Universiteit Leiden
The Netherlands

Opleiding Informatica

Fusion approaches for

Monocular Depth Estimation

David Schep

Supervisor:
Michael Lew

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

28/07/2020

Abstract

Estimating depth using a single image is a very useful research field with applications in augmented reality, self-driving cars and computational photography. Recently significant progress has been made due to the innovative Deep Convolutional Neural Networks. In this thesis we will not innovate by producing a new DCNN architecture but instead we fuse different existing DCNN models using multiple fusion methods; the average, weighted average and median. The performance of these fusions is compared using two datasets; NYU Depth-v2 and KITTI. We show that fusing models will always give better performance but choosing the right models to fuse is very important. Models that have significant difference in the architecture are more likely to fuse well. We also show that fusing more models yield better results but will quickly give diminishing returns. Fusing only two well chosen models will give performance close to the performance of a three model fusion. [1]

Contents

[1] All figures and tables were created by the author

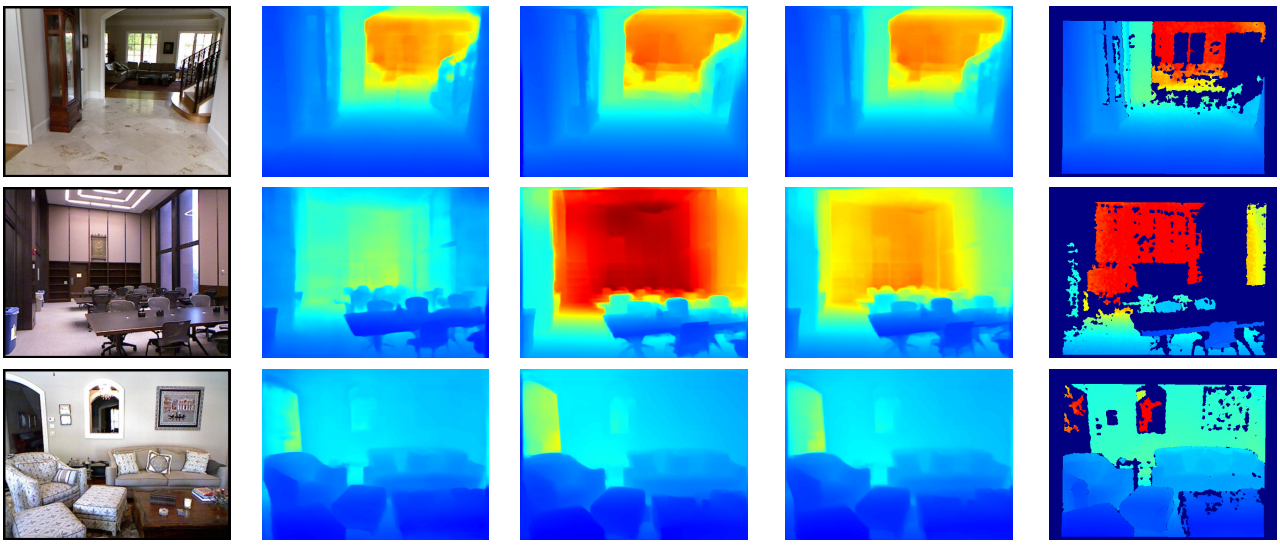

Figure 1: Monocular Depth Estimation on the NYU Depth-v2 dataset. Fusion-avg (ours) averages BTS and VNL.

1 Introduction

In this thesis we will take a different look at the problem of monocular depth estimation. Monocular depth estimation is the process of taking an RGB image as input and predicting the distance to the camera for every pixel in the scene. Figure 1 shows examples of monocular depth estimation on the NYU Depth-v2 dataset.

We as humans perceive depth using two eyes. There exist techniques that use two cameras to give a computer depth perception. The field of monocular depth estimation focuses on the subset of the depth perception problem where only a single camera is used. This makes predicting depth significantly more difficult. When we close one eye we still have an idea of how far away objects are because of the experience we have with seeing depth. But when you can not use both eyes it is possible to misjudge the distance to an object. Maybe an object is suddenly smaller and closer than usual. For example, a very detailed model of a tree placed right outside your window might give you the impression of a large tree being further away. This illustrates a key property of the monocular depth estimation problem. A single image could map to an infinite amount of correct depth maps. We could take three identical images of the window with the tree. In every image we scale up the tree and move it further away. The image would not change but the correct depth map would change. This means the problem is not solvable. Humans can only estimate the depth of objects in a scene using our experience. A tree is usually not closer than a few meters and a phone is often closer than a meter. By recognising objects we can estimate the distance. This makes the problem very difficult for a computer to solve. As a computer usually does not understand a scene and thus cannot infer whether an object is close or far away.

The first attempt to solve this problem using a machine learning based method was done by A. Saxena et al. [1]. They applied a linear regression model combined with a Markov random field

(MRF) to the problem. Machine learning quickly evolved and deep learning was introduced. Soon after, state-of-the-art Deep Convolutional Neural Networks were applied to Monocular Depth Estimation and achieved impressive results. DCNNs were often trained with very large datasets like DITW [2], KITTI [3] and NYU Depth-v2 [4]. The DCNNs are very well researched and every few months a new architecture is designed which advances the field even more. This paper will not focus on designing a new architecture but instead will show whether fusing the output of known models will lead to improved performance.

Combining models into a single model is a common practice in the machine learning field. The combined models are often called *ensembles*. It has been shown that an ensemble model will often lead to better accuracy [5]. However most often the models are combined before training and are trained all at the same time. Instead we will focus on models that have been pre-trained on the same dataset and for the same problem. In this paper we will combine three state-of-the-art DCNNs designed and trained for monocular depth estimation. The models will be combined using different fusion methods; The normal average, weighted average and median.

This chapter contains the introduction. In Section 2 related work will be discussed. Section 3 introduces the three models that have been chosen to be fused. We will go over the architectures and differences between these models. Section 4 explains the fusion methods used to combine the models and how the weighted mean weights were optimized. In Section 5 the datasets used for the experiments are introduced. Section 5.3 discusses the evaluation method and the metrics that have been used. Section 5.4 shows the results of the fused models on the NYU Depth-v2 dataset and compares them to the base models. The performance of the different fusion methods are also compared. The results of the fused models evaluated using the KITTI dataset are shown in Section 5.5. And finally Section 6 concludes.
This bachelor thesis has been written at the Leiden University LIACS faculty with supervision from Michael Lew.

2 Related Work

In the field of monocular depth estimation approaches differ in whether the model is supervised or unsupervised. When a model is supervised it is trained using RGB images annotated with depth information for each pixel. Unsupervised models fall in two categories, stereo-based and video-based. Stereo-based unsupervised models train on a dataset of stereo RGB images without annotated depth. Video-based unsupervised models train on images sequences from videos. Unsupervised models are useful as they do not need vast amounts of unlabelled data. They are also not susceptible to noise in depth sensors like LIDAR that are used to create the annotated depth maps.

2.1 Supervised monocular depth estimation

Using annotated RGBD images A. Saxena et al. [1] were the first to apply a machine learning model to supervised monocular depth estimation. They used a Markov random field to get a functional mapping from visual clues to depth. Later they extended this to a patch based model that assumes the image is made up of small patches. An MRF learns the 3D orientation and 3D location of these patches. Eigen et al. [6] applied a new multi-scale convolutional network architecture to the problem. By progressively refining predictions using a sequence of scalars, they captured many image details without the use of any superpixels or low-level segmentation. A new architecture called ResNet [7] created significant progress in convolutional networks by giving the network a good understanding of contextual and structural information in a scene. This architecture is built upon by many following works. The main hurdle that followed is decreased spatial resolution caused by repeated pooling in the deep feature extractors. Xie et al. [8] applied skip connections from deeper layers with lower spatial resolution to higher spatial resolution layers. A recent approach by Fu et al. [9] changed the problem from regression to ordinal regression. By training the network using ordinary regression loss they achieved much higher accuracy and faster convergence. Lee et al. [10] used novel local planar guidance layers located at multiple stages of the decoding phase for more effective guidance of densely encoded features to desired depth prediction. Most previous methods only use pixel-wise depth supervision to train a network. Liu et al. [11] does this differently. By combining a deep convolutional network with a continuous conditional random field (CRF) they exploit consistent information of neighbouring pixels. CRF establishes a pair-wise constraint for local regions instead of only a pixel-wise constraint. Yin et al. [12] show the importance of using higher-order 3D geometric constraints by using a loss term that enforced virtual normal directions determined my 3 randomly sampled points in 3D space. This made the 3D geometry much more consistent.

2.2 Stereo-based unsupervised monocular depth estimation

Recently, unsupervised methods using stereo images have been introduced. They attempt to recover a right view using a left view and then define the error between both as the reconstruction loss for the main training objective. Godard et al. [13] show that using only this error does not produce accurate depth maps and introduce a novel training loss that enforces consistency between the disparities produced relative to both the left and right image. This improves performance and robustness. Zhan et al. [14] show deep feature-based warping loss improves upon simple photometric warp loss.

2.3 Video based unsupervised monocular depth estimation

Previous works used separate pose and depth convolutional neural networks by minimizing the photometric consistency across monocular video datasets during training [15]. Wang et al. [16] showed that separate depth and camera pose estimators are not necessary. Inspired by recent advances in Direct Visual Odometry (DVO), they incorporated a differentiable implementation of DVO along with a novel depth normalization strategy to substantially improve performance over previous methods. Bian et al. [17] analyzed the effects of camera motion on current video based unsupervised frameworks, and revealed that the camera rotation behaves as noise for training depth CNNs, while the translation has a positive contribution. This lead to unsupervised state-of-the-art performance getting closer to the supervised state-of-the-art performance.

2.4 Ensemble models

Ensemble learning is a technique where multiple different models combine their output to increase performance. This technique has been shown to be very successful with deep CNNs as these models often have high variance and low bias [18]. Most often this technique is used with classifiers where the models vote on which class they think is correct. The diversity between the models is most important to improve performance when ensembled. [5]. Ensemble learning is widely used to create state-of-the-art models. In 2012, 5 structurally identical but differently trained versions of AlexNet won the image classification challenge ILSVRC [19]. An ensemble of six ResNet [7] models won ILSVRC in 2015. These ensemble methods rely on the sheer number of models in the ensemble and none of these methods have yet to try search for the best models to complement each other. Aakenberg et al. [20] researched the best models to form different ensembles of DCNNs for RGB-D object recognition classification. They found that the recognition performance of an existing RGB-D object recognition model can be significantly increased by forming an ensemble of two generalist models and an expert model.

3 Combined Models

3.1 Model Selection

To be able to evaluate the performance of fused models we first need to decide which models to combine. The selection process has three deciding factors. First the performance of the model and second whether the model had the code of an implementation available. The last deciding factor is the degree of difference in methods between the models. The more different the models are the higher the chance that combing them will lead to improved performance. Two of the chosen models are the highest performing of the papers with available code. The number one highest performing model is Big To Small (BTS) [10]. The second is Virtual Normal Loss (VNL) [12]. The third highest performing method according to the paper is SharpNet [21]. This paper does have an implementation available but is only trained using NYU Depth-v2 and not KITTI. The different datasets the models are evaluated on are further explained in section 5.1 and 5.2. VNL and SharpNet have quite similar architectures while BTS is very different. VNL and SharpNet use a geometric constraint to improve performance while BTS uses improved network architecture. A large difference in techniques will likely improve the chance of the two methods combining well.

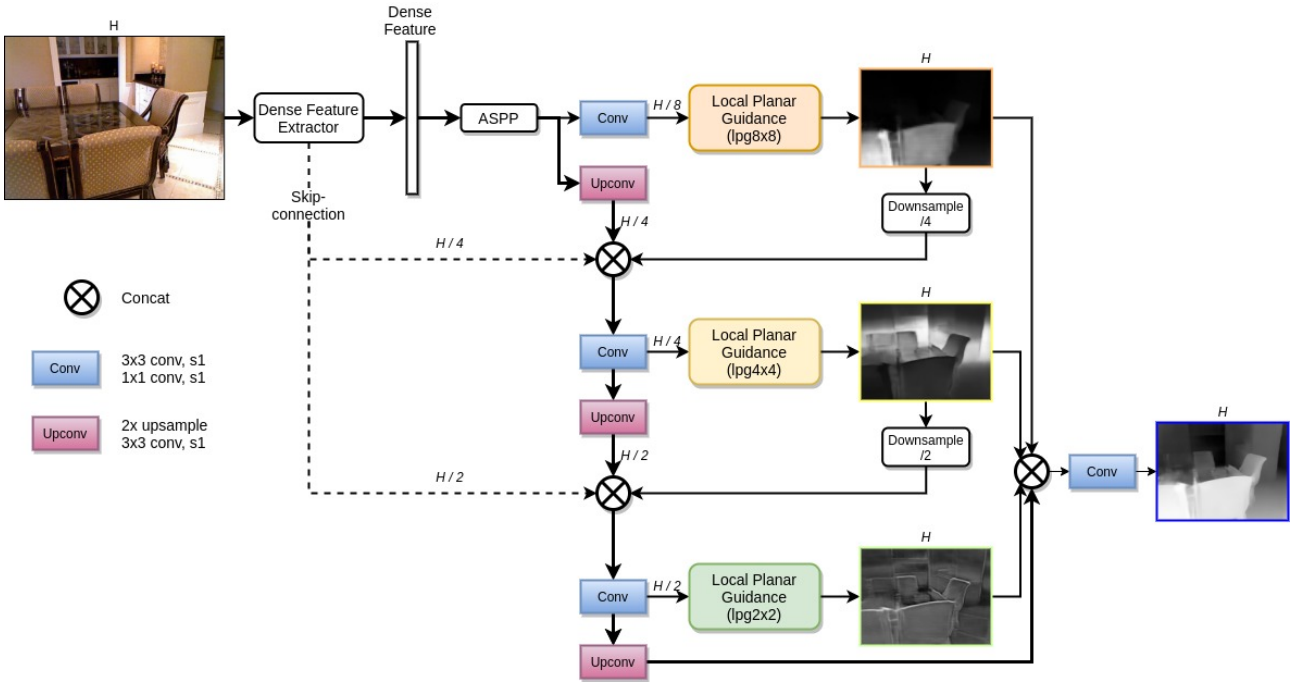

Figure 2: The BTS Architecture.

3.2 BTS

The BTS architecture consists of an encoder decoder convolutional network. The encoder uses a dense feature extractor (DFE) and a contextual feature extractor (ASPP). These extractors attempt to learn how different features in a scene correspond to different depths using repeated convolutions and pooling operations. The decoder takes these features and performs repeated

upconvolutions on them. This then leads to predicting the desired depth. However, during decoding a lot of resolution and sharpness from the input is lost. Many techniques like skip connections and multi-layer deconvolutional networks have attempted to solve this.

BTS uses a new technique called local planar guidance layers to address this problem. These are layers that store information from earlier decoding stages, which are used when decoding to define a more direct and implicit relation between internal features and the final output. BTS combines this new technique with well researched skip connections to lose as little resolution as possible. Figure 2 shows the BTS architecture and the placement of the local planar guidance layers and skip connections.

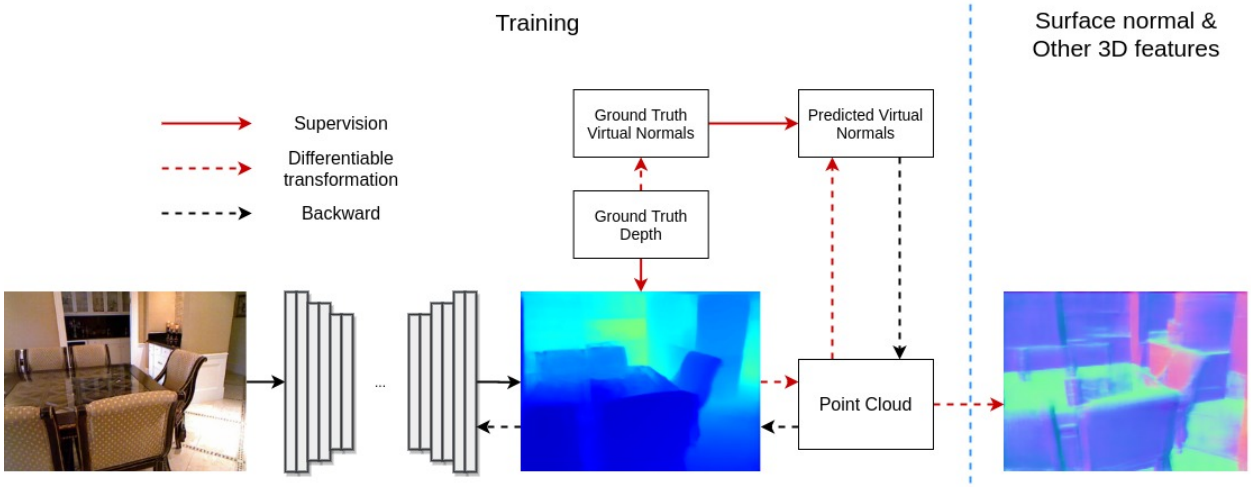

Figure 3: The VNL Architecture.

3.3 VNL

The VNL architecture starts with an encoder decoder network. This network is a state-of-the-art architecture called ResNeXt-101 [22]. It was pretrained on the ImageNet dataset [23]. This network predicts the depth. From the depth a point cloud is reconstructed. This point cloud is used to predict virtual normals which are then constrained in 3D space. VNL uses ground truth depth and ground truth virtual normals to train the network. The ground truth virtual normals are generated using differentiable transformation from the ground truth depth map. The ground truth depth is used to train the ResNeXt network and the ground truth virtual normals are used for virtual normal constraints. These constraints are used to make the point cloud more accurate. This is then used to force the depth prediction to follow the realisitic 3D geometry which is used to improve the depth prediction. Figure 3 shows the VNL architecture. By using 3D constraints VNL manages to significantly boost the base networks performance.

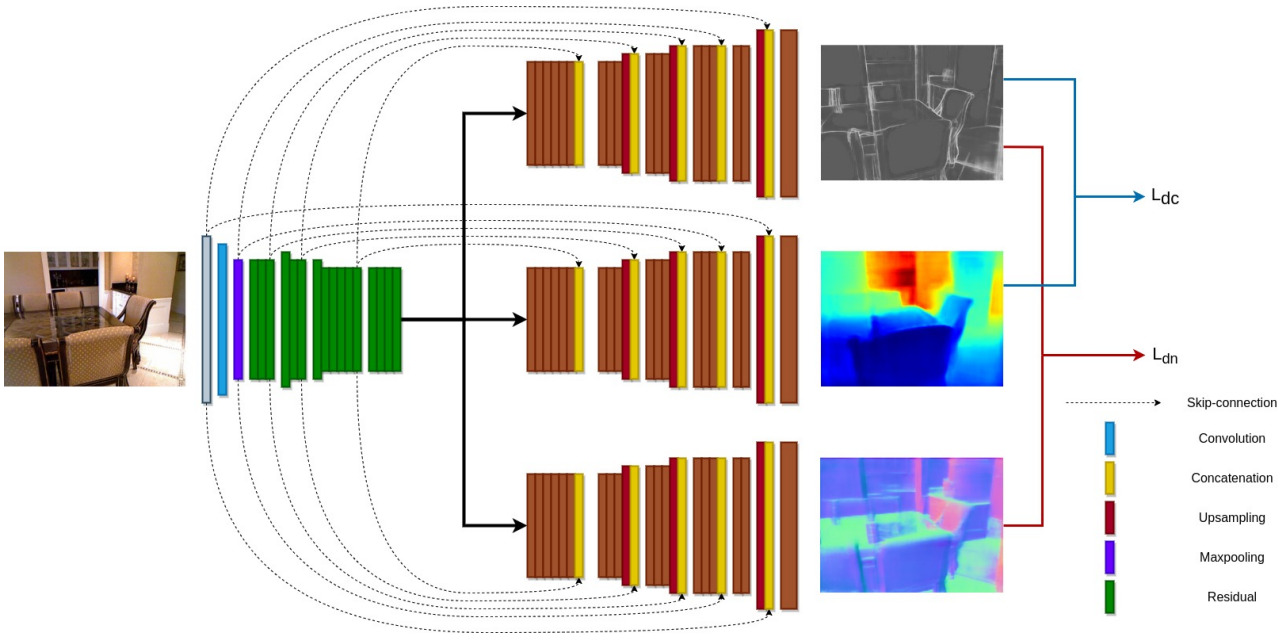

Figure 4: The SharpNet Architecture.

3.4 SharpNet

The SharpNet architecture is also based on an encoder decoder network. However, SharpNet uses the ResNet50 encoder [7]. Their architecture is called U-shaped as the encoded intermediate representation is used by three different decoders, each decoding for a different task; occluding contours, depth and surface normals. As seen in Figure 4, the network utilizes skip connections from the decoder to the encoders. The network was designed to create a depth map with very sharp corners and thus accurate occluding contours. This is why the network is split up into three different decoders. These decoders need a way to influence the other decoders. Otherwise they could never increase the accuracy of the depth prediction. This is done by using a loss function that when training forces the network to estimate very accurate depth using the occluding contours and surface normals. These loss functions are shown in Figure 4 as L_{dc} and L_{dn}.

3.5 Differences

VNL and SharpNet have very similar approaches to improving depth estimation performance. SharpNet trains additional decoders to create more output data. This is used to improve the performance of the depth estimation by enforcing a loss function while training. VNL does not train multiple decoders but does create more data in the form of normal vectors, which are constrained to improve the consistency of geometry. This is then enforced on the depth estimation with a loss function in the same way SharpNet does. BTS is significantly different and only improves the architecture of the encoder decoder network by adding local planar guidance layers. Since BTS has the highest performance and is the most different from the other models it is expected to be the best performer when fused with another model.

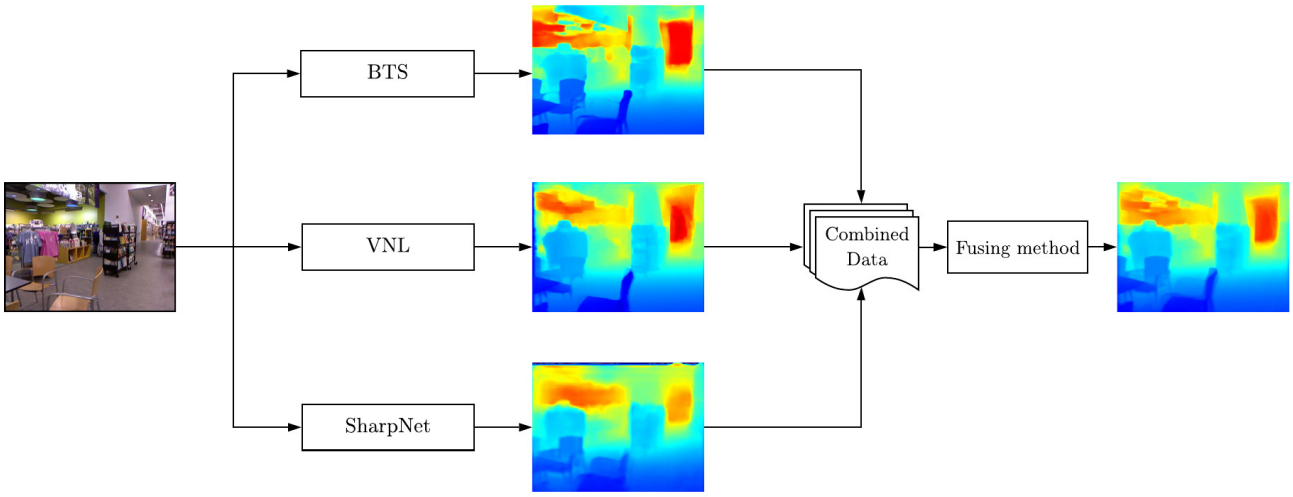

Figure 5: The steps needed to process an input image.

4 Method

Combining models takes several steps. The first is to take an instance from the dataset and hand it to all the used models. Each model will then process the instance and output the depth for each pixel. These depths will be combined into a new set of data. After which this new dataset will be fused using different fusion methods. Figure 4 shows these steps.
We will cover the three most common fusion methods. The most common method being the average. [18]. Second an third are the weighted average and median. These are the formulas for each of the methods:

$$\text{Average} : \frac{1}{|M|}\sum_{k\in M} k$$

M is a collection of models. k is the output for a single model.

$$\text{Weighted Average} : \frac{1}{\sum_{i=0}^{|M|}\alpha_i}\sum_{i=0}^{|M|}\alpha_i k_i$$

α_i is the weight of the i-th model, k_i is the output of the i-th model.

$$\text{Median} : \frac{1}{2}(p_{\lfloor (n+1)/2 \rfloor} + p_{\lceil (n+1)/2 \rceil})$$

Where $n = |M|$. p is a pixel from the model outputs. The pixel p_i denotes the same pixel from the output of model i. $\lfloor\ \rfloor$ and $\lceil\ \rceil$ are the floor and ceiling functions, respectively.

4.1 Finding the weighted average weights

To be able to calculate the weighted average we first need to find the optimal weights for each model. For classification tasks it is possible to calculate these weights by using the accuracy of each model [24]. This is used to estimate how useful each model is and thus how much it would add to the weighted average. This method does not work for the pixel based weighted averaging. This is why a simple optimization algorithm was used.

When optimizing the weights it is important to not use the same data to optimize the weights and to evaluated the performance of the weighted average. Both datasets already have been split into a train and test set. The train set was used to train the base models and the test set was used to evaluate their performance. To optimize the weights we cannot use the train set as the models have already seen the data. This is why the test set is split randomly into two equally sized portions. The first half will be used to optimize the weights and the second half to evaluate the performance of the optimized weights. The dataset specific splits will be discussed in Section 5.1 for NYU Depth-v2 and Section 5.2 for KITTI.

Now that we know what data to use for the optimization we need to decide what metric to optimize. The evaluation metrics are explained in section 5.3. Any of these metrics can be optimized but for this paper the RMSE was chosen as this metric is most used to compare different models. To start the optimization, all the weights for the base models are first initialized to 1.0. The RMSE is then minimized using the Nelder-Mead simplex algorithm [25] implemented by Sci-Py. The resulting weights are used in the Section 5.

5 Experiments

This section will show how effective different fusion methods are. This will be done by evaluating the fusion of these models on two datasets. We will first go over these datasets. Then the evaluation metrics will be discussed, after which the fused models will be evaluated using these metrics. And different combinations of models will be compared.

5.1 NYU Depth-v2 Dataset

The NYU Depth-v2 dataset [4] contains RGB images with annotated depth. These images are of size 480 X 640 captured using a Microsoft Kinect. The dataset captured 464 diverse indoor scenes of the New York University. NYU Depth-v2 provides 120K image-depth pairs. But not all pairs are useful due to asynchronous capture rates between the RGB and Depth camera. VNL randomly sampled 29K pairs to use for training. While BTS use a regular time-step between the samples to retrieve 24K image-depth pairs. Other works have pre-processed the dataset to be able to use it in its entirety to train their network [9]. SharpNet does not explicitly state what part of the dataset they use for their depth estimation training. Their occluding contour decoder uses a new dataset created by the SharpNet authors by hand-annotating the NYU Depth-v2 dataset which they called NYUv2-OC.

The NYU Depth-v2 dataset also contains 1449 selected RGBD images for training and testing. These images have the highest quality RGB and Depth information. 795 are split of for training and 654 are split for testing. Both BTS and VNL used the 654 testing images to evaluate their performance. SharpNet does not specify which images are used. In this paper testing images will be randomly split in two as specified in Section 4.1. The first half will be used to optimizing the weights for the weighted average and the second half will be used for the evaluations.

When evaluating the NYU Depth-v2 dataset BTS uses a center crop as specified in Eigen et al. [6]. This is in accordance with previous works [9]. VNL and SharpNet do not apply the Eigen crop before evaluation. We will use the Eigen crop to use as close an evaluation method to previous works as possible. This is the reason that our evaluation of VNL and SharpNet will not be identical to the values provided in their paper.

5.2 KITTI dataset

The KITTI dataset [3] is captured using LIDAR placed on top of a VW station wagon. This LIDAR data is combined with RGB images taken at the same time to provide a dataset of 61 different outdoor scenes. Eigen et al. [6] proposed a split across these different scenes. Many previous works use this split. The test split is made up of 697 images spanning 29 scenes. The train split contains the other 32 scenes containing 32488 images. Both BTS and VNL used the train images to train their model. The test images will be randomly split into two as specified in Section 4.1.

Garg et al. [26] proposed a center crop for the KITTI dataset. BTS uses this crop in accordance with previous works. VNL does not use this crop. In this paper we will use the Garg crop to keep our evaluation as close to other works as possible.

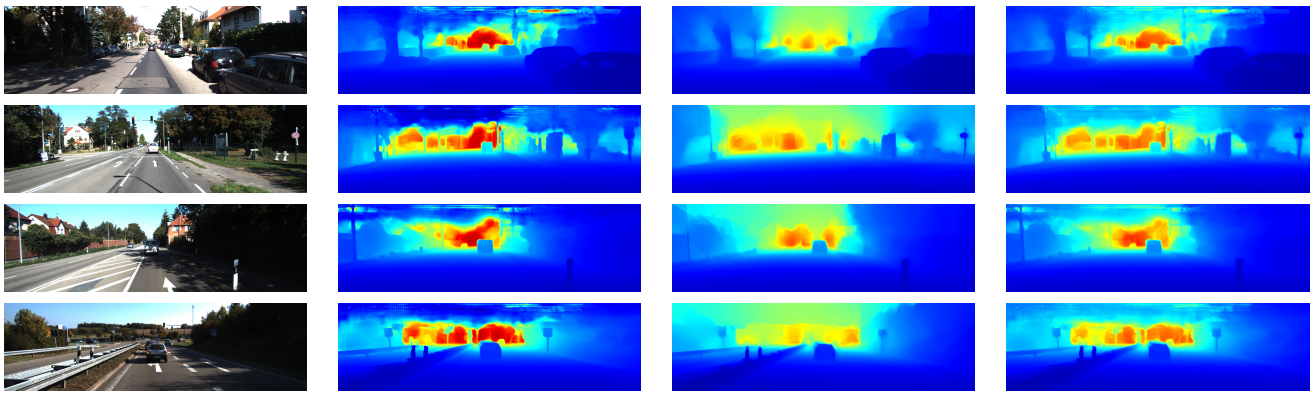

Figure 6: The estimated depth from two base models and our fused model using the average fusion method. On four images from the KITTI dataset.

5.3 Evaluation

The exact same evaluation method used by [10] will be used in this paper. This is consistent with previous monocular depth estimation works. Both datasets will be evaluated using the following metrics.

$$\text{Threshold} \ : \ \% \text{ of } y_i \text{ s.t. } \texttt{max}\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \delta < thr$$

$$\text{Abs Rel} \ : \ \frac{1}{|T|}\sum_{y \in T} \left|y - y^*\right| / y^*$$

$$\text{Sq Rel} \ : \ \frac{1}{|T|}\sum_{y \in T} \left\|y - y^*\right\|^2 / y^*$$

$$\text{RMSE} \ : \ \sqrt{\frac{1}{|T|}\sum_{y \in T} \left\|y - y^*\right\|^2}$$

$$\text{RMSE}log \ : \ \sqrt{\frac{1}{|T|}\sum_{y \in T} \left\|\log y - \log y^*\right\|^2}$$

$$\text{log10} \ : \ \frac{1}{|T|}\sum_{y \in T} \left|\log_{10} y - \log_{10} y^*\right|^2$$

T denotes a collection of pixels for which the ground truth values are available. y is the estimation value and y^* the ground truth value.

Method	*higher is better*			*lower is better*				
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	$\text{RMSE}_{\log}$	log10
VNL [12]	0.867	0.973	0.994	0.116	0.070	0.406	0.152	0.050
SharpNet [21]	0.868	0.977	0.995	0.125	0.072	0.402	0.154	0.050
BTS [10]	0.885	0.978	0.994	0.110	0.066	0.392	0.142	0.047
Fusion-median (ours)	0.904	0.984	0.997	0.101	0.053	0.353	0.131	0.043
Fusion-avg (ours)	0.910	0.987	0.997	0.098	0.048	0.334	0.126	0.041
Fusion-w-avg (ours)	**0.911**	**0.987**	**0.997**	**0.098**	**0.048**	**0.332**	**0.126**	**0.041**

Table 1: NYU Depth-v2 comparison. The fused methods consist of all three models: BTS, VNL and SharpNet. The weights for the weighted average are 0.96, 0.83 and 1.20 respectively. The weights were calculated using the first test split specified in Section 4.1

5.4 NYU Depth-v2 compared

Table 1 shows how much the ensemble models improved on both base models. All metrics clearly show that an ensemble will improve the performance of the base models. The δ thresholds show that the accuracy significantly increased for the ensembles. This tells us that the base models are significantly different and produce a different collection of inliers. Combining these collections creates a larger collection of inliers leading to higher δ values. It might also be that the prediction of the models are often on the other side of the GT. For example VNL might often overshoot depth and BTS might often undershoot. The correct depth is then the average of the models. This is also visible in the second row of Figure 1. BTS undershoots the depth of the back wall while VNL is closer but overshoots the depth. The depths averaged out are then closer to the actual GT depth.

The weighted average compared to the normal average did not show a significant difference on the NYU Depth-v2 dataset. Weighted average performed slightly better on all metrics but not by a large margin. This is likely due to the fact that the performance of the three models is already close. If the models used are more different in performance then weighing the average will be more important.

The weights of the base models in the Fusion-w-avg ensemble model are 0.96 for BTS, 0.83 for VNL and 1.20 for SharpNet. Even though BTS clearly has the best single model performance the optimizer still decided on higher weight for SharpNet than BTS. This can be caused by the models having slightly different performance on the first half of the test split than the second half. SharpNet may be better performing on the first while BTS performs better on the second half. Another reason could be that the performance of a model does not say how well a model will fuse with other models. This is why optimization of the weights is a better idea than choosing weights only based on the evaluation metrics.

5.4.1 Two model fusions

Table 2 shows how different two model fusions affect the average and weighted average performance. The performance of the weighted average is clearly not always better for these fusions. When looking at the weights assigned to the base models they are very close to 1.0. The Fusion of VNL

Method		*higher is better*			*lower is better*				
		$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	$\text{RMSE}_{\log}$	log10
BTS x VNL	avg	0.892	0.981	0.995	0.106	0.057	0.368	0.137	0.046
(1.08, 0.90)	w-avg	0.893	0.981	0.995	0.105	0.057	0.367	0.137	0.046
VNL x SN	avg	**0.907**	**0.985**	0.996	0.102	**0.051**	0.344	0.132	0.042
(0.99, 1.01)	w-avg	0.907	0.985	0.996	0.103	0.051	0.344	0.132	0.042
BTS x SN	avg	0.906	0.984	0.997	0.102	0.052	0.340	0.129	0.042
(1.05, 0.95)	w-avg	0.906	0.984	**0.997**	**0.101**	0.052	**0.340**	**0.129**	**0.042**

Table 2: NYU Depth-v2 weighted average of various model combinations. The first column shows the fused models and in parentheses their respective weight (α_i). SN stands for SharpNet

and SharpNet have the weights that are the closest to 1.0. Telling us that the fusion of these models does not benefit from a weighted average. The performance of the average for this model is also better than its weighted average which means the optimized weights on the first test split did not correctly model the weights needed for increased performance on the second half of the test split. The fusion of BTS and VNL did see slight improvement for the weighted average. The weights are also further away from 1.0 with 1.08 for BTS and 0.90 for VNL.

Fusing BTS and SharpNet gave the overall best performance for the two model fusions, with the best performance in almost every metric. The performance of this two model fusion comes close to the performance of the weighted average fusion of all three models, with an RMSE of 0.340 and 0.332 respectively. This shows that using a well chosen two model fusion might suit certain use cases more than using a three model fusion. If the goal is the absolute best performance it will be better to fuse all three models. Theoretically using more than three model fusions will give even better performance. But when looking at the results in Table 1 and 2, these fusions will quickly give diminishing returns when more models are added and will likely not be worth the processing power requirements.

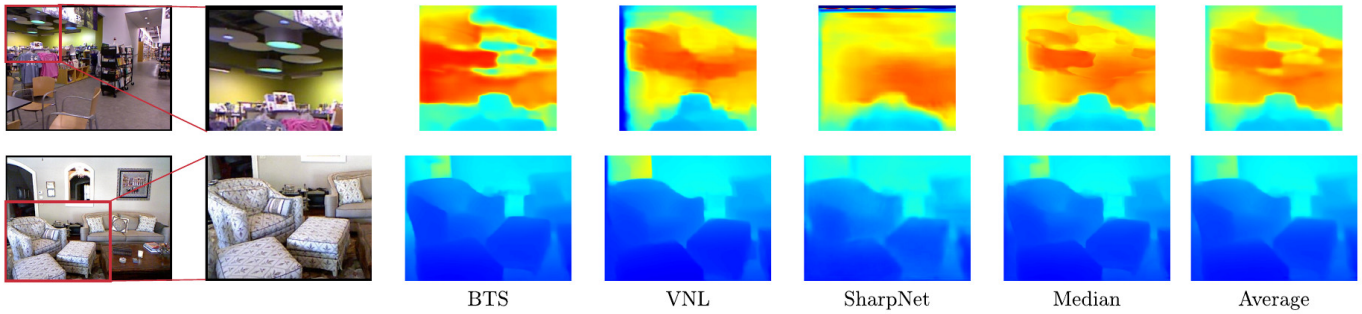

Figure 7: A comparison of the three base models and two fusion methods; the median and average. Two images from the NYU Depthv2 dataset have been cropped on an interesting region.

5.4.2 Median vs. Average

Figure 7 shows the difference between the average and median. The median takes the pixel values of one model if the value falls in between the value of the other two models. This causes it to sometimes copy artifacts from one model. This is shown in the first row of Figure 7. The BTS model shows significant difference in depth at the top of the image while the other two models output more uniform depth. The median shows clearly the BTS artifact on the top left of the image. While the average smooths out the artifact by averaging it with VNL and SharpNet.

Because the average uses all of the models to influence one pixel this often creates a less defined edge. This is visible in the second row of Figure 7. Here the edges of the furniture are clearly sharper in the median depth image. The edges of the average depth image are much more feathered. The edges in the VNL model are clearly the best defined. If the median chooses the VNL depth values the edges will be well defined. If certain SharpNet pixels happen to be the median the edges will be less well defined. This is the reason that the edges are not always perfectly sharp using the median.

The performance if the median is significantly worse than the average. With a RMSE of 0.353 and 0.336 respectively. In all of the metrics the median performs worse except the $\delta < 1.25^3$. Here the median produces the same portion of inliers as the average. The median should only be used if the sharpness of the edges is important. When all of the models combine already produce sharp edges the average will always perform better.

Method	*higher is better*			*lower is better*				
	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	RMSE log	log10
VNL [12]	0.937	0.989	0.997	0.075	0.328	3.237	0.115	0.033
BTS [10]	0.955	0.993	0.998	0.060	0.249	**2.798**	0.096	0.027
Fusion-avg (ours)	0.953	0.993	0.999	0.061	0.248	2.872	0.098	0.027
Fusion-w-avg (ours)	**0.956**	**0.993**	**0.999**	**0.058**	**0.238**	2.804	**0.095**	**0.026**

Table 3: KITTI comparison. The fused methods consist of all both models: BTS and VNL. The weights for the weighted average are 1.55 and 0.42. The weights were calculated using the first test split specified in Section 4.1

5.5 KITTI compared

Only two of the three models were trained on the KITTI dataset. This is why not every fusion method could be used. As shown in Table 3, on this dataset BTS performs significantly better than VNL on every metric. When fusing the methods using a simple average not every metric performs better. Only Sq Rel and $\delta < 1.25^3$ have better performance. This is due to the fact that BTS performs better than VNL on the KITTI dataset. This is shown in the weights for the weighted average. With 1.55 for BTS and 0.42 for VNL the weighted average gives BTS significantly more weight than VNL. This gives the weighted average better performance than BTS on every metric except the RMSE. Here the BTS base model has better performance than the weighted

average. Figure 8 shows two example images of the KITTI dataset and the results of BTS, VNL and both fusion methods. As expected the resulting outputs from BTS are much sharper. Especially around the cars in the first column. The edges from VNL look less defined. VNL also seems to underestimate depth in the first column image. BTS shows the distance to be quite far, shown by the image being more red. The weighted average weights also seem to reflect this distinction. The weighted average uses mostly the BTS depth and shows more red in the center. Looking at the fusion metrics this is much closer to the correct depth than the underestimation of VNL.

BTS shows more artifacts in the depth predictions, this is visible in Figure 8. The VNL images are much more consistent. This is likely caused by VNL using geometric constraints. These kinds of inconsistent artifacts will be filtered out by VNLs constraints.

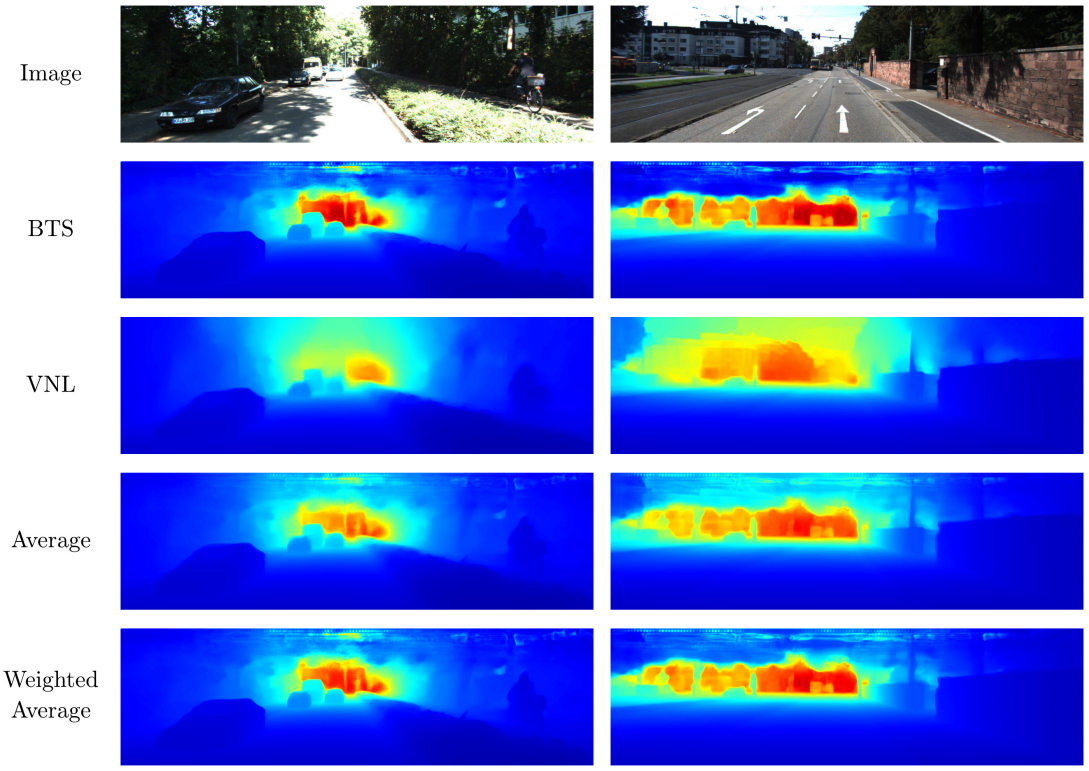

Figure 8: The estimated depth of the two base models and two fusion methods; the average and weighted average on two example images from the KITTI dataset.

6 Conclusions and Further Research

In this thesis we first introduced monocular depth estimation and provided an overview of previous work in this field. Then we decided on the different base networks to combine. We showed how they work and what differences there are in their architectures, after which the base models were combined using various different fusion methods. The fused models were evaluated using the same evaluation methods and datasets as previous works. We showed that when fusing models for monocular depth estimation it is important that the models are diverse. More diverse models yield better performance when fused. We also showed that using as many models as available for ensembles will increase performance but will quickly lead to diminishing returns. If low computation time and lower complexity are important it will often be better to find only a few models that fuse well together.

We used a few different fusion methods. Of these methods the weighted average is clearly superior but might not be necessary if the performance of the base models is already quite similar. The median consistently under performs compared to the weighted average. However it is sometimes possible for the median to produce sharper edges. The weighted average can produce less sharp feathered edges if one of it's base models does not output clear edges. The fused models performed better on the NYU Depth-v2 dataset than on the KITTI dataset. This is likely due to all available base models having similar performance on NYU Depth-v2 but one of the two available models for KITTI performing significantly worse than the other model. If the models had more similar performance on KITTI the fused performance would likely have improved significantly more.

Further research could look deeper into different fusion methods. This thesis focused on the most important methods but some more advanced methods could certainly increase performance. For example stacking: With stacking a new network is trained which takes as input the output from the base networks. This network would then learn the best way to fuse the base networks. A different fusion method could be a decision matrix. For each pixel in the input image a value will be stored in an matrix that would say which network will decide its depth. The values could either be integers choosing a certain network so that each pixel has only a single network influencing it or the values could be floats. This would work the same as the weighted average but with different weights for each pixel.

This thesis put more focus on finding a few well combining networks to increase performance. But if the goal was to find the absolute best performance, further research could look into whether inputting slightly changed images into the same network and fusing the outputs could increase performance. For example by changing the image's color balance, crop, rotation or sharpness.

References

[1] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems 18* (Y. Weiss, B. Schölkopf, and J. C. Platt, eds.), pp. 1161–1168, MIT Press, 2006.

[2] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 730–738, Curran Associates, Inc., 2016.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," in *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, Sept. 2013.

[4] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision, ECCV 2012 - 12th European Conference on Computer Vision, Proceedings*, pp. 746–760, Oct. 2012.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[6] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2366–2374, Curran Associates, Inc., 2014.

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

[8] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), pp. 842–857, Springer International Publishing, 2016.

[9] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011, 2018.

[10] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation," *arXiv e-prints*, p. arXiv:1907.10326, 2019.

[11] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, Feb. 2015.

[12] W. Yin, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5683–5692, 2019.

[13] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611, 2017.

[14] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 340–349, 2018.

[15] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619, 2017.

[16] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning depth from monocular videos using direct methods," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2022–2030, 2018.

[17] J.-W. Bian, H. Zhan, N. Wang, T.-J. Chin, C. Shen, and I. Reid, "Unsupervised Depth Learning in Challenging Indoor Video: Weak Rectification to Rescue," *arXiv e-prints*, p. arXiv:2006.02708, 2020.

[18] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, pp. 1–15, Springer Berlin Heidelberg, 2000.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems*, vol. 25, Jan. 2012.

[20] A. Aakerberg, K. Nasrollahi, and T. Heder, "Improving a deep learning based rgb-d object recognition model by ensemble learning," in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–6, 2017.

[21] M. Ramamonjisoa and V. Lepetit, "Sharpnet: Fast and accurate recovery of occluding contours in monocular depth estimation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2109–2118, 2019.

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017.

[23] J. Deng, R. Socher, L. Fei-Fei, W. Dong, K. Li, and L.-J. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 248–255, June 2009.

[24] A. Aakerberg, K. Nasrollahi, and T. Heder, "Improving a deep learning based rgb-d object recognition model by ensemble learning," in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–6, 2017.

[25] F. Gao and L. Han, "Implementing the nelder-mead simplex algorithm with adaptive parameters," in *Computational Optimization and Applications*, vol. 51, pp. 259–277, May 2012.

[26] R. Garg, V. K. B.G., G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), pp. 740–756, Springer International Publishing, 2016.