From Conventional Deep Neural Network to Semi-Parallel Deep Neural Network on Depth from Monocular Images

Shabab Bazrafkan[1], *Student, IEEE*, Hossein Javidnia[1], *Student, IEEE*, Peter Corcoran[2], *Fellow, IEEE*

I.Abstract

Computing pixel depth values ~~is a crucial way of~~provides a basis for understanding the 3D geometrical structure of an image. As it has been presented in the recent researches in this context, using stereo images provides an accurate depth due to the advantage of having local correspondences but the processing time of these methods are still an open issue. To solve this problem, it has been suggested to use single images to compute the depth values but extracting depth from monocular images requires ~~considerable~~ extracting a large number~~amount~~ of cues from the global and local information in the image. This challenge has been studied for a decade and it is still an open challenge. Recently the idea of using neural networks to solve this problem has attracted ~~a lot of~~ attention. In this paper we tackle this challenge by employing Deep Neural Network (DNN) equipped with semantic pixel-wise segmentation and utilising our recent disparity post-processing method. Four models are trained in this study and they have been evaluated in 2 ~~parts~~ stages on the KITTI dataset. The ground truth images in the first part of the experiment come from the benchmark and for the second part the ground truth images are considered to be the disparity results from applying a state-of-art ~~the~~ stereo matching method. The evaluation results ~~represent~~ demonstrate that using ~~the~~ post-processing technique~~s~~ to refine the target of the network increase~~d~~ the accuracy of ~~the~~ depth estimation on individual mono images. The ~~other~~ second evaluation shows that using the segmentation data as the input can ~~just~~ improve the depth estimation results to a point where performance is comparable with stereo depth matching ~~marginally~~.

~~1)~~II. Introduction

~~a)~~A. Depth Map

Deriving the 3D structure of an object from a set of 2D points is a fundamental problem in computer vision. Most of these conversions from 2D to 3D space are based on the depth values computed for each 2D point. In depth map each pixel is defined as the distance between an object visible in the scene and the camera, rather than a colour.

In general depth computation methods are divided into 2 categories:

1- Active methods
2- Passive methods

Active methods are computing the depth in the scene by interacting with the objects and the environment. There are different types of active methods such as Light based depth estimation which uses the active light illumination to estimate the distance of different objects. Ultrasounds based methods is another example of the active method which is quite similar to time of flight (ToF) method that uses the known speed of light to measure the time an emitted pulse of light takes to arrive to an image sensor.

Passive methods are based on the optical features of the captured images. The depth information is being extracted by computational image processing.

The research work presented here was funded under the Strategic Partnership Program of Science Foundation Ireland (SFI) and co-funded by SFI and FotoNation Ltd. Project ID: 13/SPP/I2868 on "*Next Generation Imaging for Smartphone and Embedded Platforms*".

[1] S. Bazrafkan is with the Department of Electronic Engineering, College of Engineering, National University of Ireland, Galway, University Road, Galway, Ireland. (e-mail: s.bazrafkan1@nuigalway.ie).

[1] H. Javidnia is with the Department of Electronic Engineering, College of Engineering, National University of Ireland, Galway, University Road, Galway, Ireland. (e-mail: h.javidnia1@nuigalway.ie).

[2] P. Corcoran is with the Department of Electronic Engineering, College of Engineering, National University of Ireland, Galway, University Road, Galway, Ireland. (e-mail: peter.corcoran@nuigalway.ie).

Style Definition: Normal: Space After: 6 pt

Style Definition: Heading 2: Space Before: 6 pt, Outline numbered + Level: 2 + Numbering Style: A, B, C, … + Start at: 1 + Alignment: Left + Aligned at: 2.22 cm + Indent at: 2.22 cm

Commented [CP1]: Suggest: "Depth from Monocular Images using a Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture."

Formatted: Heading 1, Left

Formatted: Heading 1, Left, Space Before: 0 pt, No bullets or numbering

Formatted: Heading 2, Left, No bullets or numbering

Generally in the category of active methods there are 2 former techniques: a) Multi-view depth estimation such as depth from stereo and b) Monocular depth estimation.

~~b)~~B. Stereo Vision Depth

Stereo matching algorithms can be used to compute depth information from multiple images. By using calibration information of the cameras, the depth images can be generated which provides useful data to identify and detect objects in the scene.

Ideally, in stereo applications the two cameras are located by a short distance and almost parallel to one another. This system is capable of acquiring two different images. Both cameras capture the same 3D point P in a different position on the 2D images. The displacement between those two 2D points is known as "Disparity" and it can be used to calculate depth information, which is the distance between 3D point P and the cameras [1]. In recent years, many applications, including time-of-flight [2, 3], structured light [4], and the Kinect were introduced to calculate depth from stereo images. Generally stereo vision algorithms are divided into 2 categories: Local and Global. Local algorithms have been introduced as some statistical methods that use the local information around a pixel to determine the depth value of the given pixel. These kinds of methods can be used for real time applications if they are implemented sufficiently. Global algorithms try to optimize an energy function to satisfy the depth estimation problem through various optimization techniques [5].

In terms of computation, global methods are more complex than local methods and they are almost impractical for real time applications but they have an important advantage; in terms of accuracy they are more accurate than local methods. This advantage recently attracted considerable attention in various researches.

A global stereo model is proposed in [6] by converting the image into a set of 2D triangles with adjacent vertices. Later, the 2D vertices are converted to a 3D mesh by computing the disparity values. To solve the problem of depth discontinuities, a two layer MRF is employed. The first layer of the MRF is assigned with a set of plane equations modelling the triangles to generate better quality stereo. The second layer is formed by the splitting probability of each vertex which is responsible for better visual rendered results. The layers are fused with an energy function allowing the method to handle the depth discontinuities. The method has been evaluated on the new Middlebury 3.0 benchmark [7] and it was ranked the first by the time of the paper's publication.

Another global stereo matching algorithm is proposed in [8] based on the texture and edge information of the image. The problem of large disparity differences in small patches in non-textured regions is tackled by using the colour intensity. Also the main matching cost produced by a CNN is augmented using the same colour based cost. Afterwards the edges that correspond to depth discontinuity are refined by combining the edge maps of the RGB image and the initial disparity map. The final results are post-processed using a 5×5 median filter and a bilateral filter. This adaptive smoothness filtering technique provides a good performance for the algorithm which made it the first in the Middlebury 3.0 benchmark [7] ~~by~~ at the time of the paper's publication.

Many other methods have been proposed for stereo depth ~~in this criterion~~ such as PMSC [7], GCSVR [7], INTS [9], MDP [10], ICSG [11] which all tried to improve the accuracy of the depth from stereo vision or bring a new method to estimate the depth from a stereo pair. ~~but~~However there is always a trade-off between accuracy and speed for stereo vision algorithms.

Table 1 shows a general comparison of the average normalized time by number of pixels (sec/megapixels) between most accurate stereo matching algorithms ranked in Middlebury 3.0 benchmark based on the "bad 2.0" metric. The ranking is on the test dense set. This comparison is a proof of the fact that "obtaining an accurate depth from a stereo pair requires ~~a high~~significant processing power". ~~This fact shows~~These results demonstrate that, today, these methods are ~~not suitable~~too resource intensive ~~enough~~ for real-time applications like street sensing or autonomous navigation due to their demand for ~~expensive~~ processing resources.

To decrease the processing power of ~~the~~ stereo matching algorithms, researchers recently ~~researchers started~~began to work on depth from monocular images. ~~which make the algorithms capable of estimating~~

Formatted: Heading 2, Left, Space Before: 0 pt, No bullets or numbering

Commented [CP2]: Not the best way to 'say' this …

Commented [CP3]: For stereo vision it is 'required' to acquire 2 images!

Commented [CP4]: Not a good way to say this …

Commented [CP5]: I'm not sure you need to cover these fundamentals; fine in a thesis chapter, but you could simply provide a good citation to cover most of the details of this paragraph …

Commented [CP6]: No citation(s)? Maybe leave it out … ?

Commented [CP7]: There has to be a better, clearer, way to explain this; maybe use a few sentences? Or maybe you don't need this much detail on this paper … ?

Commented [CP8]: "… its was ranked #1 in terms of overall accuracy and performance at the time of publication."
(#1 for what?)

Commented [CP9]: Again, not sure we need all the details here … good in a thesis chapter though!

~~the~~Such algorithms estimate depth from a single camera while keeping the processing power low. The next section ~~explains more~~gives further details about ~~the~~ current research into depth from mono images.

Table 1. Comparison of the Performance Time between the Most Accurate Stereo Matching Algorithms

Algorithm	Time/MP (s)	W × H (ndisp)	Programming IDE	Hardware
PMSC [7]	453	1500 x 1000 (<= 400)	C++	i7-6700K, 4GHz-GTX TITAN X
MeshStereoExt [6]	121	1500 x 1000 (<= 400)	C, C++	8 Cores-NVIDIA TITAN X
APAP-Stereo [7]	97.2	1500 x 1000 (<= 400)	Matlab+Mex	i7 Core 3.5GHz, 4 Cores
NTDE [8]	114	1500 x 1000 (<= 400)	n/a	i7 Core, 2.2 GHz-Geforce GTX TITAN X
MC-CNN-acrt [12]	112	1500 x 1000 (<= 400)	n/a	NVIDIA GTX TITAN Black
MC-CNN+RBS [13]	140	1500 x 1000 (<= 400)	C++	Intel(R) Xeon(R) CPU E5-1650 0, 3.20GHz, 6 Cores-32 GB RAM-NVIDIA GTX TITAN X
SNP-RSM [7]	258	1500 x 1000 (<= 400)	Matlab	i5, 4590 CPU, 3.3 GHz
MCCNN_Layout [7]	262	1500 x 1000 (<= 400)	Matlab	i7 Core, 3.5GHz
MC-CNN-fst [12]	1.26	1500 x 1000 (<= 400)	n/a	NVIDIA GTX TITAN X
LPU [7]	3523	1500 x 1000 (<= 400)	Matlab	Core i5, 4 Cores- 2xGTX 970
MDP [10]	58.5	1500 x 1000 (<= 400)	n/a	4 i7 Cores, 3.4 GHz
MeshStereo [6]	54	1500 x 1000 (<= 400)	C++	i7-2600, 3.40GHz, 8 Cores
SOU4P-net [7]	678	1500 x 1000 (<= 400)	n/a	i7 Core, 3.2GHz-GTX 980
INTS [9]	127	1500 x 1000 (<= 400)	C, C++	i7 Core, 3.2 GHz
GCSVR [7]	4731	1500 x 1000 (<= 400)	C++	i7 Core, 2.8GHz-Nvidia GTX 660Ti
JMR [7]	11.1	1500 x 1000 (<= 400)	C++	Core i7, 3.6 GHz-GTX 980
LCU [7]	9572	750 x 500 (<= 200)	Matlab, C++	1 Core Xeon CPU, E5-2690, 3.00 GHz
TMAP [14]	1796	1500 x 1000 (<= 400)	Matlab	i7 Core, 2.7GHz
SPS [7]	49.4	3000 x 2000 (<= 800)	C, C++	1 i7 Core, 2.8GHz
IDR [15]	0.36	1500 x 1000 (<= 400)	CUDA C++	NVIDIA GeForce TITAN Black

Formatted Table

Formatted Table

Formatted Table

Formatted: Normal, Left, Space Before: 0 pt, No bullets or numbering

~~c)~~C. Deep Learning

DNN is one of the most recent approaches in the pattern recognition science which is able to handle highly non-linear problems in classification and regression fields. These models use consecutive non-linear signal processing units in order to mix and re-orient their input data to give the most representative results in the output. The DNN structure learns from the input that it encounters and then it generalizes what it learns into the data samples it had never seen before [16]. The typical deep neural network model is made from several convolutional layers followed by pooling and fully connected layers accompanied by different regularization tasks. Each of these units is as follows:

Convolutional Layer: This layer convolves the (in general 3D) image "I" with (in general 4D) kernel "W" and adds a (in general 3D) bias term "b" to it. The output is given by:

$$P = I * W + b \qquad (1)$$

where * operator is nD convolution in general. In the training process the kernel and bias parameters are updated in a way to optimize the error function of the network output.

Pooling Layer: The pooling layers applies a (usually) non-linear transform (Note that the average pooling is a linear transform, but the more popular max-pooling operation is non-linear) on the input image which reduce the spatial size of the data representation after the operation. It's common to put a pooling layer between two consecutive convolutional layers. Reducing the spatial size will lead to less computational load and also prevents the over-fitting and also adds a certain amount of translation invariance to the problem.

Fully Connected Layer: Fully connected layers are exactly same as the classical Neural Network (NN) layers where all the neurons in a layer are connected to all the neurons in their subsequent layer. The neurons give the summation of their input multiplied by their weights and then passed through their activation functions.

Regularization: In general, regularization is proposed to prevent the overfitting inside the network. One can train a more complex network (more parameters) with regularization and prevent over-fitting while the same network would get over-fitted without regularization. Different kind of regularizations has been proposed. The most important ones are weight regularization, drop-out technique [17], and batch normalization [18] layer. Each of them has their own advantages and drawback which make each one more suitable for specific applications.

Formatted: Normal, Left, Space Before: 0 pt, No bullets or numbering

~~d)~~D. Monocular Vision Depth

Depth estimation from a single image is a basic problem in computer vision and has potential applications in robotic, scene understanding and 3D reconstruction. As there is no reliable cues in a single image such as temporal information or stereo correspondences, depth estimation from mono image still remains as an important challenge in computer vision.

As the result of the recent researches, deep convolutional neural networks (CNN) are setting new records for various vision applications.

~~Somehow superpixeling~~However the superpixelling technique of [19] is not a good choice to initialize the disparity estimation from mono images because of the lack of the monocular visual cues such as texture variations and gradients, defocus, color/haze in some parts of the image. To solve this issue Markov Random Field (MRF) learning algorithm has been implemented to capture some of these monocular cues [20]. The captured cues were integrated with a stereo system to obtain better depth estimation than the stereo system alone. This method is using a fusion of stereo + mono depth estimation. At small distances, the algorithm relies more on stereo vision, which is more accurate than the monocular vision. However, at larger distances, the performance of stereo degrades; and the algorithm relies more on monocular vision.

Formatted: Normal, Left, Space Before: 0 pt, No bullets or numbering

~~e)~~E. Paper~~'s~~ Overview

In this paper we ~~are~~ present~~ing~~ a hybrid DNN to estimate the depth from mono cameras based on the approach presented in [21]. The depth map from the stereo sets are estimated using the same approach as [21] and the results are used as the target to train the network while using a single image (the left image in stereo set) as input. Four models are trained and evaluated to estimate the depth from single camera images in this.

These models are designed for two different possibilities each have two different options. The target for the network could be the result of stereo depth map with and without post-processing explained in [21]. The other two possibilities are to use or not to use the semantic pixel wise labelled segmentation information to be entered as extra data to the input of the network. The technical details of each part are presented in the next section. Fig. 1 shows the general overview of the trained models in this paper.

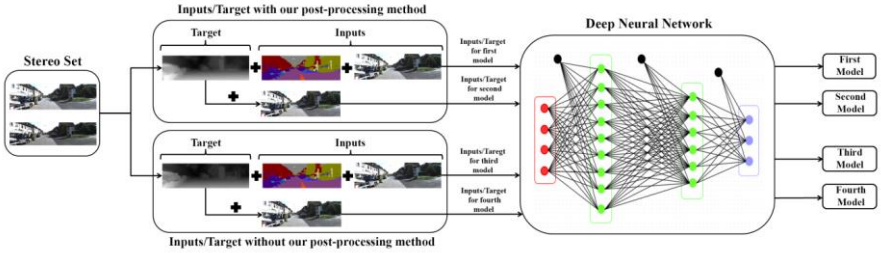


Figure 1. The overview of the trained models in this paper

~~2)~~III. Enabling Deep Learning ~~Materials~~Methodology used in this Work

~~a)~~A. A Brief Description of Our Post-Processing Method

In [21] a guided joint filter is presented which is based on the mutual information of the RGB image as reference and the estimated depth image. This approach has been used to increase the accuracy of the depth estimation in stereo vision by preserving the edges and corners in the depth map and filling the missing parts. The method was compared with top 8 depth estimation methods in Middlebury benchmark [7] ~~by~~ at the time of the paper's submission. Seven metrics were used to evaluate the performance of each method. The evaluation ranked the method as 1st in 5 metrics and 2nd and 3rd in other metrics.

The same approach is used in this paper to compute the depth from stereo set. Two of the models in this paper are using this method to compute the depth as one of the inputs of the network.

~~b)~~B. Semantic Pixel-Wise Image Segmentation

One of the principle ideas in this paper is to ~~find out~~investigate how semantic segmentation can help a neural network to better learn ~~the depth better~~ and improve ~~the~~ depth estimation in a 2D image scene. For this ~~reason in this paper~~purpose ~~we employed~~ SegNet [22] [23] is employed as the underlying DNN architecture.

SegNet is one of the most successful recent implementation of DNN for semantic pixel-wise image segmentation ~~which overtook~~and has supplanted other configurations of FCN both by accuracy and simplicity in implementation. SegNet uses the convolutional layers of the VGG16 network as the encoder of the network where by eliminating the fully connected layers of VGG16 it reduces the number of trainable parameters from 134M to just 14.7M which ~~is around~~represents a reduction of 90% ~~less~~ in the number of parameters to be trained. The encoder ~~part~~ portion of ~~the~~ SegNet consists of 13 convolutional layers with ReLU nonlinearity followed by max-pooling (2×2 window) and stride 2 in order to implement non-overlapping sliding window. These consecutive max-pooling and striding, results in a network configuration which is highly robust to translation in the input image but ~~from the other side~~on the other hand it has the drawback of losing spatial resolution of the data.

~~which~~ This loss of spatial resolution is not beneficial in segmentation tasks where one need to preserve the boundaries from the input image in the segmented output. To overcome this problem, ~~this~~ the following solution has been considered in our work: ~~Since~~ as ~~the~~ most of the spatial resolution information is lost in the max-pooling operation, saving the information of the max-pooling indices and used this information in the decoder part of the network will preserve the high frequency information of the input image from the encoder part and passes them to the decoder where the un-pooling operation is applied to the image.~~,~~

~~note~~ Note that for each layer ~~in~~ in the encoder ~~part~~ portion of the network there is a corresponding decoder layer. In ~~the~~ other words, the idea of SegNet is where the max-pooling is applied to the input data the indices of the

Commented [CP10]: An overview …
This is quite a 'long' figure; you may need to "squeeze it up" (especially the 'long' image inputs) a bit to be able to make the font sizes a wee bit larger …

Formatted: Heading 1, No bullets or numbering

Commented [CP11]: Maybe it is better to say ".. at the time of writing …" or "… at the time the paper was authored."

Commented [CP12]: Why not list them?

Formatted: Normal, Left, Space Before: 0 pt, No bullets or numbering

Commented [CP13]: FCN? Not CNN or DNN?

feature space where is maximum value is happening is saved from the 2x2 window. Therefore, for each position of the sliding window of max-pooling it just stores 2 bits to save the index of the maximum value.; afterwards Later these indices will be used employed to make a sparse feature space before the de-convolution step in each layer while applying the un-pooling step in the decoder part. A batch normalization layer [18] is placed after each convolutional layer to avoid overfitting and faster convergence. Decoder filter banks are not tied to corresponding encoder filters and are trained independently in the SegNet architecture.

The SegNet has been trained using Stochastic Gradient Descent (SGD) with learning rate 0.1 and momentum 0.9. The Caffe implementation of SegNet has been used to train the network [24]. The gray-scale CamVid road scene database (360 × 480) [25] has been used in training step

Commented [CP14]: Not a very clear sentence – try to improve it; maybe have a bit more explanation?

Commented [CP15]: Is this a general property of SegNet or is this specific to your use of this architecture/network?

3)IV. Methodology Enhancements - Proposed Approach

Formatted: Heading 1, Space Before: 0 pt, No bullets or numbering

a)A. Semi-Parallel Deep Neural Network (SPDNN)

Formatted: Heading 2, No bullets or numbering

For the first time the SPDNN idea is being introduced in this paper. In this method several deep neural networks are being parallelized to take advantage of each and all of them at the same time. In this method several neural networks are merged into one model, the final model is trained for the problem, our observations show that using this method will help to reduce overfitting and gives a better convergence.

The Merging merging of multiple several networks using SPDNN idea is next described in the context of the current depth-mapping problem. below.

B. Individual SegNet Networks for Depth Analysis

Formatted: Heading 2, Left

Write a bit about each network – what were you trying to achieve with it? What design elements are featured? Did it come from another research paper?

Formatted: Font: 9 pt, Bold, Italic

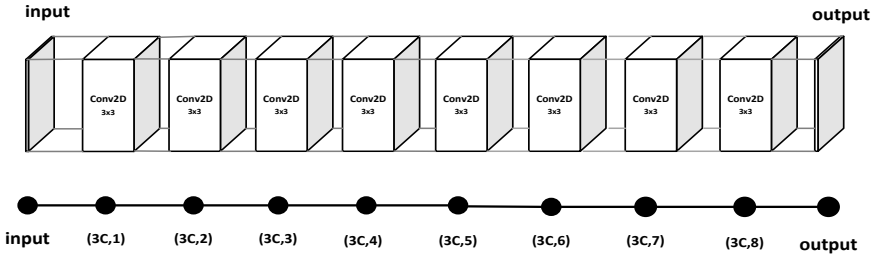


Figure 2. Top row: network 1, Bottom row: graph corresponds to network1.

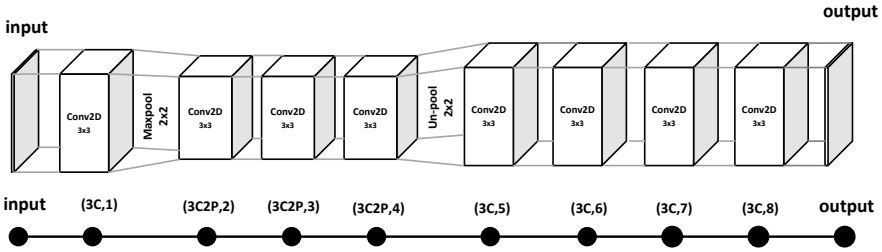


Figure 3. Top row: network 2, Bottom row: graph corresponds to network2.

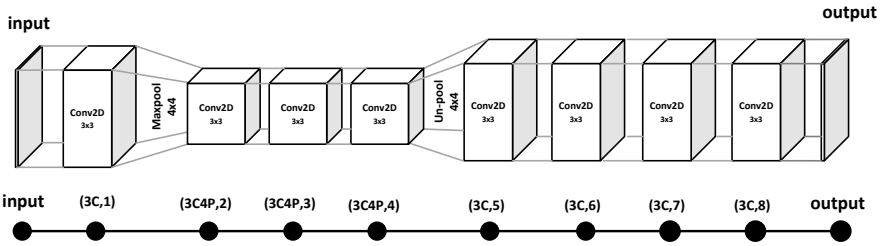


Figure 4. Top row: network 3, Bottom row: graph corresponds to network3.

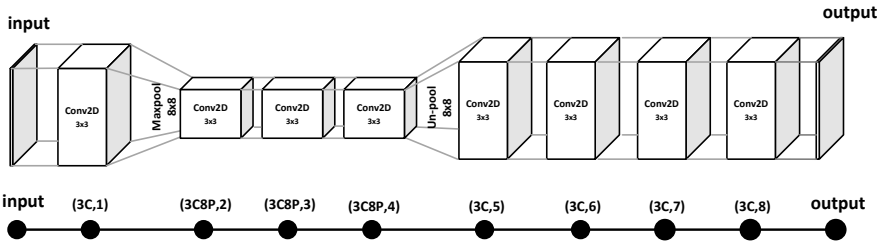


Figure 5. Top row: network 4, Bottom row: graph corresponds to network4.

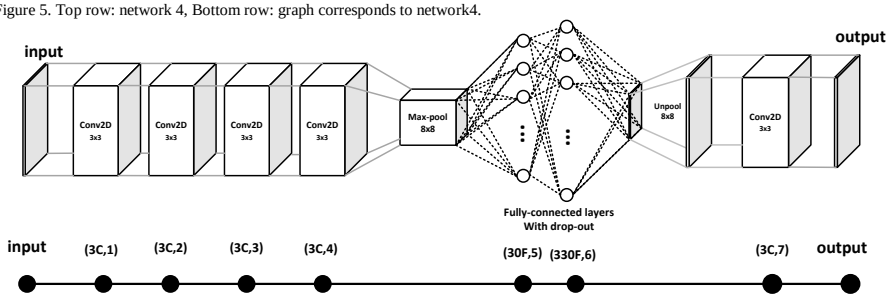


Figure 6. Top row: network 5, Bottom row: graph corresponds to network5.

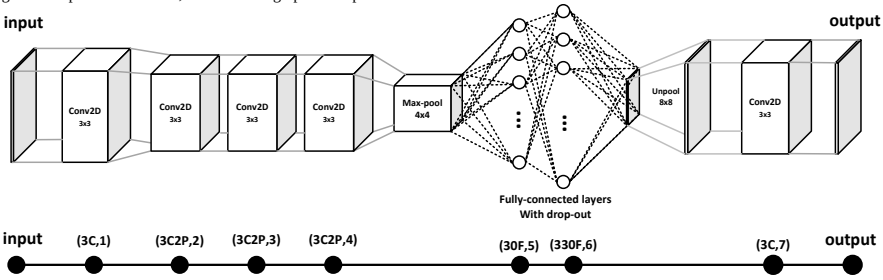


Figure 7. Top row: network 5, Bottom row: graph corresponds to network6.

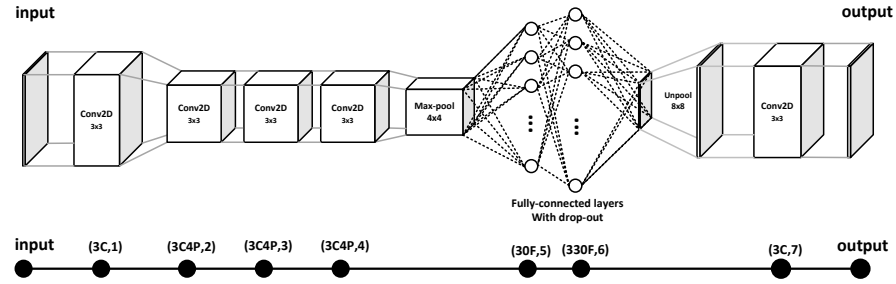


Figure 8. Top row: network 7, Bottom row: graph corresponds to network7.

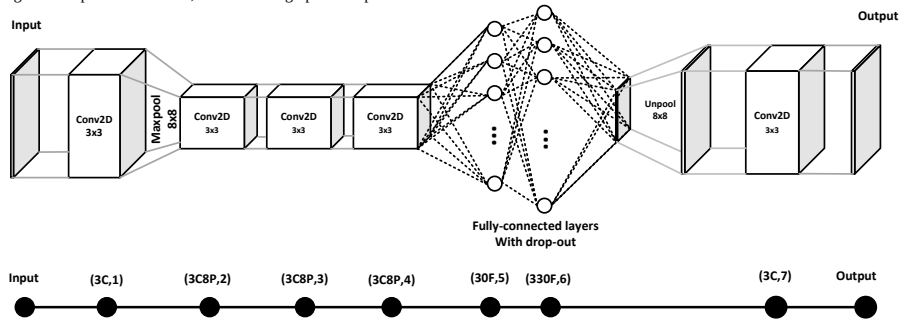


Figure 9. Top row: network 8, Bottom row: graph corresponds to network8.

Formatted: Left, Space Before: 0 pt

V. The SPDNN Parallelization Methodology

A. Graph Contraction

The problem with parallelizing networks is that if we have the same structure of layers with the same distance from the input it might lead all these layers to converge to the same values. For example the first layer in all of the networks shown in Fig. 2 – Fig. 9 is a 2D convolutional layer with 3×3 kernel. It is the first layer of all the networks; therefore to avoid redundancy in the network structure, all these layers will be merged into one single layer.

The SPDNN idea is using the idea of graph contraction method to merge several neural networks. The first step to do so is to turn each network into a graph. Turning a network into a graph needs to consider each layer of the network as a node of the graph. Each graph starts with the input node and end with output node. The nodes in the graph are connected based on the corresponding layer connections in the neural network. Note that the pooling and un-pooling layers are not considered as a node in the graph but their properties will stay with the graph labelling which will be explained.

In Fig. 2 – Fig. 9 you can see that the graph corresponding to each network is shown under each network. Two properties are assigned to each node in the graph. The first property is the layer structure, and the second one is the distance of the current node to the input node. In order to assign the first property we used different signs for different layer structures, C for convolutional layer (for example 3C mean a convolutional layer with 3×3 kernel), F for fully connected layer (for example 30F means a fully connected layer with 30 neurons), and P for pooling property (for example 4P means that the data has been pooled by the factor of 4 in this layer).

Some properties like convolutional and fully connected layer are happening in that specific node, but pooling and un-pooling will stick with data to the next layers. In other words, the pooling property will stay with the data except we reach an un-pooling or fully connected layer. For example a node with the properties (3C8P,4) correspond to a convolutional layer with a 3×3 kernel, which the data has been pooled with the order of 8 in previous layers, and has the distance 4 from the input layer. In Fig. 2 – Fig. 9 under each network the corresponding graph with assigned properties is illustrated.

The next step is to put all these graphs in a parallel format sharing a single input and single output node. Fig. 10 shows the graph in this step.

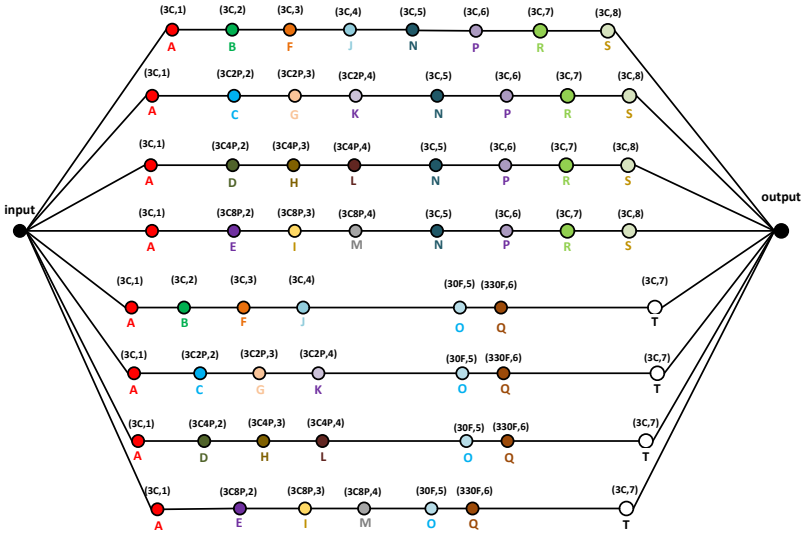


Figure 10. Parallelized version of the graphs shown in Fig. 2 – Fig. 9 sharing a single input node and single output node

In order to merge layers with the same structure and the same distance from input node, the nodes with the exact same properties are labelled with the same letters. For example all the nodes with properties (3C, 1) are labelled with letter A, and all the nodes with the properties (3C2P, 4) are labelled as K and so on.

After labelling the nodes the next step is to apply the graph contraction on the large graph. In the graph contraction procedure the nodes with the same label will be merged to a single node while saving their connections to the previous/next nodes. For example all the nodes with label A will be merged in to one node, but its connection to the input node and also nodes B,C,D, and E will be preserved. The contracted version of the graph in Fig. 10 is shown in Fig. 11.

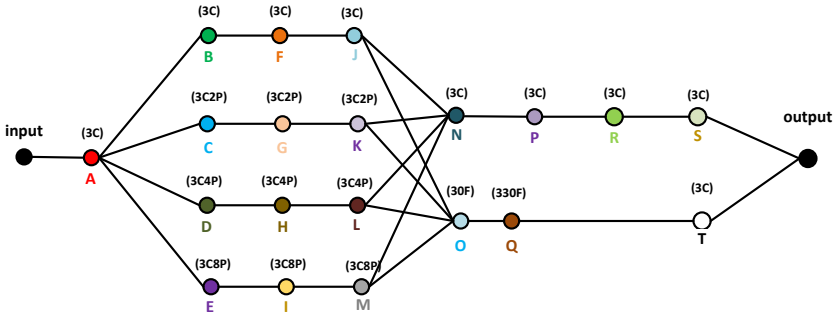


Figure 11. Contracted version of the big graph shown in Fig. 10

The next step is to convert the graph back to the neural network structure. We still have the structural properties of each node in our graph. In order to convert the graph back to a neural network, these properties will be used. For example we know that node C is 3×3 convolutional layer which has experienced a pooling operation. Note that the pooling quality will be recalled from the original network.

The concatenation layer has been used in the neural network in order to implement the nodes wherein several other nodes are lead to one node. For example in nodes N, and O, the outputs of nodes J, K, L, and M have been concatenated with the pooling qualities took from their original networks.

B. The Combined Model/Architecture

Formatted: Heading 2, Left

In the final model presented in Fig. 13, the input image is first processed in 4 parallel fully convolutional networks with different pooling sizes. This would give us the opportunity of getting the advantage of using different networks with different pooling sizes at the same time. The results of these networks are concatenated in two different ways one with pooling the larger images to be as the same size as the smallest image in the previous part and the other is to un-pool the smaller images of the previous part to be the same size of the largest image.

After merging the results, the data is led to two different networks. One is the fully convolutional network to deepen the learning and release more abstract features of the input, and the other network is an auto-encoder network with different architecture for encoder and decoder.

Based on our observations, the network needs to see the whole image as one input in some stage in order to be able to estimate the depth; from the other side putting a convolutional layer with the filter same size of the image is equal to use a fully connected layer. Therefore, having fully connected layer in the network became crucial in order to get a reasonable estimation of the depth which is happening in the neck of the auto encoder. The results from the auto encoder and the fully convolutional layer are again merged in order to give a single output after applying a one channel convolutional layer.

In order to regularize the network to prevent overfitting and increase the convergence, the batch normalization [18] is applied after every convolutional layer, and the drop-out technique [17] is used in fully connected layers. Using the weight regularization in the fully connected layers seems to give slower convergence in our observations so we didn't use that technique in our final version. All the nonlinearities in the network are the well-known ReLU non linearity which is widely used in Deep Neural Networks except the output layer which took advantage of sigmoid nonlinearity. And the value repeating technique has been used in un-pooling layer due to non-specificity of the corresponding pooled layer especially in the decoder part of the auto-encoder sub-network.

The value repeating technique is illustrated in the Fig. 12 which is easily repeating the value from the previous layer in order to obtain the un-pooled image. The figure shows the 2×2 un-pooling and the process is the same for other un-pooling sizes.

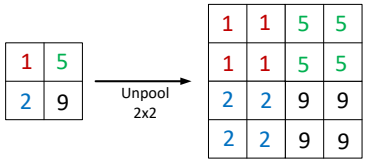


Figure 12. Repeating technique used in un-pooling layers.

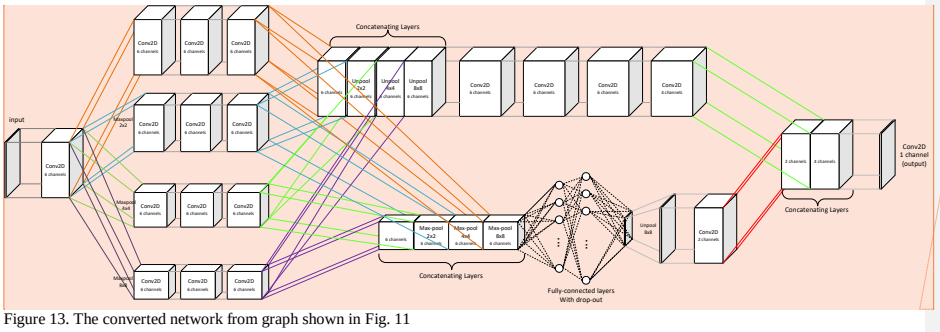


Figure 13. The converted network from graph shown in Fig. 11

Commented [CP16]: You'll need to enlarge the text in the boxes … probably a starting point is to compress the figure along its length …

~~b)~~C. Training Set

Formatted: Heading 2, Space Before: 0 pt, No bullets or numbering

In this paper KITTI Stereo 2012, 2015 [26] are used for training purposes. The disparity of each set is estimated by the method which has been explained earlier in the paper. The left images of the sets along with the estimated disparity map are used to train the network. Each image is flipped vertically, horizontally and each vertically flipped image is flipped horizontally as well in order to augment the database to cover all the possibilities of the disparity in case of camera rotation. In total 33,096 images are used in this research. 70% of the initial set is considered for training, 20% for validation and 10% for test purposes. Each model is trained for two set of input samples and two sets of output targets. Therefore we have four different experiments as follows:

1. First Model: Input: Left RGB Image + Pixel-wise Segmented Image. Target: Post-Processed Disparity.
2. Second Model: Input: Left RGB Image. Target: Post-Processed Disparity.
3. Third Model: Input: Left RGB Image + Pixel-wise Segmented Image. Target: Disparity.
4. Fourth Model: Input: Left RGB Image. Target: Disparity.

The images are resized to 80×264 pixels during the whole process. Training is done on a standard desktop with an NVIDIA GTX 1080 GPU with 8GB memory.

D. Experimental Results

Formatted: Heading 2, Left

In all our experiments the mean square error value between the output of the network and the target values has been used as the loss function and Nestrov momentum technique [27] with learning rate 0.01 and momentum 0.9 has been used to train the network. The Train Loss and Validation Loss for each of these experiments are shown in Fig. 14 and Fig. 15 respectively. As it is shown in these figures using the Post-Processed Disparity as target results in lower loss values which means that the network was able to learn better features in those experiments.

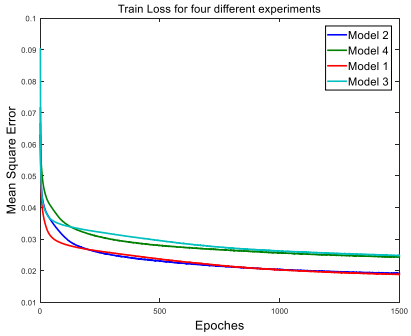


Figure 14. Train Loss for each experiment

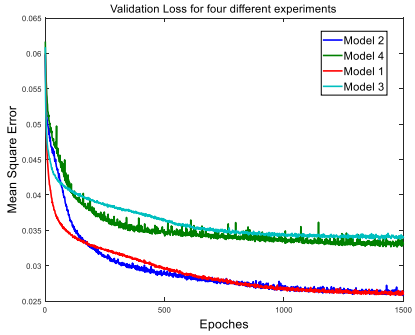


Figure 15. Validation Loss for each experiment

~~4)~~2) Results

The evaluation in this paper has been done in 2 parts. In the first part the results of the trained models are compared against the ground truth provided by KITTI benchmark.

In the second part we considered the disparity maps computed by our recent stereo matching technique [21] as ground truth. The goal behind the second part of the comparison is to find out how close are the disparity maps from the trained model to the disparity maps computed by a stereo matching method and to briefly evaluate the performance of the stereo matching technique in this type of applications.

For the evaluation purposes 8 metrics including PSNR, MSE, RMSE, SNR, MAE, Structural Similarity Index (SSIM) [28], Universal Quality Index (UQI) [29] and Pearson Correlation Coefficient (PCC) [30] are used. Table 2 shows the results of the first part of our experiment, the numerical comparison of the trained models against the ground truth provided by the benchmark. The best value for each metric is highlighted in green. Fig. 16 – Fig. 18 represent the colour coded disparity maps computed by the trained models using the proposed DNN. For the visualization purposes all the images presented in this section are upsampled using Joint Bilateral Upsampling [31].

Table 2. Numerical Comparison of the Models – Experiment 1

	After With SegNet	After Without SegNet	Before With SegNet	Before Without SegNet
PSNR	14.3424	13.7677	13.8333	13.8179
MSE	0.0382	0.0436	0.0435	0.0439
RMSE	0.1937	0.2069	0.206	0.2066
SNR	4.4026	3.8279	6.1952	6.1798
MAE	0.1107	0.1212	0.1236	0.1234
SSIM	0.9959	0.9955	0.9955	0.9955
UQI	0.9234	0.9252	0.9053	0.9064
PCC	0.7687	0.8485	0.7702	0.7729

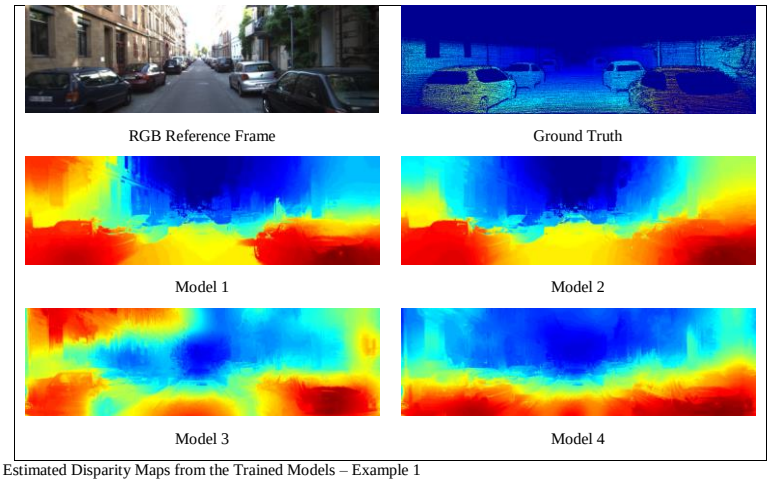
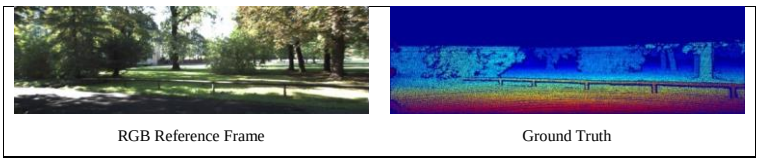


Figure16. Estimated Disparity Maps from the Trained Models – Example 1



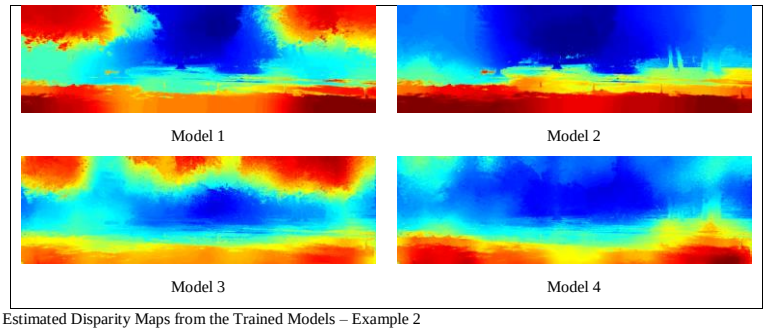


Figure 17. Estimated Disparity Maps from the Trained Models – Example 2

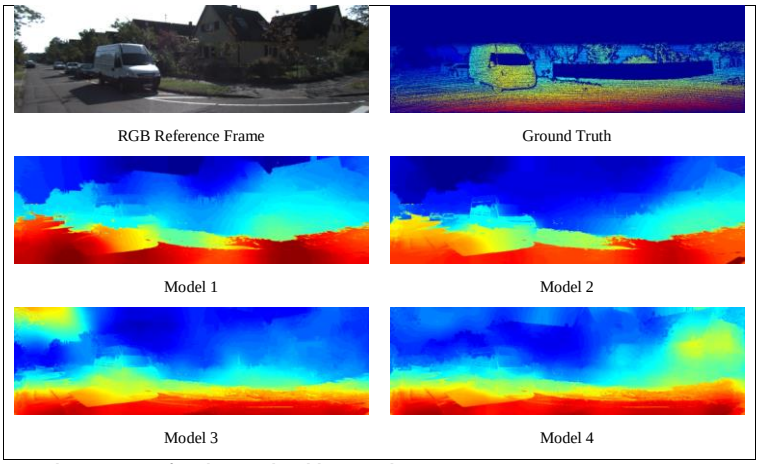
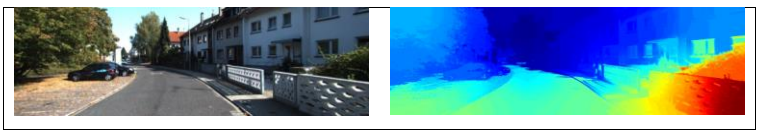


Figure 18. Estimated Disparity Maps from the Trained Models – Example 3

The numerical comparison of the trained models for the second part of our experiment is shown in Table 3. The colour coded disparity maps computed by the trained models using the proposed DNN are illustrated in Fig. 16 – Fig. 18.

Table 3. Numerical Comparison of the Models – Experiment 2

	After With SegNet	After Without SegNet	Before With SegNet	Before Without SegNet
PSNR	15.0418	14.1895	13.3819	14.0491
MSE	0.0378	0.0447	0.0535	0.0441
RMSE	0.1854	0.203	0.2223	0.2039
SNR	8.822	7.9696	5.4271	6.0943
MAE	0.1442	0.1581	0.1673	0.153
SSIM	0.9952	0.9943	0.994	0.9951
UQI	0.8401	0.8369	0.7951	0.8178
PCC	0.8082	0.795	0.704	0.6919



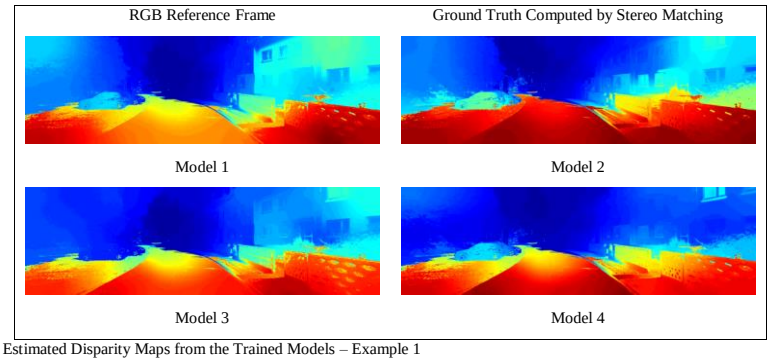


Figure19. Estimated Disparity Maps from the Trained Models – Example 1

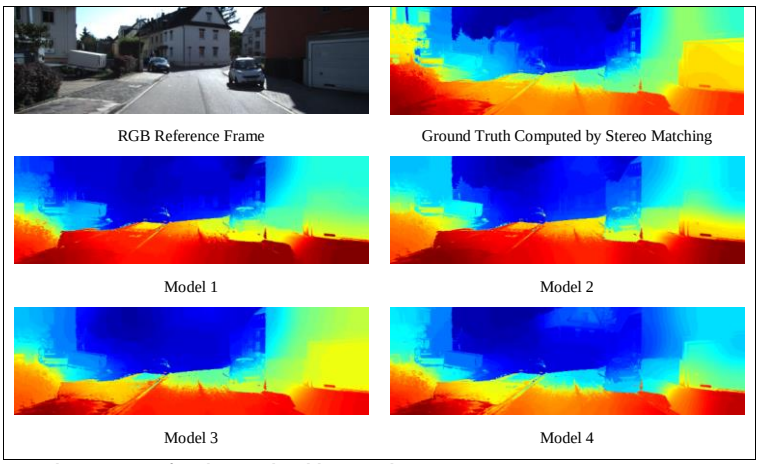


Figure 20. Estimated Disparity Maps from the Trained Models – Example 2

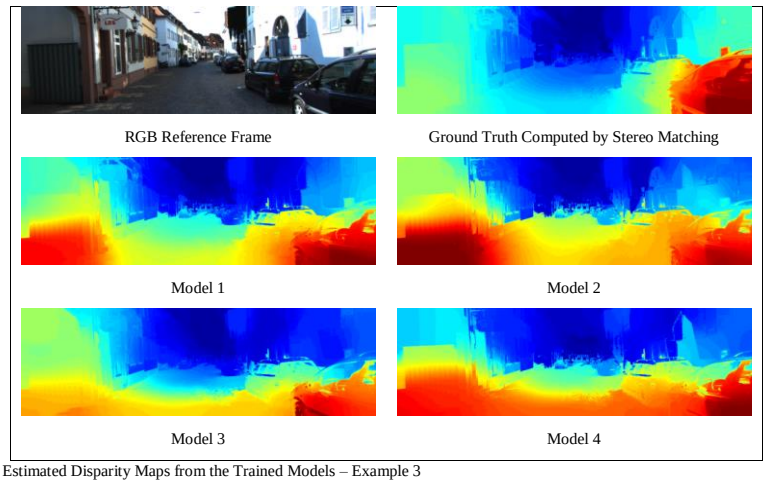


Figure 21. Estimated Disparity Maps from the Trained Models – Example 3

The evaluation results at this stage are a proof for the idea of estimating depth map from monocular images/cameras by employing DNN. At this point we can claim that trained models estimate depth values with a

close accuracy to the depth values computed by a stereo matching method by considering an important advantage of ~0.23 second computational time (on the NVIDIA GTX 1080 GPU).
Table 4 shows the numerical evaluation of our stereo matching technique against the ground truth images used at the first part of the experiment. Clearly the comparison of these numbers against Table 2 shows that the higher accuracy is achievable by using the stereo matching technique but the differences are not that significant.

Table 4. Numerical Comparison of the Models – Experiment 2

	After Post-Processing	Before Post-Processing
PSNR	14.8234	14.1384
MSE	0.0351	0.043
RMSE	0.1845	0.2017
SNR	4.8836	6.5003
MAE	0.1017	0.1213
SSIM	0.9966	0.9955
UQI	0.9353	0.9069
PCC	0.823	0.7797

~~5)~~VI. Conclusions & Future Work

Formatted: Heading 1, Space Before: 0 pt, No bullets or numbering

In this paper we have presented a deep neural network to train a highly accurate model for estimating depth from monocular images. In total 4 models have been trained. Pixel-wise segmentation and our post-processing technique have been used to provide inputs for 2 of the models. KITTI benchmark has been used for training and experimental purposes. Each model has been evaluated in 2 sections, first against the ground truth provided by the benchmark a second against the disparity maps computed by the stereo matching method. The results showed that a slightly higher accuracy can be obtained by employing the stereo matching technique but our evaluations show that there is not a big difference between the depths from the models trained by proposed DNN and the values computed by stereo matching. It is also worth pointing out an important advantage of these models which is the processing time of ~0.23 second and it makes the whole process of the depth estimation applicable for real time purposes. Using the pixel-wise segmentation as one of the inputs of the network has slightly increased the performance of the models in terms of accuracy but it brought some artifacts such as wrong depth patches on a surfaces. The evaluation results also illustrate the higher accuracy of the models where post-processed disparity was used as the target in training procedure and it is a proof for the performance of our post-processing method.

References

1. D. Nair. (Aug 24, 2016). *3D Imaging with NI LabVIEW*. Available: http://www.ni.com/white-paper/14103/en/
2. C. Niclass, M. Soga, H. Matsubara, and S. Kato, "A 100m-Range 10-Frame/s 340×96-Pixel Time-of-Flight Depth Sensor in 0.18-µm CMOS," in *ESSCIRC (ESSCIRC), 2011 Proceedings of the*, 2011, pp. 107-110.
3. C. Niclass, K. Ito, M. Soga, H. Matsubara, I. Aoyagi, S. Kato, *et al.*, "Design and characterization of a 256x64-pixel single-photon imager in CMOS for a MEMS-based laser scanning time-of-flight sensor," *Optics Express*, vol. 20, pp. 11863-11881, 2012/05/21 2012.
4. D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. I-195-I-202 vol.1.
5. R. K. Gupta and S.-Y. Cho, "A Correlation-Based Approach for Real-Time Stereo Matching," in *Advances in Visual Computing: 6th International Symposium, ISVC 2010, Las Vegas, NV, USA, November 29 – December 1, 2010, Proceedings, Part II*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, R. Chung, R. Hammound, *et al.*, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 129-138.
6. C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "MeshStereo: A Global Stereo Model with Mesh Alignment Regularization for View Interpolation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2057-2065.
7. *Middlebury Stereo Evaluation - Version 3*. Available: http://vision.middlebury.edu/stereo/eval3/

8. K. R. Kim and C. S. Kim, "Adaptive smoothness constraints for efficient stereo matching using texture and edge information," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3429-3433.
9. X. Huang, Y. Zhang, and Z. Yue, "Image-Guided Non-Local Dense Matching with Three-Steps Optimization," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. III-3, pp. 67-74, 2016.
10. A. Li, D. Chen, Y. Liu, and Z. Yuan, "Coordinating Multiple Disparity Proposals for Stereo Computation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, pp. 4022-4030.
11. M. Shahbazi, G. Sohn, J. Théau, and P. Ménard, "Revisiting Intrinsic Curves for Efficient Dense Stereo Matching," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. III-3, pp. 123-130, 2016.
12. J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, pp. 2287-2318, 2016.
13. J. T. Barron and B. Poole, "The Fast Bilateral Solver," *arXiv:1511.03296*, vol. abs/1511.03296, 2016.
14. E. T. Psota, J. Kowalczuk, M. Mittek, P. L. C, and rez, "MAP Disparity Estimation Using Hidden Markov Trees," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2219-2227.
15. J. Kowalczuk, E. T. Psota, and L. C. Perez, "Real-Time Stereo Matching on CUDA Using an Iterative Refinement Method for Adaptive Support-Weight Correspondences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 94-104, 2013.
16. B. Yegnanarayana, *Artificial neural networks*: PHI Learning Pvt. Ltd., 2009.
17. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929-1958, 2014.
18. S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167*, vol. abs/1502.03167, 2015.
19. F. Liu, C. Shen, G. Lin, and I. Reid, "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2024-2039, 2016.
20. A. Saxena, J. Schulte, and A. Y. Ng, "Depth estimation using monocular and stereo cues," in *Proceedings of the 20th international joint conference on Artifical intelligence*, Hyderabad, India, 2007, pp. 2197-2203.
21. H. Javidnia and P. Corcoran, "A Depth Map Post-Processing Approach Based on Adaptive Random Walk With Restart," *IEEE Access*, vol. 4, pp. 5509-5519, 2016.
22. V. Badrinarayanan, A. Kendall, and R. Cipolla, " SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv:1511.00561*, vol. abs/1511.00561, 2015.
23. A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.
24. A. Kendall, V. Badrinarayanan, and R. Cipolla. *Caffe Implementation of SegNet*. Available: https://github.com/alexgkendall/caffe-segnet
25. G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and Recognition Using Structure from Motion Point Clouds," in *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I*, D. Forsyth, P. Torr, and A. Zisserman, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 44-57.
26. M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061-3070.
27. I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," *ICML (3)*, vol. 28, pp. 1139-1147, 2013.
28. W. Zhou, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600-612, 2004.
29. W. Zhou and A. C. Bovik, "A Universal Image Quality Index," *IEEE Signal Processing Letters*, vol. 9, pp. 81-84, 2002.
30. K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240-242, 1895.
31. J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph.*, vol. 26, p. 96, 2007.s