

Calculating Disparity Images with Denoising Joint Filtering using Deep Convolutional Networks

Shai Weismann

300106358

shai.weisman@gmail.com

Moran Altmark

301328647

moran.altmark@gmail.com

Abstract— Disparity estimation is a challenging task in stereo vision because the correspondence technique fails in images with texture less, repetitive and/or caged regions. Our proposed network is a combination of two independent CNNs, DenseMapNet [1] for disparity estimation and JointFilterNet [2] for post processing and noise reduction. The DenseMapNet is CNN implementation inspired by dense networks to reduce the number of parameters and allowing fast implantation. The JoinFilterNet is based on a CNN, that can selectively transfer salient structures that are consistent with both guidance and target images. In our end results we can see on certain images improvement between the Dense map using the JF filter than without.

1 INTRODUCTION

Stereo imaging is one of the most active research areas in computer vision. The idea of stereo imaging is to extract 3D information by comparing information from two vantage points, right and left. Using the two images we can compute the disparity of a pixel, for a given (x, y) on the left image the disparity is the offset d of its location at $(x - d, y)$ on the right image. The depth z can be computed as well (eq. 1):

$$1. \quad z = \frac{fB}{d}$$

B is the stereo cameras baseline and f is the camera focal length. Both constants can be measured through camera calibration. The larger the disparity d the closer it is to the baseline; an example can be seen in Figure 1.

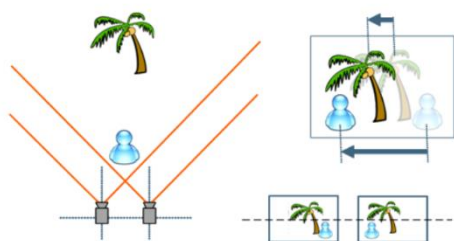


Figure 1. We can see two cameras taking an image of two objects which locating in different depth, we can see that the disparity of the farther object (tree) is smaller than the disparity of the closer object (blue man)

Depth image estimation is common in many fields such as autonomous driving, robot motion planning, intermediate view generation, and 3D scene reconstruction. In many stereo models after the disparity estimation we performed **post processing** to reduce noise and improve object contours. For example the stereo model in the OpenCV library uses Semi-Global Matching [3] algorithm for disparity estimation and Fast Global Smoothing algorithm [4] for post processing. One of the most common methods for post processing is Joint Bilateral Filtering [5] which uses the left image as guide for noise reduction in the disparity map. Our proposed network is a combination of two independent CNNs, DenseMapNet [1] and JointFilterNet [2]. Our goal is to improve the DenseMapNet results by optimizing the network loss function. In addition, we chose to apply JointFilterNet CNN for post processing, Changing the activation function, adding dropout and batch normalization layers and optimize the network loss function. The DenseMapNet implementation is based on prior work while the JointFilterNet was implemented based on the article without any code reference in python. All the network was built using the Keras API. We trained the DenseMapNet on 144 images and tested on 17 images from the “Monkaa Family” [6] dataset and reached similar results as in the original article of DenseMapNet; error rate of average 4.5 end point error(EPE) eq. 2.

After training the JointFilterNet on 50K patches images the average EPE on 5K decreases by average in 60% percent compared to the DenseMapNet (Table 3).

2 RELATED WORK

Generally, **stereo matching** uses one or more building blocks in the form of: Matching computation, Cost aggregation, Disparity computation/optimization and Disparity refinement. The local algorithms attempt to minimize a cost function between image patches by different methods such as Sum of Absolute Differences (SAD) or MC-CNN [7]. The global algorithms aim to minimize a global function with smoothness assumptions to arrive at disparity estimates. Example is the Semi Global Matching method, which minimizes a global cost function and energy term. In patch-based correspondences, the idea is to compute the correspondence matching cost between the left image patch and the candidate right image patch. Since the input stereo images are rectified, the search for a match is limited to the corresponding pixel row on the right image. The clear disadvantage of the MC-CNN patch-based method is it is just replacing the matching algorithm of the classical method with deep neural networks. The rest of the pipeline is still expensive computationally because of the huge number of patches to consider even if GPU parallel processing is involved. Furthermore, there are still post-processing steps involved preventing a complete end-to-end learning. Up sampling or super resolution is applied to get the full disparity map in case the predicted image is of lower dimensions, but this may cause artifacts and quantized edges. The DenseNet [8] introduced a new CNN architecture wherein all layers are connected to all previous inputs and outputs. DenseNet argues that most of the time residual networks have residual layers with little contribution or effect in the final prediction. Since residual layers have their own set of parameters, the number of weights to train increases unnecessarily. In DenseNet all previous layer's inputs are combined with the current layer inputs and share the same set of parameters. In effect, DenseNet uses significantly less number of parameters. The model we have used, DenseMapNet, differs in few key areas, first, it utilized full resolution images on both input and output, Using Max pooling and up sampling to manage the number of parameters and memory use of the network. Second, to resolve the problem of vanishing gradient and to reduce the number of parameters to train, the DenseMapNet model has each layer connected to all previous layers like DenseNet. Third, it is known that to generate disparity map we should use one image as reference (e.g. left image) and increase the intensity of each pixel in proportion to its disparity (by finding its corresponding match on the right image using Correspondence Network). This reasoning yields the design of the DenseMapNet, a two-network model - a Correspondence Network and a Disparity Network (Figure 3). Each layer has direct access to all previous output layers and stereo images, in every stage there is dropout to address the model generalization.

Joint image filters. Joint image filters can be categorized into two main classes based on explicit filter construction or global optimization of data fidelity and regularization terms. Explicit joint filters compute the filtered output as a weighted average of neighboring pixels in the target image. The bilateral filters [5], [9] and guided filters [10] are representative algorithms in this class. The filter weights, however, depend only on the local structure of the guidance image. Therefore, inaccurate or extraneous structures may be transferred to the target image due to the lack of consistency constraints. The network model we've implemented as described in 'Joint Image Filtering with Deep Convolutional Networks' [2] which named as JointFilterNet use both images based on feature maps and enforces consistency implicitly through learning from examples. Numerous approaches formulate joint filtering based on a global optimization method. The objective function typically consists of two terms: data fidelity and regularization terms. The data fidelity term ensures that the filtering output is close to the input target image. Global optimization-based methods rely on hand-designed objective functions that may not reflect the complexities of natural images. Furthermore, these approaches involve iterative optimization are often time-consuming. In contrast, the method in [2] learns how to selectively transfer important details directly from the RGB/depth data. Although the training process is time consuming the learned model is efficient during run-time.

Learning-based image filters. Substantial efforts have been made to construct image filters using learning algorithms and CNNs. For example, the conventional bilateral filter can be improved by replacing the predefined filter weights with those learned from a large amount of data [11]. In the context of joint depth up sampling, Tai et al. [12] use a multi-scale guidance strategy to improve up sampling performance. Gu et al. [13] adjust the original guidance dynamically to account for the iterative updates of the filtering results. However, these methods [12], [13] are limited to the application of depth map up sampling. In contrast, the model we've implemented construct a generic joint filter for various applications using target/guidance image pairs in different visual domains.

Skip connections. As deeper the networks are, the information contained in the input or gradients can vanish and wash out by the time it reaches the end or beginning of the network. He et al. [14] address this problem through bypassing the signals from one layer to the next via skip connections. This residual learning method facilitates to train very deep networks effectively.

3 DATA

We used the Monkaa Family dataset, which contain stereo pairs and the ground truth disparity. This dataset is from an open-source animated movie rendered in Blender. There are over 8,000 synthesized stereo images in this dataset. Monkaa has the same image specifications as Driving. Maximum disparity value is 10,500 pixels. For our proposed model we've normalized the images and trained only on 144 images and tested on 17 images for the DenseMap net, and used 50K patches of 64x64 pixels from 14 out of the 17 test images for training the joint filter Net. 3 images were used for testing the final output. The advantage of this synthetic dataset vs. *KITTI* or *Middlebury* datasets is that the ground truth disparity information is accurate and not calculated.

4 METHODS

The purpose of our proposed network is to improve the DenseMapNet results by post filtering to decrease the error significantly. Our network is a combination of two independent CNNs (Figure 2), DenseMapNet (Figure 3), and JointFilterNet (Figure 4). We would like to explain **DenseMapNet** method as described in the paper, it's two-network model – a Correspondence Network and a Disparity Network. Each layer has direct access to all previous output layers and stereo images. Lastly, we use dropout in every stage to address model generalization. To generate the correspondence map, the algorithm imposes that we choose a reference image, like the left image. Assuming rectified stereo images, for each pixel on the left image, we look for the same pixel on the right image by searching on the corresponding row. If there is no occlusion, we will find the match. Otherwise, we approximate its location. The row column offset is called the disparity. Given a calibrated camera, we can determine the corresponding depth using eq. 1. For each pixel disparity, we can generate a disparity map using disparity as the measure of intensity or brightness. The intensity is assigned to every pixel in the reference image. Hence, for disparity images, the brighter the object, the closer it is to the camera coordinate system origin. Using the description of the algorithm for disparity map estimation, Atienza et al. [1] designed DenseMapNet as shown in Figure 2. DenseMapNet has 18 CNN layers and has two networks to mimic the algorithm for disparity estimation: 1) Correspondence Network and 2) Disparity Network. The idea is for the Correspondence Network to learn stereo matching while the Disparity Network applies the disparity on the reference image. The Correspondence Network aims to find pixel correspondences of stereo images. Hence, instead of making the network deep, the Correspondence Network is designed to be wide to increase the coverage of the kernel. The Disparity Network utilizes the learned representations from the Correspondence Network to estimate the amount of disparity to be applied on the

reference image. As shown in Figure 3, the Disparity Network processes both feature maps from the reference image (left image) and the Correspondence Network to lead DenseMapNet in estimating the disparity map. The Disparity Network has 13 CNN layers. The important feature of DenseMapNet is the Dense Network-type of connection wherein the loss function has access to all feature maps down to the input layer. The output of the immediate previous layer and inputs of all previous layers are inputs to the current layer. The loss function's immediate access to all CNN layers prevents gradients from vanishing as they travel down the shallow layers. Parameter sharing makes this type of CNN efficient by significantly reducing the total number of weights to train. Immediate access to weights and parameter sharing make DenseMapNet easy to train.

The joint image filter based on CNNs model we've implemented and modified (JointFilterNet) consists of three sub-networks: the target network CNN_T , the guidance network CNN_G , and the filter network CNN_F as shown in Figure 3. First, the sub-network CNN_T takes the target image as input and extracts a feature map. Second, like CNN_T , the sub-network CNN_G extracts a feature map from the guidance image. Third, the sub network CNN_F takes the concatenated feature responses from the subnetworks CNN_T and CNN_G as input and generates the residual, i.e., the difference between the degraded target image and ground truth. By adding the target input through the skip connection, we obtain the final joint filtering result. Here, the main roles of the two sub-networks CNN_T and CNN_G are to serve as nonlinear feature extractors that capture the local structural details in the respective target and guidance images. The sub-network CNN_F can be viewed as a non-linear regression function that maps the feature responses from both target and guidance images to the desired residuals. Note that the information from target and

guidance images is simultaneously considered when predicting the final filtered result. Such a design allows to selectively transfer structures and avoid texture-copying artifacts.

Methods Implementation

Since the first article already being implemented in Keras using residual layers, dropout and batch normalizations we focused on optimizing the loss while using smaller dataset. In the article they used as loss function the binary cross-entropy, we've tested the following loss functions: 1. MSE 2. Logcosh 3. binary cross-entropy. The DenseMapNet for stereo calculation was chosen due to two main reasons: 1. Number of parameters, The DenseMapNet, is compact and requires 290k parameters only compared to 3.5M or more parameters in other similar CNN-based approaches. As consequence of its small size, DenseMapNet is faster than other Stereo methods. 2. Full image-based calculation while patch images-based calculation i.e. MC-CNN requires additional processing, for

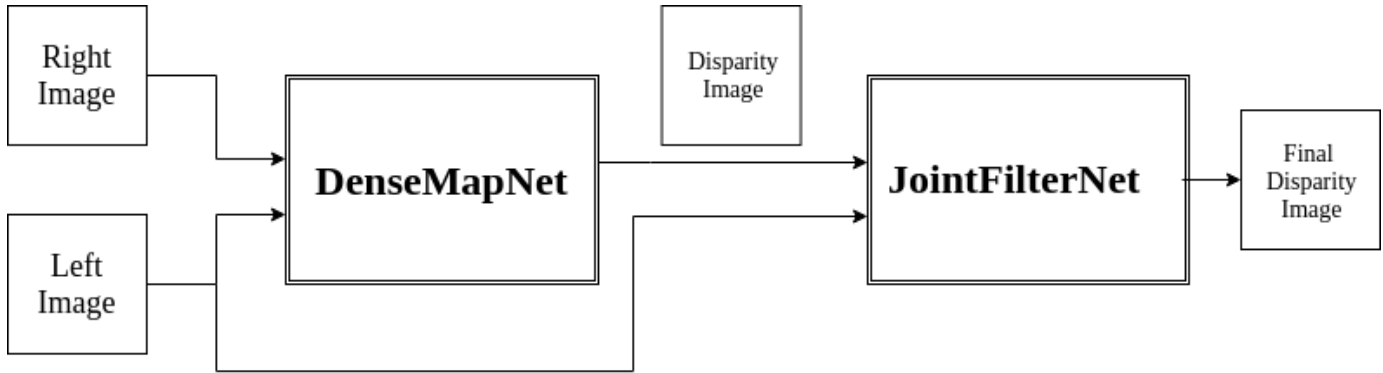


Figure 2 Our model diagram

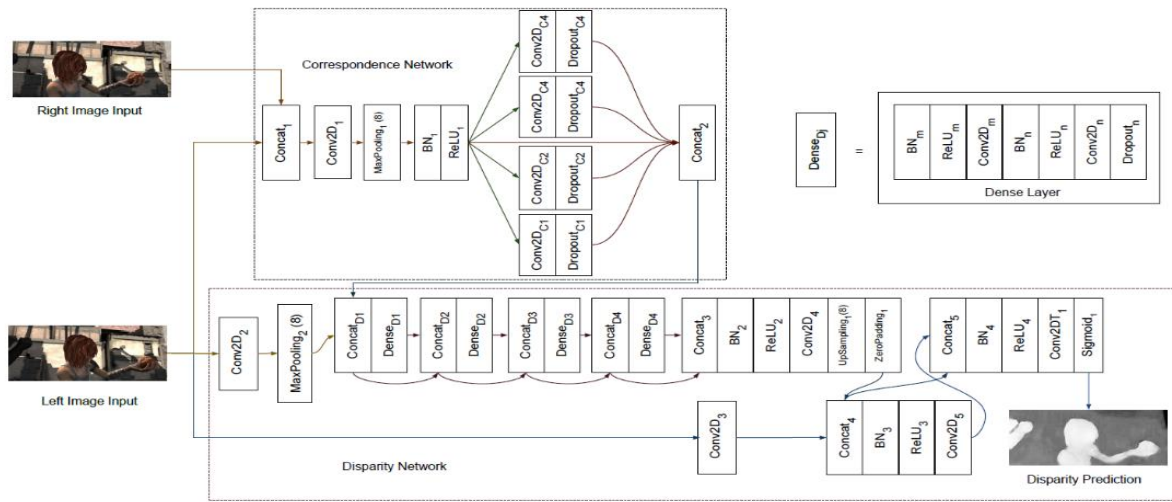


Figure 3 DenseMapNet architecture (from the paper). The Correspondence Network learns how to estimate stereo matching between left and right images. The Disparity Network applies the correspondence on the left image. The detail of each Dense Layer is also shown.

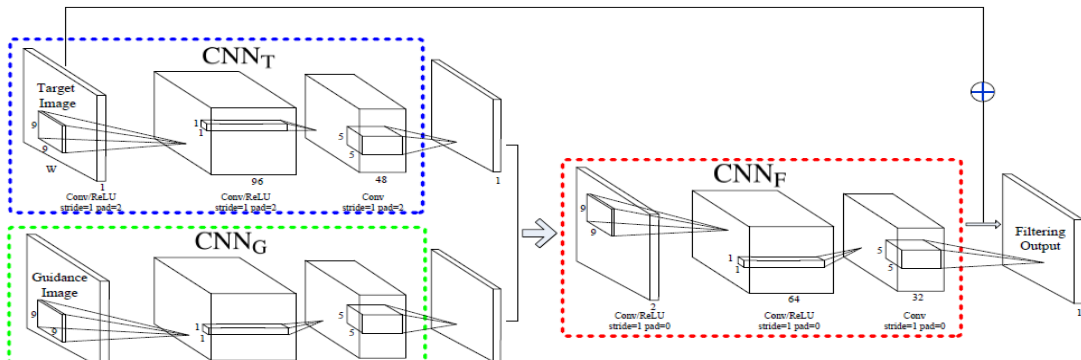


Figure 4 JointMapNet architecture as shown in the paper.

example finding the minimum cost and performing SGM. To improve the DenseMapNet results by post filtering we've added the JointFilterNet, this is a new approach for denoising stereo images. The post filtering CNN as described in the article was relatively thin.

The JointFilterNet was implemented fully by us, we performed several experiments:

1. Activation Functions: ReLU (as in the paper) ELU
2. Batch normalization
3. Dropout
4. Loss functions: MSE, Logcosh, Binary cross-entropy

The idea to change the activation function was following various papers which showed better results using ELU. To avoid overfitting and vanishing gradients we've considered to test the network performance with batch normalization and dropout which weren't used in the original paper. Regarding the loss function, it wasn't defined in the paper therefore we've tested as described above.

The proposed deep joint image filter model consists of three major components. Each component is a three-layer network, which has Batch normalization and Dropout between each 2 layers. The sub-networks CNN_T and CNN_G aim to extract informative feature responses from the target and guidance images, respectively. We then concatenate these features responses together and use them as input for the network CNN_F . In addition, we introduce a skip connection so that the network CNN_F learns to predict the residuals between the input target image and the desired ground truth output. CNN_F has batch normalization and dropout after each layer and also dropout after the skip connection. We train the network with two patches; target patch T is 64X64 pixels from the disparity image created by

the DenseMapNet and Guide image patch G which is from the corresponding lefts patch image. The parameters of all three sub-networks are updated simultaneously during the training stage.

5 EXPERIMENTS

5.1 DenseMapNet

To better optimize the DenseMapNet we performed network fitting based on 3 different losses; Binary Cross Entropy (as suggested in the article), Mean Square Error and Logcosh. Logcosh was selected, due to the fact that it acts as L2 loss in large values and L1 in small values, and it showed value in encoder and regression models. The training was done on a Dataset of 144 images for train and 17 images for test (in the

original article the training was on 7200 images and 800 images for test) and the error was calculated by the average end point error on pixel

$$2. \quad err = \frac{1}{N} \sum_{p \in I} |GT(p) - I(p)|$$

In which N is number of pixels, GT is Ground Truth and I is the predicted image.

Table 1 below summarizes our optimization experiment for the DenseMapNet. We can note based on the plot (which was calculated on test data set), that initially binary cross entropy is the best loss method, but after 100 iterations MSE and Logcosh, all loss methods receive better error rates, and after 200 iterations all the methods reach similar values. This can also be seen in the images in table 1, in the first iteration it is clear that the network sees mostly noise, but after 50 epochs, the main artifacts disappear and after 100 epochs we receive an image which is similar to the output in later epochs. In total we reached an epe score of 4.5, while in the original article they reached 4.45 and if using SGM the epe score is 20. meaning we successfully reproduced the original article result using a smaller data set.

5.2 JointFilterNet

Due to the fact that we implemented the JointFilterNet by ourselves, we decided to verify our code and test that we get similar outputs as the original article. we've trained the network using a resized image (T) by down sample and up sample the Ground Truth disparity image and used the corresponding Left image as the guidance image (G). We assumed that down sampling and up sampling will add noise and quantize the contours in the original image. The results we receive from this sanity test as shown in Figure 5, is that the JointFilterNet net successfully reconstructed the contour of the disparity image and smoothed the surface, similar to a joint bilateral filter.

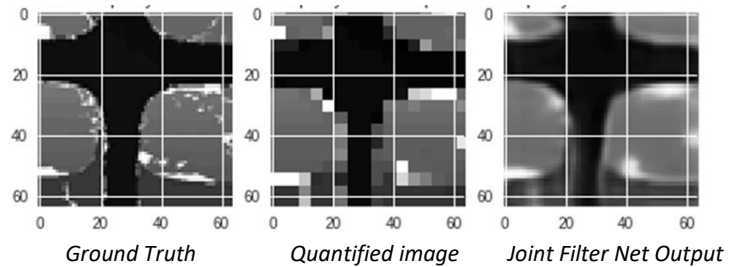














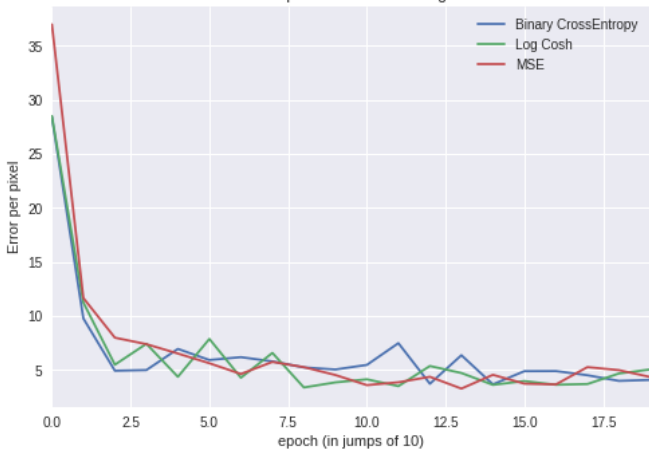



Figure 2 Ground truth, Resized image, final map after JointFilterNet

Table 1 Loss Optimization on DenseMapNet

Binary Cross Entropy			
			
Epoch 1	Epoch 10	Epoch 50	Epoch 100
Mean Square Error			
			
Epoch 1	Epoch 10	Epoch 50	Epoch 100
Log CosH			
			
Epoch 1	Epoch 10	Epoch 50	Epoch 100
			
Left	Right		
			
Ground Truth			

Afterwards we focused to optimize the network performance first by choosing activation function and secondly by the following three parameters: loss, dropout and batch normalization. The ReLU vs. ELU comparison can be seen in Figure 6, we've selected the ELU which provided 0.005 loss vs. ReLU with 0.008 loss

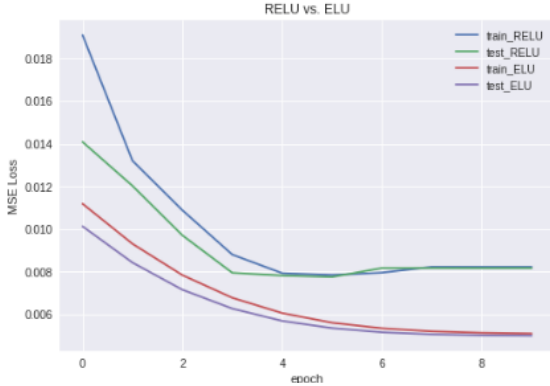


Figure 6 JointFilterNet activation function ReLU vs. ELU comparison

In Table 3 the experiment results can be shown, the error was calculated using eq. 2 and the number of epochs we run in each configuration was 10. To use the JointFilterNet on the test images we took patches of 64x64 but inserted into the output image only the center of the patch (32x32 pixels), this is to reduce any errors and to receive a clear overlap between patches.

Table 2 JointFilterNet optimization experiment results

Loss Function	Batch Normalization	Dropout	Error	JF Error/DenseMap Error
MSE	X	X	6.59	1.15
MSE	V	X	2.23	0.39
MSE	V	0.2	3.24	0.57
MSE	V	0.5	24.98	4.37
Binary	X	X	3.63	0.63
Binary	V	X	8.53	1.49
Binary	V	0.2	2.99	0.52
Binary	V	0.5	7.68	1.34
Logcosh	X	X	9.97	1.74
Logcosh	V	X	7.65	1.34
Logcosh	V	0.2	4.56	0.80
Logcosh	V	0.5	27.66	4.84

The full network was tested on 4 images which weren't used to train any model of the network. The average error (end point error [epe]) of the images after the DenseMapNet was 5.71 and in table 3 it's possible to observe the average error using different loss function, batch normalization and dropout after 10 epochs. From going over the table, it can be noticed that all loss function got a relatively good value in the error rate using batch normalization and dropout 0.2 (which weren't part of the original network design). The best result is while using MSE, Batch normalization and without dropout, improving the average error from 5.71 to 2.23. We can also see that if we used the model suggested in the article (without dropout or batch normalization), the network fails and actually adds noise to the image.

In Figure 7 and in table 4 we can see the network outputs, it's edge preserving quality and it's capability for spatial denoising, allowing surfaces in the same depth to receive similar

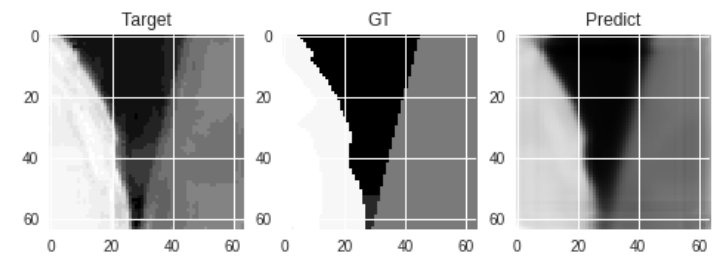


Figure 7 An example of our result, Target is the DenseMapNet output, GT is ground truth and Predict is the final image after JointFilterNet

6 CONCLUSION

6.1 DenseMapNet

Comparing to the original paper results (Table 5) we can state that we received similar is the errors summary from the original paper. For Monkaa dataset the error of DenseMapNet was 4.45, our network reached an error of 4.5. Due to the fact that we've trained our network on x50 smaller data, the comparison isn't correct, if by the fact that it may lead to overfitting or won't allow to minimize the loss and error.
















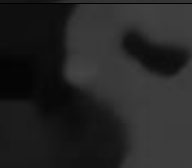
With that said, we believe that our results are close enough to the original article and are capable to allow proof of concept for the joint filter net and to show the use of the logcosh loss might improve results of the original article (if used on the complete data set).

Table 5

BENCHMARK ON DIFFERENT DATASETS. ALL ERRORS ARE END-POINT-ERRORS (EPE). BASELINE DATA ARE FROM [2].

Method	Sintel	Driving	FlyingThings3D	Monkaa	KITTI 2015	Parameters	Speed	GPU
DispNet	5.38	15.62	2.02	5.99	2.19	38.4M	16.67Hz	NVIDIA Titan X
SGM	19.62	40.19	8.70	20.16	7.21	-	0.91Hz	NVIDIA Titan X
MC-CNN-fast	11.94	19.58	4.09	6.71	-	0.6M	1.25Hz	NVIDIA Titan X
DenseMapNet	4.41	6.56	5.07	4.45	2.52	0.29M	≥30Hz	NVIDIA GTX 1080Ti

Table 4 JointFilterNet outputs

	Full image	Branch	head	tail
Left image				
GT				
Dense map output				
Joint Filter Net Output				

6.2 JointFilterNet

When we've proposed our model we targeted to improve the DenseMapNet results by post processing its Output using a novel approach of a neural net joint filter. We successfully reduced the error in half and improved the suggested model by adding batch normalization layers showing the potential of using the Joint filter net for denoising layer , improving contour and preforming additional post processing tasks.

7 DISCLAIMER

For this project we used a small data set to prove concept and feasibility of a suggested model. although we show improved results, it is important to state (as we did several times in this report) that the data set we used was small and although we separated the test data during all the process, the final results are an example of potential and expected to change (for better or worse) when using a different\larger data set.

8 REFERENCES

- [1] R. Atienza, "Fast Disparity Estimation using Dense Networks," 2018.
- [2] J.-B. H. N. A. a. M.-H. Y. Yijun Li, "Joint Image Filtering with Deep Convolutional Networks," 2017.
- [3] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.
- [4] "Fast Global Smoothing," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 23, 2014.
- [5] J. Kopf, M. F. Cohen, M. F. Cohen and M. Uyttendaele, "Joint Bilateral Upsampling," 2004.
- [6] "uni-freiburg website," [Online]. Available: <https://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16/>.
- [7] Y. L. Jure Zbontar, "Computing the Stereo Matching Cost with a Convolutional Neural Network," 2014.
- [8] Z. L. K. Q. W. a. L. v. d. M. G. Huang, "Densely connected convolutional networks," in *IEEE conference on computer vision and pattern recognition*, 2017.
- [9] R. Y. J. D. a. D. N. Q. Yang, "Spatial-depth super resolution," in *IEEE Conference on Computer Vision and Pattern*, 2007.
- [10] J. S. a. X. T. K. He, "Guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, 2013.
- [11] M. K. a. P. V. G. V. Jampani, "Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [12] C. C. L. a. X. T. T.-W. Hui, "Depth map super-resolution by deep multi-scale guidance," in *European Conference on Computer Vision*, 2016.
- [13] W. Z. S. G. Y. C. C. C. a. L. Z. S. Gu, "Learning dynamic guidance for depth image enhancement," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [14] X. Z. S. R. a. J. S. K. He, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [15] R. Manduchi and C. Tomasi, "Bilateral Filtering for Gray and Color Images," in *Proceedings of the 1998 IEEE International Conference on Computer Vision*, Bombay, 1998.
- [16] roatienza, "<https://github.com/roatienza/densemapnet>," [Online].
- [17] M. F. C. M. F. C. M. U. Johannes Kopf, "Joint Bilateral Upsampling," 2004.