		V+I		L+I		L	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	1.02	0.46	-	-	1.90	0.80	2.77	1.78	-	-	-	-	1.6	**0.38**	**0.78**	-
01	**0.55**	0.24	-	-	37.83	0.86	3.76	0.80	-	-	-	-	5.9	**0.19**	1.43	-
02	1.76	0.95	-	-	2.05	0.81	4.82	2.26	-	-	-	-	1.96	**0.23**	**0.92**	-
03	-	-	8.49	6.89	2.85	1.43	2.75	**1.39**	-	-	-	-	-	-	**0.86**	-
04	1.05	0.71	7.19	6.97	1.54	0.87	1.81	1.48	-	-	-	-	3.7	**0.12**	**0.71**	-
05	1.93	1.05	2.62	3.61	1.72	0.92	3.81	1.43	4.25	1.67	-	-	1.24	**0.21**	**0.57**	-
06	0.89	0.32	5.42	5.82	0.84	0.47	4.03	1.22	-	-	-	-	1.97	**0.14**	**0.65**	-
07	0.77	0.42	3.91	4.60	0.70	0.67	3.61	1.41	4.46	2.17	-	-	1.92	**0.32**	**0.63**	-
08	**0.99**	0.39	-	-	1.81	1.02	2.75	1.61	-	-	-	-	2.34	**0.34**	1.12	-
09	3.32	1.58	-	-	6.55	2.19	3.76	1.92	-	-	8.68	1.54	4.4	**0.21**	**0.77**	-
10	4.28	2.24	8.11	8.83	7.74	2.84	4.65	0.51	5.81	1.55	7.46	2.26	4.0	**0.51**	**0.79**	-

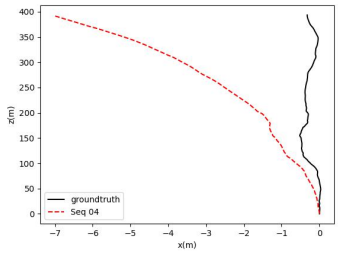


(**a**) Sequence 00

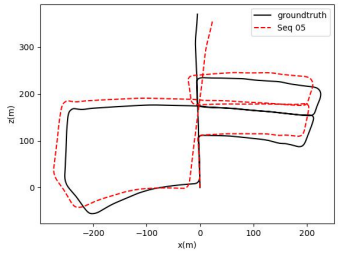
(**b**) Sequence 01



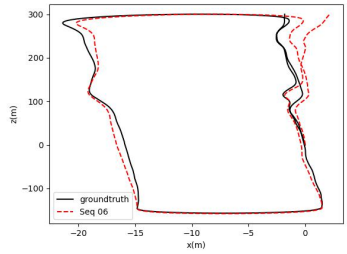
(**c**) Sequence 02



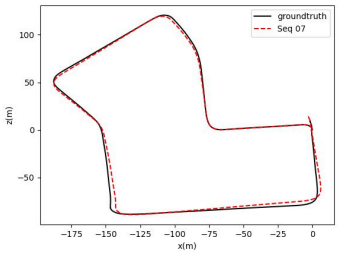
(**d**) Sequence 04



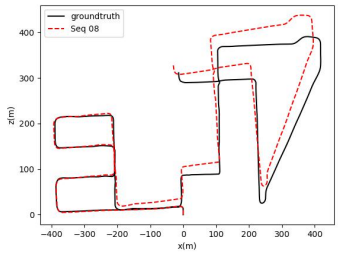
(**e**) Sequence 05



(**f**) Sequence 06



(**g**) Sequence 07



(**h**) Sequence 08

Figure 5. Motion trails of training set.

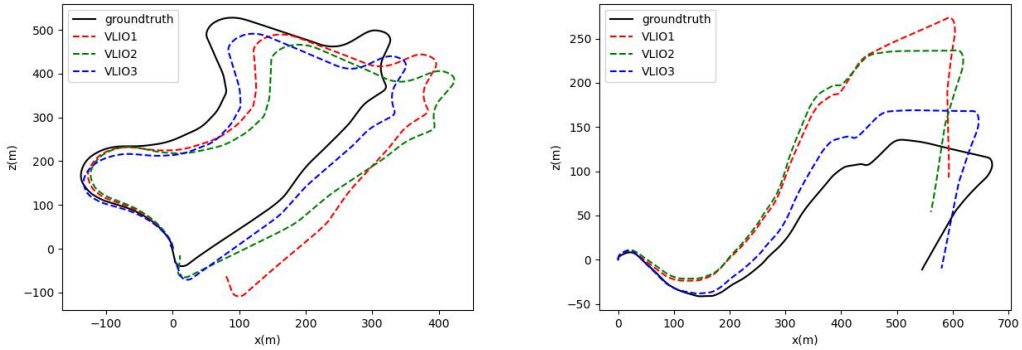
4.5. Ablation Studies

We conduct several experiments to investigate the effectiveness of our feature fusion network by generating variants of our framework. For this purpose, we test the following variants: (1) VLIO1: this framework concatenates feature vectors directly; (2) VLIO2: this framework uses the function 8 to fuse feature vectors but do not concatenate the reorganized feature vector with six independent vectors respectively; (3) VLIO3: the full framework. To prove the necessity of feature fusion, we also test three other variants: (1) VLO: this framework takes grayscale images and 3D point clouds as input; (2) VIO: this framework takes grayscale images and inertial data as input; (3) LIO: this framework takes 3D point clouds and inertial data as input. The above three frameworks use full fusion network.

Table 5 shows the relative pose estimations. When comparing VLIO with VLO, VIO, and LIO, we observe that the fusion of different sensors enhances the estimation performance. When compared to VLIO1, the performance of VLIO2 validates the effectiveness of our fusion network. Similarly, when compared to VLIO2, the performance of VLIO3 validates the effectiveness of the remapping. Figure 6 shows the motion trails of Sequence 09 and Sequence 10 in the x-z plane. It is apparent that the motion trail generated by VLIO3 closely aligns with the ground truth in comparison to the trails of other variants. Additionally, Figure 7 presents the translational and rotational errors of Sequence 09 and Sequence 10, averaged over sub-sequences with a length of (100, 200, . . . , 800) meters and different speeds of measurement platform. We can see more clearly that our method's performance is superior to other variants. In addition, our method has a flatter curve and is insensitive to the change of speed, indicating that our proposed fusion module effectively enhances the robustness of the system.

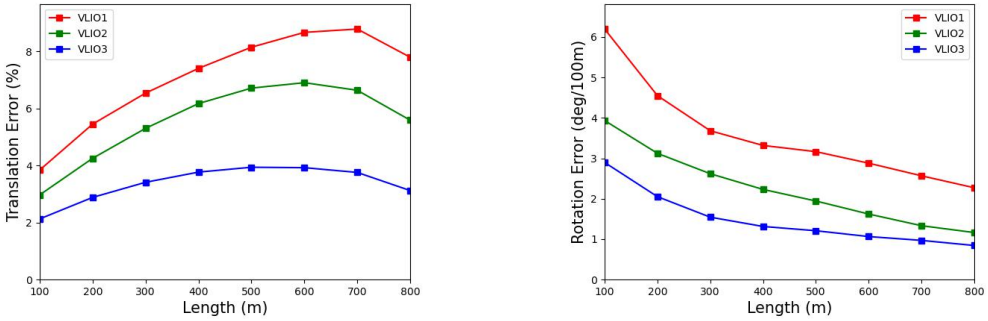
Table 5. Comparison of our framework and its variants. t_{rel} is the translational error (%) and r_{rel} is the rotational error (deg/100 m).

Models	Seq. 09		Seq. 10		Mean	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
VLIO1	6.86	3.76	6.85	3.87	6.86	3.82
VLIO2	5.42	2.39	6.12	3.75	5.77	3.07
VLIO3	3.32	1.58	4.28	2.24	3.80	1.91
VLO	8.02	4.39	7.85	4.40	7.94	4.40
VIO	9.73	5.11	7.76	5.04	8.75	5.08
LIO	9.10	4.92	7.84	3.94	8.47	4.43



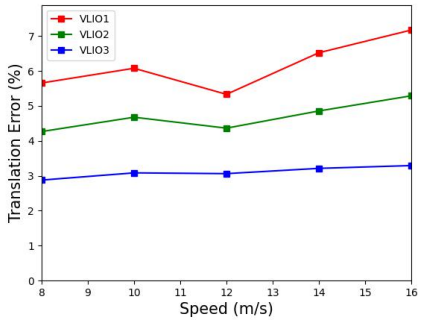
(**a**) Sequence 09 (**b**) Sequence 10

Figure 6. Motion trails of Sequence 09 and 10.

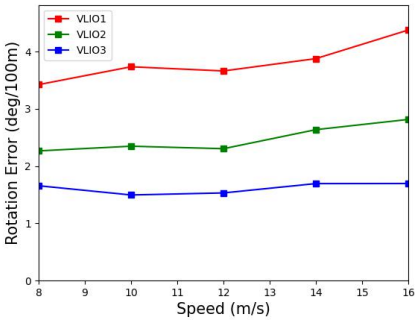


(**a**) Translational error of Sequence 09 (length) (**b**) Rotational error of Sequence 09 (length)

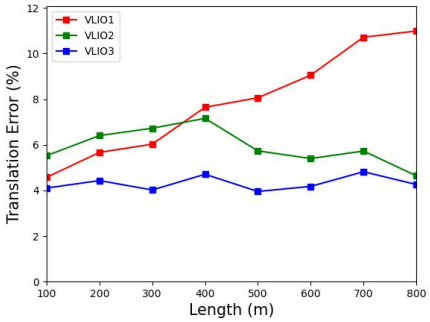
Figure 7. *Cont.*



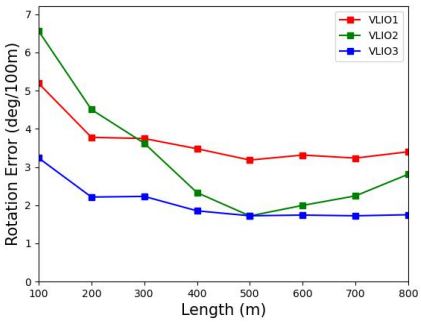
(**c**) Translational error of Sequence 09 (speed)



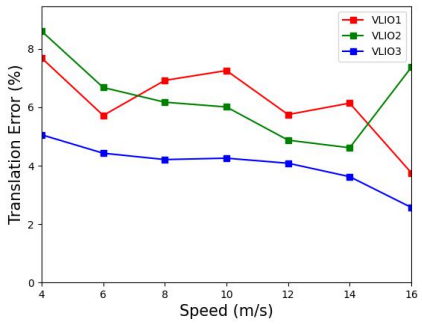
(**d**) Rotational error of Sequence 09 (speed)



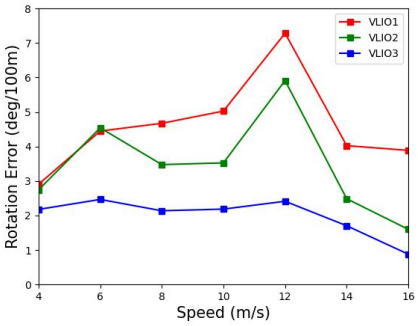
(**e**) Translational error of Sequence 10 (length)



(**f**) Rotational error of Sequence 10 (length)



(**g**) Translational error of Sequence 10 (speed)



(**h**) Rotational error of Sequence 10 (speed)

Figure 7. Main relationships between persuasive devices, consumer factors, and types of motivations in car sharing.

5. Conclusions

In this paper, we present a novel attention-based odometry framework for multi-sensory UGV. Our method fully leverages the complementary properties of monocular cameras, lidar, and IMU by taking grayscale images, 3D point clouds, and inertial data as inputs. To enhance the robustness and prediction accuracy of the system, we employ neural networks as encoders and carefully tune the network parameters to optimize overall performance. CNN excels at processing image information, enabling us to extract feature vectors from grayscale and depth images. Notably, the depth images are generated from 3D point clouds, simplifying point cloud data processing for neural networks. We utilize RNN to extract feature vectors from inertial data, modeling the temporal correlation within consecutive inertial data sequences. After extracting these three feature vectors, we introduce a novel feature fusion method inspired by the attention mechanism. Initially, we employ a fully connected network to adjust the weights of the three feature vectors, enhancing the granularity of the resulting features. Subsequently, we remap the reorga-

nized feature vector into six different outputs, thereby achieving greater accuracy. For our experiments, we select the KITTI dataset, which contains data from various sensors and diverse environmental conditions. The results of our experiments demonstrate that our method outperforms other deep learning methods with similar encoder structures. This comparison emphasizes that multisensory odometry offers higher accuracy and robustness in challenging scenarios compared to single-sensor odometry. Ablation studies further confirm that our proposed feature fusion module significantly enhances the performance of deep learning methods. Nevertheless, we have employed a simple CNN structure as an encoder and fine-tuned neural network parameters.

In this paper, our method has been exclusively tested on the KITTI dataset, encompassing urban, highway, and country scenes. However, it does not cover diverse environments like indoor or wooded areas. To adapt our method for alternative datasets, it is crucial to synchronize timestamps across different sensors. Additionally, each sensor's data undergo preprocessing to attain a format suitable for CNN and RNN processing.

Despite the promising results, our method faces several challenges. The neural network structure proposed exhibits limited generalization ability, performing less optimally on testing sets compared to training sets. To address this, we plan to explore more advanced encoders and broaden training to encompass diverse datasets, thereby enhancing generalization capabilities. Furthermore, we observed an accumulation of errors over time, resulting in a significant final error between the predicted trails and the ground truth. Introducing closed-loop optimization is imperative to continuously correct deviations. Lastly, our method currently relies on ground truth, limiting its applicability in scenarios where ground truth is unavailable. Consequently, we aim to incorporate unsupervised learning to mitigate dependence on ground truth.

Author Contributions: Z.X. is the primary contributor to this manuscript, including innovation, visualization, writing, and code editing. G.Z. is the corresponding author and proofreader of the paper, providing funding support. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Jiangsu Provincial Social Developing Project under Grant BE2020116 and BE2021750.

Data Availability Statement: The datasets we used in this paper are available for download at any time: KITTI odometry benchmark: https://www.cvlibs.net/datasets/kitti/eval_odometry.php (accessed on 20 October 2023). KITTI raw data: https://www.cvlibs.net/datasets/kitti/raw_data.php (accessed on 20 October 2023).

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

Abbreviations

The following abbreviations are used in this manuscript:

UGV	unmanned ground vehicle
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago
ORB	oriented FAST and rotated BRIEF
MLP	multilayer perceptron
SMAF	soft mask attention fusion
CNN	convolutional neural network
RNN	recurrent neural network
IMU	inertial measurement unit
KF	Kalman filter
EKF	extended Kalman filter
LSTM	long short-term memory

References

1. Xu, X.; Zhang, L.; Yang, J.; Cao, C.; Wang, W.; Ran, Y.; Tan, Z.; Luo, M. A Review of Multi-Sensor Fusion SLAM Systems Based on 3D LIDAR. *Remote Sens.* **2022**, *14*, 2835. [CrossRef]

2. Chen, C.; Wang, B.; Lu, C.X.; Trigoni, N.; Markham, A. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arXiv* **2020**, arXiv:2006.12567.
3. Debeunne, C.; Vivet, D. A review of visual-LiDAR fusion based simultaneous localization and mapping. *Sensors* **2020**, *20*, 2068. [CrossRef] [PubMed]
4. Khodarahmi, M.; Maihami, V. A review on Kalman filter models. *Arch. Comput. Methods Eng.* **2023**, *30*, 727–747. [CrossRef]
5. Behley, J.; Stachniss, C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In Proceedings of the 14th Conference on Robotics—Science and Systems, Pittsburgh, PA, USA, 26–30 June 2018; p. 59.
6. 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]
7. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625. [CrossRef] [PubMed]
8. Zhen, W.; Zeng, S.; Soberer, S. Robust localization and localizability estimation with a rotating laser scanner. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 6240–6245.
9. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
10. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24–30 October 2020; pp. 5135–5142.
11. Graeter, J.; Wilczynski, A.; Lauer, M. LIMO: Lidar-Monocular Visual Odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7872–7879.
12. Shan, T.; Englot, B.; Ratti, C.; Rus, D. LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 5692–5698.
13. Li, Q.; Chen, S.; Wang, C.; Li, X.; Wen, C.; Cheng, M.; Li, J. LO-Net: Deep Real-time Lidar Odometry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8473–8482.
14. Adis, P.; Horst, N.; Wien, M. D3DLO: Deep 3D LiDAR Odometry. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2021; pp. 3128–3132.
15. Konda, K.R.; Memisevic, R. Learning visual odometry with a convolutional network. *Learn. Vis. Odometry Convolutional Netw.* **2015**, *2015*, 486–490.
16. Zhou, T.; Brown, M.; Snavely, N.; Lowe, D.G. Unsupervised Learning of Depth and Ego-Motion from Video. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1851–1858.
17. Chen, C.; Lu, X.; Markham, A.; Trigoni, N. IONet: Learning to Cure the Curse of Drift in Inertial Odometry. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
18. Clark, R.; Wang, S.; Wen, H.; Markham, A.; Trigoni, N. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
19. Han, L.; Lin, Y.; Du, G.; Lian, S. DeepVIO: Self-supervised Deep Learning of Monocular Visual Inertial Odometry using 3D Geometric Constraints. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 6906–6913.
20. Son, H.; Lee, B.; Sung, S. Synthetic Deep Neural Network Design for Lidar-inertial Odometry Based on CNN and LSTM. *Int. J. Control. Autom. Syst.* **2021**, *19*, 2859–2868. [CrossRef]
21. Sun, L.; Ding, G.; Qiu, Y.; Yoshiyasu, Y.; Kanehiro, F. TransFusionOdom: Transformer-based LiDAR-Inertial Fusion Odometry Estimation. *IEEE Sens. J.* **2023**, *23*, 22064–22079. [CrossRef]
22. Aydemir, E.; Fetic, N.; Unel, M. H-VLO: Hybrid LiDAR-Camera Fusion for Self-Supervised Odometry. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 3302–3307.
23. Li, B.; Hu, M.; Wang, S.; Wang, L.; Gong, X. Self-supervised Visual-LiDAR Odometry with Flip Consistency. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2021; pp. 3844–3852.
24. Song, Z.; Lu, J.; Yao, Y.; Zhang, J. Self-Supervised Depth Completion from Direct Visual-LiDAR Odometry in Autonomous Driving. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 11654–11665. [CrossRef]
25. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
26. Hori, C.; Hori, T.; Lee, T.Y.; Zhang, Z.; Harsham, B.; Hershey, J.R.; Marks, T.K.; Sumi, K. Attention-based multimodal fusion for video description. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 4193–4202.
27. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
28. Wang, S.; Clark, R.; Wen, H.; Trigoni, N. DeepVO: Towards End to End Visual Odometry with Deep Recurrent Convolutional Neural Networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 2043–2050.
29. Cho, Y.; Kim, G.; Kim, A. DeepLO: Geometry Aware Deep LiDAR Odometry. *arXiv* **2019**, arXiv:1902.10562.

30. An, Y.; Shi, J.; Gu, D.; Liu, Q. Visual LiDAR SLAM Based on Unsupervised Multi channel Deep Neural Networks. *Cogn. Comput.* **2022**, *14*, 1496–1508. [CrossRef]
31. Chen, C.; Rosa, S.; Lu, C.X.; Wang, B.; Trigoni, N.; Markham, A. Learning Selective Sensor Fusion for States Estimation. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *10*, 1–15. [CrossRef] [PubMed]
32. Tu, Z.; Chen, C.; Pan, X.; Liu, R.; Cui, J.; Mao, J. Ema vio: Deep visual inertial odometry with external memory attention. *IEEE Sensors J.* **2022**, *22*, 20877–20885. [CrossRef]
33. Iwaszczuk, D.; Roth, S. DeepLIO: Deep LIDAR inertial sensor fusion for odometry estimation. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2021**, *1*, 47–54.
34. Zhang, J.; Singh, S. Low drift and real time lidar odometry and mapping. *Auton. Robot.* **2017**, *41*, 401–416. [CrossRef]