sensors

MDPI

Review

A Critical Review of Deep Learning-Based Multi-Sensor Fusion Techniques

Benedict Marsh [1,*], Abdul Hamid Sadka [1] and Hamid Bahai [2]

[1] Institute of Digital Futures, Brunel University London, Kingston Ln, Uxbridge UB8 3PH, UK
[2] Institute of Materials and Manufacturing, Brunel University London, Kingston Ln, Uxbridge UB8 3PH, UK
* Correspondence: benedict.marsh@brunel.ac.uk

Abstract: In this review, we provide a detailed coverage of multi-sensor fusion techniques that use RGB stereo images and a sparse LiDAR-projected depth map as input data to output a dense depth map prediction. We cover state-of-the-art fusion techniques which, in recent years, have been deep learning-based methods that are end-to-end trainable. We then conduct a comparative evaluation of the state-of-the-art techniques and provide a detailed analysis of their strengths and limitations as well as the applications they are best suited for.

Keywords: sensor fusion; stereo; LiDAR; deep learning

1. Introduction

Multi-sensor fusion is the process of combining data that have been captured from multiple sensors. The data from the different sensors will be fused into a single representation of the scene that is more accurate than if it were to be computed using any one of the input data alone.

These fusion techniques can be applied to generate an accurate 3D representation of a real-world scene using multiple sensors. The fusion of the sensor data will allow more accurate 3D models in environments where one sensor is less accurate. In this paper, the techniques that we will consider use RGB cameras and a LiDAR sensor and combine the data in a single dense depth map.

The RGB cameras will capture two stereo images and the LiDAR sensor will capture a 3D point cloud. The point cloud will be projected onto one of the RGB images and the fusion method will be used to combine these data into a single dense depth map.

The depth computed from the stereo will be less reliable in environments with high occlusion and a lack of textured surfaces. LiDAR sensors will capture point clouds which will result in sparse depth maps when projected. These modalities can be combined into a depth map which is dense and can be more accurate in an environment with a lack of occlusion and less texture, which is useful for autonomous driving applications.

The depth map can be computed using the two RGB cameras with stereo matching to obtain a disparity map which can be converted into a depth map with the camera parameters. Recently, the most predominant techniques for computing disparity from rectified stereo images have been deep learning-based methods [1–11], starting with the MC-CNN [12], and then extended-use end-to-end trainable networks, starting with DispNet [13].

The depth map can also be computed by using the LiDAR-projected sparse depth map and using its corresponding RGB image as guidance to compute the dense depth map. The recent methods for depth completion from sparse depth maps are mostly deep learning-based methods [14–22]. These methods usually use extra input data modalities as well as the sparse LiDAR, such as an RGB image or a semantic map.

check for updates

Citation: Marsh, B.; Sadka, A.H.; Bahai, H. A Critical Review of Deep-Learning-Based Multi-Sensor Fusion Techniques. *Sensors* **2022**, *22*, 9364. https://doi.org/10.3390/s22239364

Academic Editor: Andrzej Stateczny

Received: 28 October 2022
Accepted: 26 November 2022
Published: 1 December 2022

The depth map can be computed using both the RBG stereo images and the sparse LiDAR depth map. The recent methods are deep learning based [23–25] and use similar network architectures to the deep learning-based stereo methods.

The other methods we will not consider are computing the depth map from a single RGB image and using only the sparse LiDAR depth map because we are only covering the methods that use data from multiple sensors for the data fusion.

2. State-of-the-Art Multi-Sensor Fusion Techniques and Their Performance Evaluation

In this section, we review the state-of-the-art methods used for computing the disparity or depth maps using RGB and LiDAR. We evaluate the performance of the method using the metrics reported and the visual results from the depth or disparity maps computed, using colour to visualise the depth or disparity values. The methods can be categorised by the input data they use to compute the fused depth map as shown in Table 1. The main types are using stereo images as the input data, using a LiDAR-projected depth map and stereo images or using a LiDAR-projected depth map and a single RGB image.

Table 1. The methods categorised by the input data used.

Stereo Methods	LiDAR and Stereo Methods	LiDAR and RGB Methods
• Raft-stereo: Multilevel recurrent field transforms for stereo matching [26]. • Practical Stereo Matching via Cascaded Recurrent Network with Adaptive Correlation [27]. • Hierarchical Neural Architecture Search for Deep Stereo Matching [28].	• PENet: Towards Precise and Efficient Image Guided Depth Completion [29]. • SemAttNet: Towards Attention-based Semantic Aware Guided Depth Completion [30]. • Dynamic Spatial Propagation Network for Depth Completion [31].	• Volumetric Propagation Network: Stereo-LiDAR Fusion for Long-Range Depth Estimation [32]. • 3D LiDAR and Stereo Fusion using Stereo Matching Network with Conditional Cost Volume Normalization [33]. • Noise-Aware Unsupervised Deep LiDAR-Stereo Fusion [34].

The methods that use stereo or LiDAR and stereo as the input data follow a similar technique. They use a feature network to extract the feature maps from both of the stereo images. Then, by matching the features from one of the stereo images with the features from the other stereo image, a matching cost can be computed for each possible disparity. Then, further networks can be used to compute the final disparity map prediction, and if it is a LiDAR and stereo method, the networks will use the LiDAR data as an extra input to improve the accuracy when computing the matching cost.

The methods that use LiDAR and a single RGB image as input data also follow a similar technique of using a network to compute an initial coarse depth map prediction. This prediction is then refined using features from the network to compute multiple updates to the depth map which will increase the accuracy of the depth map after each update.

2.1. Raft-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching

This method [26] computes the disparity from only RGB stereo images using an approach similar to RAFT [35]. First, the feature maps are extracted from the left and right images using a feature network which is a convolutional neural network. Then, the feature maps are combined into a correlation volume by taking the dot product of each possible feature pair corresponding to a possible disparity. A correlation pyramid is constructed by repeatedly downsampling the dimension corresponding to the disparity in the correlation volume to obtain multiple correlation volumes.

An initial prediction of 0 disparity for all pixels is used as a starting point on the disparity map. The initial disparity map is updated by using a recurrent neural network. To obtain the inputs for the recurrent network, the correlation pyramid is indexed by selecting windows in each volume around the current disparity estimates. The windows are concatenated together into a single feature map which is then fed through convolutional layers to obtain the correlation features, and the current disparity prediction is also fed through convolutional layers to obtain the disparity features. The left image is fed through a context network to extract the context features. The correlation, disparity and context

features are concatenated into a single feature map and used as the input for a multi-scale GRU.

The multi-scale GRU (Figure 1) is composed of three GRUs at multiple scales. The first is 1/32 the resolution of the disparity map and the output is upsampled and imputed into the higher-resolution 1/16 GRU. Then, the output is upsampled and inputted into the highest resolution 1/8 GRU to output the update to the disparity map estimate to obtain the new disparity map update with an element-wise addition. The correlation-pyramid lookup is used to input into the highest resolution 1/8 GRU and the downsampled current disparity map estimate is used as an input for the other two GRUs, and the middle resolution 1/16 GRU also uses the highest resolution 1/8 feature map as an input.

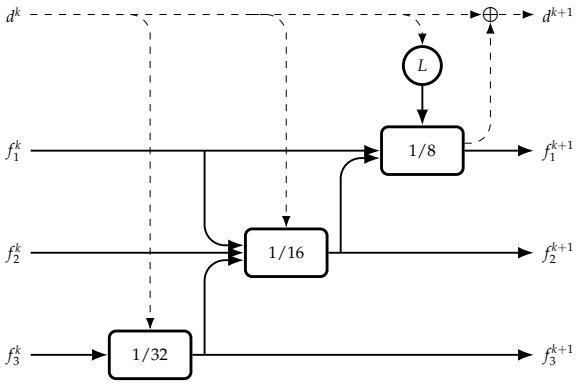

Figure 1. The multi-scale GRU shown with 3 feature maps f_1, f_2, f_3 at different scales $1/8, 1/16, 1/32$. The disparity map d^k shown is updated by adding the output of the lowest scale GRU 1/8, the correlation-pyramid lookup operation is shown as L and the result is inputted in the lowest scale GRU 1/8. Downsampling and upsampling are used to change the scales of the feature maps or disparity map when being passed between the GRUs.

For training the network, the loss is computed after each update to the disparity map prediction with disparity predictions that took more update steps having a greater weighting.

Performance Evaluation

This method was tested on the KITTI-2015 stereo test dataset [36]. This dataset contains 200 scenes captured from a stereo camera setup mounted on a car. The ground truth is captured using a LiDAR sensor and the evaluation is performed on the percentage of the erroneous pixels averaged over all 200 scenes. The predicted disparity map values represent the pixel disparity between the left–right image so erroneous pixels are where the absolute difference between the ground truth and prediction is greater than a pixel threshold which is chosen to be 3 pixels. The method resulted in 1.96% for all pixels, 2.89% for foreground pixels and 1.75% for background pixels.

The qualitative results in Figure 2 show the fine depth detail in the fence and sign posts. This is due to it outputting the image at full resolution without the need to upsample from a lower-resolution prediction.

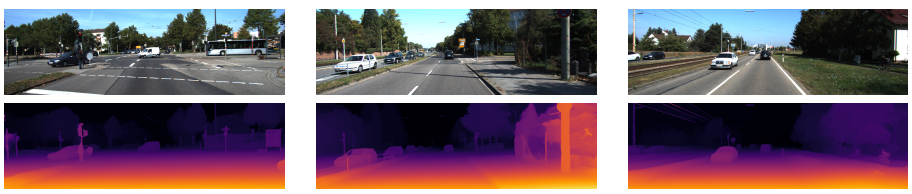

Figure 2. The qualitative results from [26] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values.

2.2. Practical Stereo Matching via Cascaded Recurrent Network with Adaptive Correlation

This method, CREStereo [27], computes a disparity map from a pair of stereo images using a deep convolutional neural network. The disparity map is computed using a recurrent-based architecture to refine the disparity prediction starting from a zero-initialised disparity map.

First, a feature network is used on both left–right stereo images to extract the feature maps which are used to construct three-level feature pyramids for both of the input images. Next, a positional encoding is added to the feature pyramid and then it is inputted through a self-attention layer to obtain the final feature pyramids which are used as the inputs for the recurrent update module.

Next, the recurrent update module uses an adaptive group correlation layer similar to [2]. This layer outputs the inputs for the recurrent neural network from the feature pyramids. This is performed by first using a cross-attention layer to obtain the grouped features which are then sampled using the current predicted disparity map. The sampling is conducted using a search window using fixed offsets which alternates between a 1D and 2D search to allow for stereo matching that does not lie on the epipolar line. The feature pyramid is also used to compute the learned offsets to allow for a deformable search window to be used by extending the method for deformable convolutions [37]. The pyramid sampling will collect features in a window to be used to construct a correlation volume of size $H \times W \times D$ where H, W is the input image height and width, and D is the number of correlation pairs which is smaller than W. The correlation at x, y, d in the correlation volume is computed as follows:

$$corr(x, y, d) = \frac{1}{c} \sum_{i=1}^{C} F_1(i, x, y) F_2(i, x'', y'') \tag{1}$$

where $x'' = x + f(d) + dx$, $y'' = y + g(d) + dy$. F_1, F_2 are the feature maps, $f(d), g(d)$ are the fixed offsets which alternate between 1D when $g(d) = 1$ and 2D when $g(d) \in [-4, 4]$ with $f(d) \in [-4, 4]$ for both and dx, dy are the learned offset which are computed from the feature pyramids.

The correlation volume is used as an input to the recurrent neural network which is formed of multiple gated recurrent units (GRU). The GRUs are each at different scales, and then the final disparity map is computed using cascaded refinement where the outputted disparity map from a lower-level GRU is upsampled and then used as the initial disparity map for the next GRU level to finally obtain the highest level GRU output which is used as the final disparity map.

Performance Evaluation

This method was tested on the KITTI-2015 stereo test dataset using the percentage of erroneous pixels with a three-pixel threshold. The method resulted in 1.69% for all pixels, 2.86% for foreground pixels and 1.45% for background pixels.

The qualitative results in Figure 3 show fine detail in the fence and sign posts similar to [26] because it can also output the full resolution.

Figure 3. The qualitative results from [27] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values.

2.3. Hierarchical Neural Architecture Search for Deep Stereo Matching

This method, LEAStereo (Learning Effective Architecture Stereo) [28], computes a disparity map from only two RGB stereo images using a deep convolutional neural network. The network architecture is found using an NAS (Neural Architecture Search) and then the found network architecture is trained to obtain the final model.

The neural network has four stages to compute the disparity map from the stereo image pair. The first stage uses a 2D feature network to extract a pair of feature maps from the stereo images. The second stage combines the feature maps into a 4D feature volume. The third stage is a 3D matching network that will compute a matching cost for each possible disparity using the 4D feature volume as the input. The final stage will project the 3D cost volume to a 2D disparity map using a soft-argmin layer. Only the feature network and the matching network have trainable parameters so an NAS is used to find the best architecture for these networks.

The architecture search is conducted using a differentiable architecture search [38] where there is a continuous relaxation of the discrete architectures, so the best architecture can be found using gradient descent.

2.3.1. Cell-Level Search Space

The feature network and the matching network are both deep convolutional neural networks that are constructed with computational cells which are stacked in layers and share the same structure. The architecture search is conducted using a two-level hierarchical search [39] that will find the best cell-level and network-level structures.

The cells (Figure 4) are direct acyclic graphs that consist of an ordered sequence of n nodes. The nodes are the feature maps in the convolutional network and the edges are operations that transform the feature maps; so, a node is computed by applying the operations and then combining the feature maps with an element-wise addition. The first two nodes are used as inputs, so for cell C_l in layer l, the two input nodes are outputs from the two preceding cells (C_{l-2}, C_{l-1}). The cell output is computed by concatenating nodes that are not input nodes to the cell. The cells in the final model that will be found will only have two input connections for each node. The architecture search will find these connections and the operation.

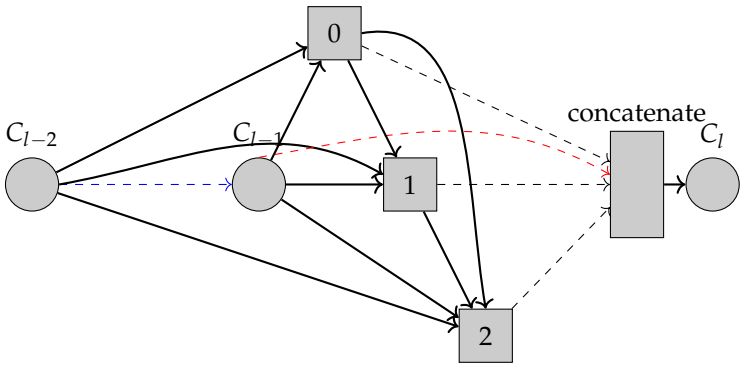

Figure 4. Cell-level search space. The cell output C_l is computed using the previous 2 cell outputs C_{l-2}, C_{l-1}. The intermediate nodes are shown as $0, 1, 2$, and for each input connection to these nodes, an operation will be searched for. The red dashed arrow shows the residual connection and the blue dashed arrow shows the previous cell that uses C_{l-2} as one of its inputs to compute C_{l-1}. The intermediate nodes are concatenated, shown with black dashed arrows to obtain the final output C_l.

To make the architecture continuous in the search phase, the nodes have all the possible connections, so node i is connected to nodes $\{1, 2, ..., i-1\}$, and the edges have all the

possible operations from the set of candidate operations. In the search phase, the output of a node is computed as follows:

$$s^{(j)} = \sum_{i \rightsquigarrow j} o^{(i,j)}(s^{(i)}). \tag{2}$$

where $\rightsquigarrow$ denotes node i is connected to node j. $o^{(i,j)}$ is defined as follows:

$$o^{(i,j)}(x) = \sum_{r=1}^{v} \frac{\exp(\alpha_r^{(i,j)})}{\sum_{s=1}^{v} \exp(\alpha_s^{(i,j)})} o_r^{(i,j)}(x) \tag{3}$$

where $o_r^{(i,j)}(x)$ is the r-th operation between nodes i and j. $\alpha^{(i,j)} = (\alpha_1^{i,j}, \alpha_2^{i,j}, ..., \alpha_v^{i,j})$ is a weight-mixing vector that is used in the softmax function to weight the operations. After the search phase, the operation to be used $O_{r^*}^{(i,j)}$ is selected as the operation with the highest corresponding weight after the search phase $r^* = \arg\max_r \alpha_r^{(i,j)}$. The set of candidate operations for the feature net is $\mathcal{O}^F = \{$"3×3 convolution", "skip connection"$\}$. The set of candidate operations for the matching net is $\mathcal{O}^M = \{$"$3 \times 3 \times 3$ convolution", "skip connection"$\}$. The top two highest operations are selected for each node, so each node has two input connections. Following ResNet [40], the cells are modified to include a residual connection between the output of the previous cell and the cell's output.

2.3.2. Network-Level Search Space

The network-level search space will have a fixed number of layers L that will allow the network to downsample, upsample or keep the same spatial resolution of the feature maps at each layer. The search space has a maximum and minimum resolution the feature map can be, and the network will downsample by a scale of $1/2$ and upsample by a scale of 2.

To make the architecture continuous at the network level, all possible configurations are computed with weighting parameters. In the search phase, the output $h_{(s)}^{(l)}$ of layer l at spatial resolution s is computed as follows:

$$\begin{aligned} H_{(s)}^{(l)} = &\bar{\beta}_{(s/2)} Cell(h_{(s/2)}^{(l-1)}, h_{(s)}^{(l-2)}) \\ &+ \bar{\beta}_{(s)} Cell(h_{(s)}^{(l-1)}, h_{(s)}^{(l-2)}) \\ &+ \bar{\beta}_{(2s)} Cell(h_{(2s)}^{(l-1)}, h_{(s)}^{(l-2)}) \end{aligned} \tag{4}$$

where $Cell(h^{(l-1)}, h^{(l-2)})$ is the cell computation which takes the outputs from the previous two layers $(l-1, l-2)$ as inputs. $\bar{\beta}_s$ is the weight parameter after the softmax function has been applied. The best network architecture is selected as the highest weights for each layer.

The network used 6 layers for the feature network and 12 layers for the matching network. A maximum resolution of $1/3$ of the input size and a minimum of $1/24$ were used for the search space. The cells used five nodes.

2.3.3. Optimisation

The network is searched and trained in an end-to-end manner, so the loss function uses the final disparity output computed by the model and the ground-truth disparity. The loss function is defined as follows:

$$\mathcal{L} = \ell(\mathbf{d}_{pred} - \mathbf{d}_{gt}), \text{where } \ell(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 0 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{d}_{pred}$ is the predicted disparity map by the network and $\mathbf{d}_{gt}$ is the ground-truth disparity map.

The training is performed with two disjoint training sets; one optimises the network parameters and the other optimises the architecture search parameter α, β. The optimisation is performed by alternating the update of the network weights and the network search parameters. The network has been trained on the SceneFlow dataset [13].

2.3.4. Performance Evaluation

This method was tested on the KITTI-2015 stereo test dataset using the percentage of erroneous pixels with a three-pixel threshold. The method resulted in 1.65% for all pixels, 2.91% for foreground pixels and 1.40% for background pixels.

The qualitative results in Figure 5 show less fine detail with smoother edges. This is due to it outputting the disparity map at a lower resolution and then upsampling it.

Figure 5. The qualitative results from [28] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values.

2.4. PENet: Towards Precise and Efficient Image-Guided Depth Completion

This method, PENet [29], computes a depth map from a sparse LiDAR depth map and a single RGB image. The method uses a deep convolutional neural network to compute an initial depth map which is then refined with a convolutional spatial propagation network.

The neural network for computing the initial depth map uses a two-branch backbone as shown in Figure 6. Both of the branches will output a dense depth map prediction and confidence weight map which are used to fuse into a single depth map prediction.

The first branch is the colour-dominant branch which uses both the RGB image and the sparse depth map as inputs. The network uses an encoder–decoder architecture with symmetric skip connections. This network will output a colour-dominant depth map prediction and a confidence map which has confidence values for each pixel.

The second branch is the depth-dominant branch which uses the output depth map prediction of the colour-dominant branch and the sparse depth map for the network inputs, similar to [41]. This branch has a similar architecture as the colour-dominant branch and it also takes the features in the colour-dominant decoder as extra inputs in the corresponding layers for the depth-dominant encoder.

In both the network branches, a geometric convolutional layer is used which is similar to a standard convolutional layer, but it uses additional 3D position maps which are concatenated to the feature inputs to form the input to the geometric convolutional layer. The position map is derived from the sparse LiDAR depth map D as follows:

$$Z = D, X = \frac{(u - u_0)Z}{f_x}, Y = \frac{(v - v_0)Z}{f_y} \tag{6}$$

where (u, v) are the coordinates of a pixel and u_0, v_0, f_x, f_y are the intrinsic parameters of the camera.

The initial depth map prediction is then refined using a convolutional spatial propagation network [14] with dilated convolutions [42].

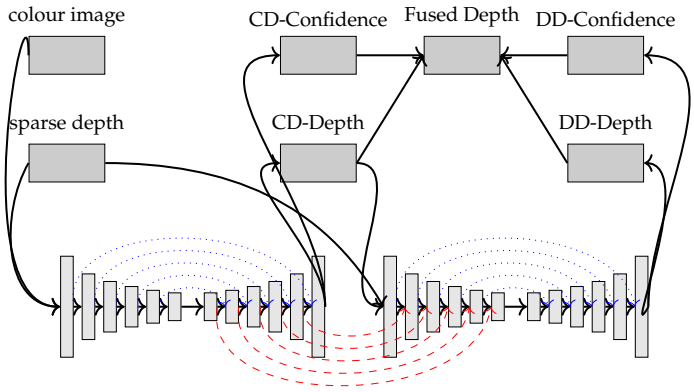

Figure 6. The inputs shown are the colour image and the sparse depth which are inputted into the network to compute the fused depth map which is then further refined. The blue connections show where the features are combined with addition and the red connection show where they are combined with concatenation.

The depth refinement is conducted in iterations using affinity weight maps learnt by the network. For each pixel, it aggregates information propagated from pixels within the neighbourhood for the initial depth map D_0 that has been produced by the network as follows:

$$D_i^{t+1} = W_{ii} D_i^0 + \sum_{j \in \mathcal{N}(i)} W_{ji} D_j^t \tag{7}$$

where W_{ij} is the affinity between pixel i and pixel j. The network will learn the affinity maps.

The l_2 loss is used for training which is defined as follows:

$$L(\hat{D}) = ||(\hat{D} - D_{gt}) \odot \mathbb{1}(D_{gt} > 0)||^2 \tag{8}$$

where $\hat{D}$ is the final predicted depth map, D_{gt} is the ground truth, $\mathbb{1}$ is an indicator and $\odot$ is an element-wise multiplication. This is so it only uses the valid pixels in the ground truth for training. For early epochs, a loss is applied to the intermediate depth predictions from the colour-dominant branch and the depth-dominant branch as follows:

$$L = L(\hat{D}) + \lambda_{cd} L(\hat{D}_{cd}) + \lambda_{dd} L(\hat{D}_{dd}) \tag{9}$$

where λ_{cd} and λ_{dd} are hyperparameters.

Performance Evaluation

This method was tested on the KITTI depth completion validation dataset. This dataset contains 200 scenes captured from a LiDAR sensor and a stereo camera setup mounted on a car. The ground truth is captured using the LiDAR sensor, accumulated from multiple frames. The depth is measured in metres in the data and scaled to millimetres for the evaluation which is performed with the mean absolute error (MAE) in mm, the root mean squared error (RMSE) in mm, the inverse mean absolute error (iMAE) in 1/km and the inverse root mean squared error (iRMSE) in 1/km. The method resulted in 209.00 for the MAE, 757.20 for the RMSE, 0.92 for the iMAE and 2.22 for the iRMSE.

The qualitative results in Figure 7 show noise in the disparity maps because the sparse LiDAR data have large areas with no depth information so there will be high uncertainty in those regions.

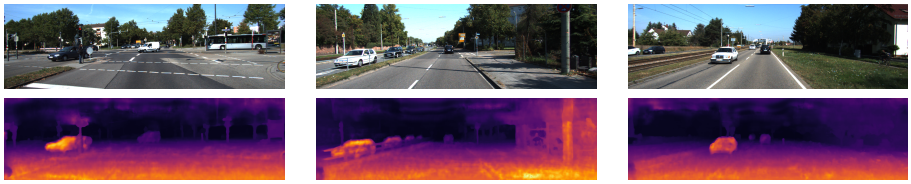

Figure 7. The qualitative results from [29] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values. The raw dataset was used for the LiDAR data mapped onto the scenes in the stereo dataset and then the results were converted from depth values to disparity using the calibration parameters.

2.5. SemAttNet: Towards Attention-Based Semantic-Aware-Guided Depth Completion

This method, SemAttNet [30], computes a depth map using a sparse LiDAR depth map and a single RGB image as inputs (and a semantic map). This method first uses a convolutional neural network to output an initial coarse depth map which is refined with a spatial propagation network.

The network used for computing the initial depth map uses a three-branch backbone which consists of a colour branch, a depth branch and a semantic branch as shown in Figure 8. The depth branch takes the output depth map predictions computed by the colour and semantic branches as well as the sparse LiDAR depth as inputs.

The network uses a semantic-aware multi-modal attention-based fusion block in the initial network [43]. It is first used to fuse the feature maps from the colour and semantic branches and then it is used in the depth branch to fuse the features from the other two branches.

The colour branch uses an encoder–decoder network with skip connections which takes the colour image and the sparse depth map as inputs and will output a dense depth map and a confidence map.

The semantic branch takes the depth map computed with the colour-guided branch as an input as well as the sparse depth map and the semantic image, similar to [29,41]. The network will output a dense depth map and a confidence map. The network uses a similar encoder–decoder architecture and uses the decoder features from the colour-guided branch and fuses them with the features in the encoder network. The semantic image is computed from the RGB image with a pretrained WideResNet38 [44] model.

The depth branch uses the outputted depth maps from the colour branch and the semantic branch as inputs as well as the sparse depth map. The branch uses a similar encoder–decoder network architecture to the other branches, with the decoder features from both branches being fused with the features in the encoder network.

The decoder features are fused with the encoder features using a semantic-aware multi-modal attention-based fusion block. This uses channel-wise attention and then spatial-wise attention to output a single feature map.

The final depth maps outputted from each branch are fused using the corresponding learnt confidence maps outputted by each branch, similar to [29,45].

The training loss computes the l_2 loss for each branch-outputted depth map and the final fused depth map using the ground-truth depth map.

Then, the fused depth map from the three-branch network is refined using the CSPN++ [14] with dilated convolutions.

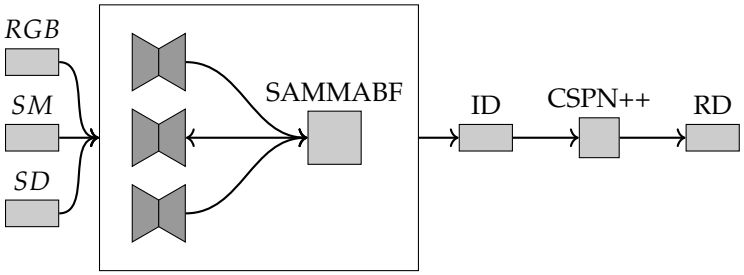

Figure 8. The inputs shown on the left are the RGB image, the semantic map computed from the RGB image and the sparse depth captured by the LiDAR sensor. The features from the three-branch backbone are fused using the semantic-aware multi-modal attention-based fusion block (SAMMABF) and will output the initial depth map shown as *ID*. This is refined with the CSPN++ [14] with dilated convolutions [42] to output the refined depth map shown as *RD*.

Performance Evaluation

This method was tested on the KITTI depth completion validation dataset. The method resulted in 205.49 for the MAE, 709.41 for the RMSE, 0.90 for the iMAE and 2.03 for the iRMSE.

The qualitative results in Figure 9 show that, similar to Figure 7, the disparity map has noise in the regions where there are no LiDAR data.

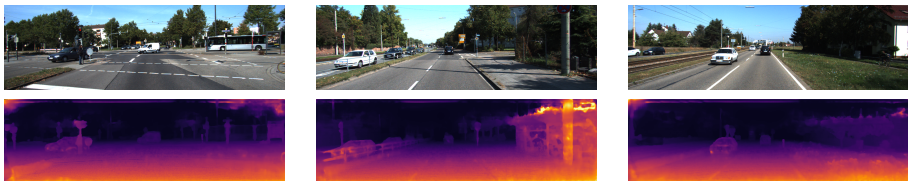

Figure 9. The qualitative results from [30] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values. The raw dataset was used for the LiDAR data mapped onto the scenes in the stereo dataset and then the results were converted from depth values to disparity using the calibration parameters.

2.6. Dynamic Spatial Propagation Network for Depth Completion

This method, DySPN [31], computes a depth map from a sparse LiDAR depth map with a single RGB image as well for guidance. A coarse dense depth map is initially computed with a convolutional neural network and then the coarse depth map is refined using a dynamic spatial propagation network.

The initial coarse depth map is computed from the sparse LiDAR depth map using a Unet-like [46] encoder–decoder network with ResNet-34 [40] as the backbone. This network will output attention maps, the initial depth map, an affinity matrix and a confidence prediction for the input depth [14,29,47] as shown in Figure 10.

The depth map is updated with adaptive affinity weights to refine the depth map over N steps. This can be defined as $V^{t+1} = G^t V^t$ for $t \in \{0, 1, 2, \ldots, N\}$ with the depth map reshaped to the one-dimensional vector $V^t \in \mathbb{R}^{mn}$, and the global adaptive affinity matrix $G^t \in \mathbb{R}^{mn \times mn}$ which contains all adaptive affinity weights can be defined as follows:

$$G^t = \begin{bmatrix} 1 - \tilde{\lambda}^t_{0,0} & \tilde{w}^t_{0,0}(1,0) & \ldots & \tilde{w}^t_{0,0}(m,n) \\ \tilde{w}^t_{1,0}(1,0) & 1 - \tilde{\lambda}^t_{1,0} & \ldots & \tilde{w}^t_{1,0}(m,n) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{w}^t_{m,n}(1,0) & \tilde{w}^t_{m,n}(1,0) & \ldots & 1 - \tilde{\lambda}^t_{m,n} \end{bmatrix} \tag{10}$$

where $\tilde{w}^t_{i,j}(a,b) = \pi^t_{i,j}(a,b)w_{i,j}(a,b)$ is the adaptive affinity weight which is computed using the fixed affinity weights $w_{i,j}(a,b)$ and the global attention $\pi^t_{i,j}(a,b)$. $\tilde{\lambda}^t_{0,0} = \sum_{a \neq i, b \neq j}$ so that each row will sum to 1.

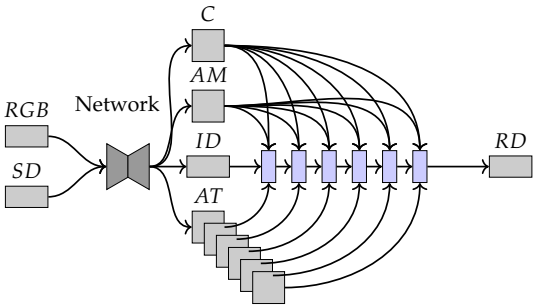

Figure 10. The inputs shown on the left are the RGB image and the sparse depth captured by the LiDAR sensor. The encoder–decoder network outputs the confidence shown as *C*, the affinity matrix as *AM*, the initial depth prediction as *ID* and the attention maps as *AT*. The initial depth is refined over 6 steps of the DySPN, shown in blue, to obtain the final refined depth map shown as *RD*.

The DySPN reduces the computational complexity of the global adaptive affinity update by only computing for a neighbourhood. The affinity weights for the neighbours of the same distance are set to be equal and the neighbourhood is also deformable [48].

The L_1 and L_2 loss are both used to compute the training loss which is defined as $Loss(h^{gt}, h^N) = L_1(h^{gt}, h^N) + L_2(h^{gt}, h^N)$ where h^N is the depth map after N steps and h^{gt} is the ground-truth depth map.

Performance Evaluation

This method was tested on the KITTI depth completion validation dataset. The method resulted in 192.71 for the MAE, 709.12 for the RMSE, 0.82 for the iMAE and 1.88 for the iRMSE.

The qualitative results in Figure 11 also show similar noise in the depth map to the other depth completion methods.

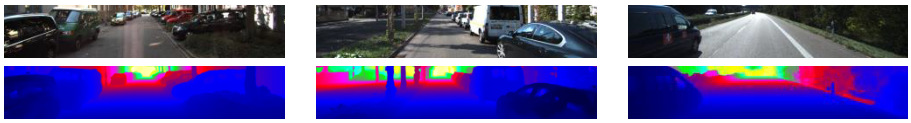

Figure 11. The qualitative results from [31] show the left RGB images from 3 scenes in the KITTI depth completion dataset in the top row and the corresponding depth map predictions in the bottom row are shown using colour to represent the depth values.

2.7. Volumetric Propagation Network: Stereo-LiDAR Fusion for Long-Range Depth Estimation

This method [32] computes a depth map using the RGB stereo images and a LiDAR point cloud of the scene. The method pipeline is shown in Figure 12. First, a fusion volume is constructed to represent the scene as a three-dimensional voxel grid, evenly distributed along the depth range. The fusion volume is filled with information from the stereo images, first by using a feature extraction network and then converting the voxel coordinates onto the stereo images using the calibration parameters. Next, the point clouds are also converted into the fusion volume and stored as a binary occupancy representation.

The final information embedded into the fusion volume is the point cloud features which are extracted using FusionConv layers that cluster neighbouring points from the point cloud that lie within a voxel window. These points are fused with the corresponding features in the left feature map and then the weighted geometric distance is computed to obtain the weights for the convolution operation on the neighbouring point features.

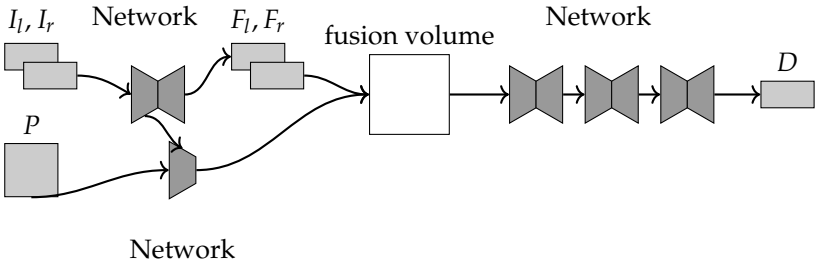

Figure 12. The inputs shown are the point cloud P and the left–right RGB image pair I_l, I_r used to compute feature maps F_l, F_r, and the final depth map D is computed from the fusion volume.

The final fusion volume is fed through a stacked hourglass network [1,49] that consists of multiple encoder and decoder networks which compute multiple cost–volume estimates. The depth is computed using the soft-min function. The training total loss is computed with a weighted sum of the loss for each depth map estimate in the hourglass network.

Performance Evaluation

This method was tested on the KITTI depth completion validation dataset. The method resulted in 205.1 for the MAE, 636.2 for the RMSE, 0.9870 for the iMAE and 1.8721 for the iRMSE.

The qualitative results in Figure 13 show the depth map has less noise than the depth completion method because it also uses stereo matching.

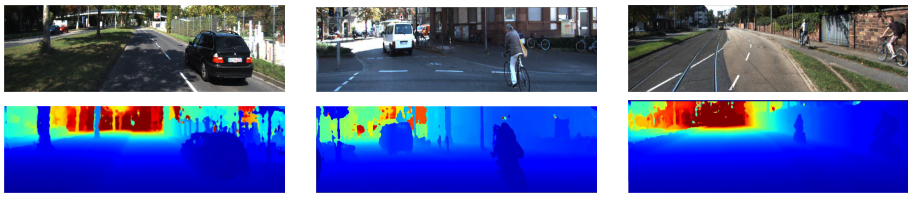

Figure 13. The qualitative results from [32] show the left RGB images from 3 scenes in the KITTI raw dataset in the top row and the corresponding depth map predictions in the bottom row are shown using colour to represent the depth values.

2.8. Three-Dimensional LiDAR and Stereo Fusion using Stereo Matching Network with Conditional Cost Volume Normalization

This method [33] computes a disparity map by using the RGB images and sparse disparity maps of a LiDAR point cloud projected onto the RGB image views. The stereo and LiDAR data are fused by concatenating the RGB image with the LiDAR disparity map to create a four-channel image and then this is inputted into a single feature extraction network using 2D convolutional layers to obtain the feature maps for both the left and right images. The features are used to construct a 4D feature volume which contains all the potential matches. Next, a matching network is used to apply regularisation by using 3D convolutional layers to obtain the final cost volume. Finally, the soft-min function can be used to obtain the final disparity map.

The cost–volume is normalised by using Conditional Cost–Volume Normalisation which is inspired by Conditional Batch Normalisation [50,51]. In conditional Batch Normalisation , the learnable parameters are defined as functions of conditional information, and for conditional cost–volume normalisation, the information used is the pixels in the LiDAR data. For the pixels with no values, a learnable constant value is used instead. The conditional cost–volume method is approximated with a hierarchical extension to reduce the number of parameters by computing an intermediate vector conditioned on each LiDAR pixel and then modulating this with learnable parameters for each depth in the volume. This conditional cost–volume regularisation is used in the matching network to compute the final cost–volume from the feature volume.

Performance Evaluation

This method was tested on the KITTI depth completion validation dataset. The method resulted in 252.5 for the MAE, 749.3 for the RMSE, 0.8069 for the iMAE and 1.3968 for the iRMSE.

The qualitative results in Figure 14 show less noise than the depth completion methods; however, the visual quality is lower than the methods that only use stereo data.

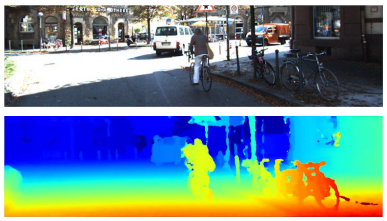

Figure 14. The qualitative results from [33] show the left RGB image from a scene in the KITTI depth completion dataset in the top row and the corresponding depth map prediction in the bottom row are shown using colour to represent the depth values.

2.9. Noise-Aware Unsupervised Deep LiDAR–Stereo Fusion

This method [34] fuses a pair of stereo RGB images with the LiDAR-projected disparity maps using an unsupervised training method without the need for training data with a ground-truth disparity map.

The method uses a deep convolutional neural network that computes the final disparity maps using multiple stages, similar to [7] as shown in Figure 15. The first stage extracts the feature from the stereo images and features from the LiDAR disparity maps. The network uses sparsity invariant convolutional layers [52] for the sparse LiDAR disparity maps. The stereo and LiDAR feature maps are concatenated into a single feature map for both the left and right Images. The next stage constructs a 4D feature volume using both feature maps and then feature matching is computed using 3D convolutions to compute the cost–volume. Finally, the disparity map is computed using a soft-argmin on the cost–volume.

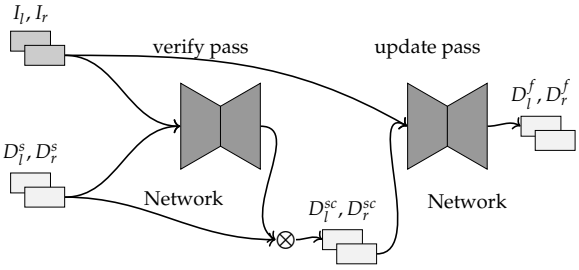

Figure 15. The inputs shown are the left–right RGB images I_l, I_r and the left–right sparse disparity maps D_l^s, D_r^s. After the verify pass, the filtered disparity maps are computed D_l^{sc}, D_r^{sc} and this removes disparities that are not consistent. After the update pass, the fused disparity maps are computed D_l^f, D_r^f.

The data are passed through this network two times. The first produces the initial disparity maps that are used to remove the noisy LiDAR disparities so the filtered LiDAR disparity maps are used in the second pass to compute the final disparity map. The network is trained using an unsupervised loss function that is the sum of an image warping loss, a LiDAR loss, a smoothness loss and a plane fitting loss. The image warping loss computes the photometric consistency between the stereo images using the final disparity map. The LiDAR loss computes the distance between the filtered LiDAR disparity map and the final disparity map. The smoothness loss computes the smoothness of the final disparity map. The plane fitting loss computes the distance between the final disparity map and a disparity map created by projecting the disparity values onto a local plane. The network is

only updated after the LiDAR disparities have been filtered. Backpropagation will only go through the final network pass after filtering.

Performance Evaluation

This method was tested on the KITTI stereo dataset with only the frames that have LiDAR data available to create a subset of 141 frames (KITTI-141 subset). The evaluation was performed with the absolute relative error (Abs rel), the maximum ratio threshold error ($\delta < 1.25$) and the percentage of erroneous pixels, with pixel thresholds of 2, 3 and 5 (>2 px, >3 px, >5 px). The method resulted in 0.0350 for the Abs rel, 0.0287 for >2 px, 0.0198 for >3 px, 0.0126 for >3 px and 0.9872 for $\delta < 1.25$.

The qualitative results in Figure 16 show less fine detail in the disparity map than [26,27] but less noise than the depth completion methods.

Figure 16. The qualitative results from [34] show the left RGB images from 3 scenes in the KITTI-2015 stereo dataset in the top row and the corresponding disparity map predictions in the bottom row are shown using colour to represent the disparity values. The raw dataset was used for the LiDAR data mapped onto the scenes in the stereo dataset and then the results were converted from depth values to disparity using the calibration parameters.

3. Comparative Evaluation of the State-of-the-Art Fusion Techniques

In this section, we compare the performance of the multi-modal fusion methods by using the results from the KITTI depth completion dataset and the KITTI stereo dataset.

We compare the Volumetric Propagation Network (VPN) method, the 3D LiDAR and Stereo Fusion using a Stereo Matching Network with Conditional Cost–Volume Normalization (CCVN) method, the PENet: Towards Precise and Efficient Image-Guided Depth Completion (PENet) method, the SemAttNet: Towards Attention-based Semantic-Aware-Guided Depth Completion (SemAttNet) method and the Dynamic Spatial Propagation Network for Depth Completion (DySPN).

We compare the Raft-stereo: Multilevel recurrent field transforms for the stereo matching (RAFT) method, the Noise-Aware Unsupervised Deep LiDAR–Stereo Fusion (LiDARStereoNet) method and the Hierarchical Neural Architecture Search for Deep Stereo Matching (LEAStereo) method on the KITTI stereo dataset.

The compared results (Table 2) show the methods tested on the KITTI depth completion dataset which has LiDAR and stereo data. The other results (Table 3) show the methods tested on the KITTI stereo dataset and evaluated using a pixel threshold error which is the percentage of values with a greater error than 3 pixels. The LEAStereo method has the highest accuracy on the stereo dataset, using only a pair of RGB stereo images. The VPN method has the highest accuracy with the RMSE metric and the CCVN method with the iMAE and iRMSE metrics, and these are evaluated on the depth completion dataset which has LiDAR and RGB so the accuracy will be higher than RGB only. However, the DySPN method has the highest accuracy with the MAE metric which uses LiDAR and only one RGB image.

The compared methods (Table 4) show the strengths and limitations of each method. The methods with a slower inference time take over 1 s to compute the depth map due to using stereo and LiDAR data as the input. The methods with a very fast inference time take less than 0.25 s to compute the depth map due to using LiDAR and RGB without stereo data. The methods that have a high memory usage will be limited by the memory available when computing the depth maps so downsampling is used to reduce the memory usage which can reduce the accuracy.

Table 2. Comparison of the methods that have been tested on the KITTI depth completion dataset showing the mean absolute error (MAE) and root mean squared error (RMSE), both in millimetres, and also the inverse mean absolute error (iMAE) and inverse root mean squared error (iRMSE), both in one/kilometres. The results with the highest accuracy are shown in bold.

	MAE (mm)	RMSE (mm)	iMAE (1/km)	iRMSE (1/km)
CCVN [33]	252.5	749.3	**0.8069**	**1.3968**
PENet [29]	209.0	757.2	0.9	2.22
SemAttNet [30]	205.49	709.41	0.90	2.03
VPN [32]	205.1	**636.2**	0.9870	1.8721
DySPN [31]	**192.71**	709.12	0.82	1.88

Table 3. Comparison of the methods that have been tested on the KITTI stereo dataset showing the percentage of pixels with an error greater than 3 pixels. The result with the highest accuracy is shown in bold.

	3.0 px
LiDARStereoNet [1] [34]	1.98
RAFT [26]	1.96
CREStereo [27]	1.69
LEAStereo [29]	**1.65**

[1] LiDARStereoNet is evaluated on a subset of the stereo dataset containing 141 scenes out of 200.

Table 4. The strengths and limitations of each method.

Methods	Strengths	Limitations
Raft [26]	• lower memory usage from indexing features • model is available to download	• only RGB sensors used • lower accuracy
CREStereo [27]	• lower memory usage from indexing features • fast inference time of 0.41 s • higher accuracy • model is available to download	• only RGB sensors used
LEAStereo [28]	• fast inference time of 0.30 s • higher accuracy • model is available to download	• higher complexity from using neural architecture search • higher memory usage due to using 3D convolutions • only RGB sensors used
PENet [29]	• multiple sensor modalities used • lower memory usage due to only using 2D convolutions • very fast inference time 0.032 s • model is available to download	• lower accuracy
SemAttNet [30]	• lower memory usage due to only using 2D convolutions • multiple sensor modalities used • very fast inference time of 0.2 s • model is available to download	• higher complexity from using attention-based modules • lower accuracy
DySPN [31]	• multiple sensor modalities used • lower memory usage due to only using 2D convolutions • very fast inference time of 0.16 s • higher accuracy	• model is not available to download

Table 4. *Cont.*

Methods	Strengths	Limitations
VPN [32]	• multiple sensor modalities used • higher accuracy	• higher memory usage due to using 3D convolutions • slow inference time of 1.4 s • model is not available to download
CCVN [33]	• multiple sensor modalities used • higher accuracy	• higher memory usage due to using 3D convolutions • slow inference time of 1.011 s • model is not available to download
LiDARStereoNet [34]	• multiple sensor modalities used • model is available to download	• higher complexity from using an unsupervised training loss • higher memory usage due to using 3D convolutions • lower accuracy

4. Results Analysis, Discussions and Application Areas

The methods that we compare are all evaluated using the KITTI dataset [36] which is developed for the application of autonomous driving. The dataset was created using sensors mounted to a vehicle and contains diverse scenes of urban, rural and highways.

Shown in Table 4, the methods VPN, CCVN, LiDARStereoNet and LEAStereo are limited by the high memory usage because they all use a feature volume to compute the final depth map which has a high memory cost. This can limit the methods to outputting a depth map at a lower resolution and then upsampling it, which will result in losing fine 3D detail. These methods perform well on the KITTI dataset, with VPN having the highest accuracy and MAE and CCVN having the highest accuracy for the iMAE and IRMSE. LEAStereo also has the highest accuracy on the KITTI stereo dataset; however, these methods would not be able to efficiently compute depth maps for datasets with larger resolution data.

The methods RAFT and CREStereo use a recurrent architecture that indexes features to improve the computational and memory efficiency, so increasing the resolution of the RGB input images would not have as high a memory cost as the feature volume methods.

The methods PENet, SemAttNet and DySPN do not use stereo RGB images as inputs; instead, they only use a single RGB image which reduces the memory cost of the computation and would be more efficient at computing depth maps for higher resolution inputs.

The specific area these methods can be applied to is computing a 3D model of a road environment to be used for autonomous driving. The methods RAFT, CREStereo and LEAStereo all rely on feature matching from stereo images so they would only be suitable for autonomous driving in environments with texture so features can be easily matched. These methods would perform worse for autonomous driving in environments with textureless areas, such as building interiors with walls without any patterns. The methods using the LiDAR depth map as an input, such as the PENet, SemAttNet, DySPN, VPN, CCVN, and LiDARStereoNet, would be more suitable for these indoor environments.

5. Conclusions

The state-of-the-art LiDAR and stereo fusion methods show a lower accuracy than the current state-of-the-art depth completion or stereo methods when using the MAE or pixel threshold error to evaluate accuracy. In the other metrics used (RMSE, iMAE and iRMSE), the LiDAR–stereo fusion methods show a higher accuracy. The LiDAR–stereo methods also require more computing time and have higher memory usage when compared to the LiDAR–RGB methods. The LiDAR–stereo methods would be more suitable in environments where one of the sensor modalities will have a low accuracy.

Author Contributions: Conceptualisation, B.M., A.H.S. and H.B.; Methodology, B.M. and A.H.S.; Software, B.M.; Validation, B.M. and A.H.S.; Formal Analysis, B.M. and A.H.S.; Investigation, B.M. and A.H.S.; Resources, B.M.; Data Curation, B.M.; Writing—Original Draft Preparation, B.M.; Writing—Review and Editing, B.M., A.H.S. and H.B.; Visualisation, B.M. and A.H.S.; Supervision, A.H.S. and H.B.; Project Administration, A.H.S. and H.B.; Funding Acquisition, A.H.S. and H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository.

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

References

1. Chang, J.R.; Chen, Y.S. Pyramid stereo matching network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5410–5418.
2. Guo, X.; Yang, K.; Yang, W.; Wang, X.; Li, H. Group-wise correlation stereo network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3273–3282.
3. Kendall, A.; Martirosyan, H.; Dasgupta, S.; Henry, P.; Kennedy, R.; Bachrach, A.; Bry, A. End-to-end learning of geometry and context for deep stereo regression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 66–75.
4. Yao, Y.; Luo, Z.; Li, S.; Fang, T.; Quan, L. Mvsnet: Depth inference for unstructured multi-view stereo. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 767–783.
5. Zhang, F.; Prisacariu, V.; Yang, R.; Torr, P.H. Ga-net: Guided aggregation net for end-to-end stereo matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 185–194.
6. Zhang, F.; Qi, X.; Yang, R.; Prisacariu, V.; Wah, B.; Torr, P. Domain-invariant stereo matching networks. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 420–439.
7. Zhong, Y.; Dai, Y.; Li, H. Self-supervised learning for stereo matching with self-improving ability. *arXiv* **2017**, arXiv:1709.00930.
8. Tankovich, V.; Hane, C.; Zhang, Y.; Kowdle, A.; Fanello, S.; Bouaziz, S. Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14362–14372.
9. Li, Z.; Liu, X.; Drenkow, N.; Ding, A.; Creighton, F.X.; Taylor, R.H.; Unberath, M. Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 6197–6206.
10. Khamis, S.; Fanello, S.; Rhemann, C.; Kowdle, A.; Valentin, J.; Izadi, S. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September2018 ; pp. 573–590.
11. Yang, G.; Manela, J.; Happold, M.; Ramanan, D. Hierarchical deep stereo matching on high-resolution images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5515–5524.
12. Zbontar, J.; LeCun, Y. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.* **2016**, *17*, 2287–2318.
13. Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; Brox, T. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4040–4048.
14. Cheng, X.; Wang, P.; Guan, C.; Yang, R. CSPN++: Learning Context and Resource Aware Convolutional Spatial Propagation Networks for Depth Completion. *arXiv* **2019**, arXiv:1911.05377.
15. Qiu, J.; Cui, Z.; Zhang, Y.; Zhang, X.; Liu, S.; Zeng, B.; Pollefeys, M. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3313–3322.
16. Yan, Z.; Wang, K.; Li, X.; Zhang, Z.; Xu, B.; Li, J.; Yang, J. RigNet: Repetitive image guided network for depth completion. *arXiv* **2021**, arXiv:2107.13802.
17. Lee, B.U.; Lee, K.; Kweon, I.S. Depth completion using plane-residual representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13916–13925.
18. Zhang, C.; Tang, Y.; Zhao, C.; Sun, Q.; Ye, Z.; Kurths, J. Multitask gans for semantic segmentation and depth completion with cycle consistency. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 5404–5415. [CrossRef] [PubMed]

19. Zhao, S.; Gong, M.; Fu, H.; Tao, D. Adaptive context-aware multi-modal network for depth completion. *IEEE Trans. Image Process.* **2021**, *30*, 5264–5276. [CrossRef] [PubMed]
20. Liu, L.; Song, X.; Lyu, X.; Diao, J.; Wang, M.; Liu, Y.; Zhang, L. FCFR-Net: Feature Fusion based Coarse-to-Fine Residual Learning for Depth Completion. *arXiv* **2020**, arXiv:2012.08270.
21. Chen, Y.; Yang, B.; Liang, M.; Urtasun, R. Learning joint 2d-3d representations for depth completion. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October 2019–2 November 2019; pp. 10023–10032.
22. Cheng, X.; Wang, P.; Yang, R. Depth estimation via affinity learned with convolutional spatial propagation network. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 103–119.
23. Park, K.; Kim, S.; Sohn, K. High-precision depth estimation with the 3d lidar and stereo fusion. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2156–2163.
24. Zhang, J.; Ramanagopal, M.S.; Vasudevan, R.; Johnson-Roberson, M. Listereo: Generate dense depth maps from lidar and stereo imagery. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 7829–7836.
25. Maddern, W.; Newman, P. Real-time probabilistic fusion of sparse 3d lidar and dense stereo. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; pp. 2181–2188.
26. Lipson, L.; Teed, Z.; Deng, J. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In Proceedings of the 2021 International Conference on 3D Vision (3DV), London, UK, 1–3 December 2021; pp. 218–227.
27. Li, J.; Wang, P.; Xiong, P.; Cai, T.; Yan, Z.; Yang, L.; Liu, J.; Fan, H.; Liu, S. Practical stereo matching via cascaded recurrent network with adaptive correlation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 16263–16272.
28. Cheng, X.; Zhong, Y.; Harandi, M.; Dai, Y.; Chang, X.; Li, H.; Drummond, T.; Ge, Z. Hierarchical neural architecture search for deep stereo matching. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22158–22169.
29. Hu, M.; Wang, S.; Li, B.; Ning, S.; Fan, L.; Gong, X. Penet: Towards precise and efficient image guided depth completion. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 13656–13662.
30. Nazir, D.; Liwicki, M.; Stricker, D.; Afzal, M.Z. SemAttNet: Towards Attention-based Semantic Aware Guided Depth Completion. *arXiv* **2022**, arXiv:2204.13635.
31. Lin, Y.; Cheng, T.; Zhong, Q.; Zhou, W.; Yang, H. Dynamic spatial propagation network for depth completion. *arXiv* **2022**, arXiv:2202.09769.
32. Choe, J.; Joo, K.; Imtiaz, T.; Kweon, I.S. Volumetric propagation network: Stereo-lidar fusion for long-range depth estimation. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4672–4679. [CrossRef]
33. Wang, T.H.; Hu, H.N.; Lin, C.H.; Tsai, Y.H.; Chiu, W.C.; Sun, M. 3D lidar and stereo fusion using stereo matching network with conditional cost volume normalization. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 5895–5902.
34. Cheng, X.; Zhong, Y.; Dai, Y.; Ji, P.; Li, H. Noise-aware unsupervised deep lidar-stereo fusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6339–6348.
35. Teed, Z.; Deng, J. Raft: Recurrent all-pairs field transforms for optical flow. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 402–419.
36. Menze, M.; Geiger, A. Object scene flow for autonomous vehicles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3061–3070.
37. Zhu, X.; Hu, H.; Lin, S.; Dai, J. Deformable convnets v2: More deformable, better results. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9308–9316.
38. Liu, H.; Simonyan, K.; Yang, Y. Darts: Differentiable architecture search. *arXiv* **2018**, arXiv:1806.09055.
39. Liu, C.; Chen, L.C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.L.; Fei-Fei, L. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 82–92.
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27 June–1 July 2016; pp. 770–778.
41. Tang, J.; Tian, F.P.; Feng, W.; Li, J.; Tan, P. Learning guided convolutional network for depth completion. *IEEE Trans. Image Process.* **2020**, *30*, 1116–1129. [CrossRef] [PubMed]
42. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2015**, arXiv:1511.07122.
43. Fooladgar, F.; Kasaei, S. Multi-modal attention-based fusion model for semantic segmentation of rgb-depth images. *arXiv* **2019**, arXiv:1912.11691.
44. Wu, Z.; Shen, C.; Van Den Hengel, A. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognit.* **2019**, *90*, 119–133. [CrossRef]

45. Van Gansbeke, W.; Neven, D.; De Brabandere, B.; Van Gool, L. Sparse and noisy lidar completion with rgb guidance and uncertainty. In Proceedings of the 2019 16th International Conference on Machine vision applications (MVA), Tokyo, Japan, 27–31 May 2019; pp. 1–6.
46. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
47. Cheng, X.; Wang, P.; Zhou, Y.; Guan, C.; Yang, R. Omnidirectional depth extension networks. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 589–595.
48. Xu, Z.; Yin, H.; Yao, J. Deformable spatial propagation networks for depth completion. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 913–917.
49. Newell, A.; Yang, K.; Deng, J. Stacked hourglass networks for human pose estimation. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 483–499.
50. Perez, E.; De Vries, H.; Strub, F.; Dumoulin, V.; Courville, A. Learning visual reasoning without strong priors. *arXiv* **2017**, arXiv:1707.03017.
51. De Vries, H.; Strub, F.; Mary, J.; Larochelle, H.; Pietquin, O.; Courville, A.C. Modulating early visual processing by language. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6597–6607 .
52. Uhrig, J.; Schneider, N.; Schneider, L.; Franke, U.; Brox, T.; Geiger, A. Sparsity invariant CNNs. In Proceedings of the 2017 International Conference on 3D Vision (3DV), Qingdao, China, 10–12 October 2017; pp. 11–20.