A Project Report on:

"Depth estimation Using CNN's"

Submitted by:

Saikumar Dande (181EC140)

Chandravaran K (181EC156)

VII SEM BTECH (ECE)

Under the guidance of

Dr. Shyam Lal

Department of ECE, NITK Surathkal

in partial fulfillment for the award of the degree
of
Bachelor of Technology
in
Electronics and Communications
at

National Institute of Technology Karnataka, Surathkal

November 2021

Table Of Contents

1 Abstract

We present 2 new networks for depth estimation using CNNs. The networks are named O-net and Y-net. These models have been tested on the NYU dataset and compared with other models. The O-net model is shaped like an 'O' and uses U-net as a baseline and improved to get better results. The Y-net is an improvement from O-net to reduce the complexity of the model and is shaped like a 'Y'. Both these models achieve better than the base U-net model. The results obtained from these models are compared with others and noted the difference. All the models have been coded and run on Colab and local systems. The project aims to develop a model that can be used for robot mapping in real-time.

2 Introduction

Over the years, robots have gained significant importance and have helped alleviate some of the problems faced in many fields such as construction, medicine, and technology. For the robot to interact with its environment, it needs to understand what is around it and where it is. This is achieved through navigation and localization of the robot through the environment. 3D maps are used to help recreate the scene of the world.

One of the first steps is to calculate the depth of the objects present in the environment. There are a variety of methods generally used to obtain depth. These include LIDAR, IR, Stereo, Motion Based, and One Shot Structured light to name a few. LIDAR cameras provide accurate depth but are found to be sparse in vertical and horizontal resolution. IR cameras are often used to detect depth, however, they often have limitations in the distance they can measure and also run into problems when the environment contains shiny or transparent objects. Active illumination-based methods are found to either have a good acquisition time or a good resolution of the 3D shapes, but not both. One-shot methods are sensitive to the texture of the surface and can only recreate sparse reconstructions. In this paper, we have used images obtained from a stereo dataset.

While calculating depth, it is often convenient if the two images obtained from the stereo cameras only differ in the horizontal direction. To achieve this, cameras have to go through the processes of rectification and calibration. Stereo calibration is firstly done to remove any lens distortions that may exist. Further rectification is done to ensure that a pixel P if present at (x1,y) in the first image, can be found in the same horizontal line at (x2,y) in the second image. This is often achieved through photogrammetric calibration as shown in Figure 1, by observing a stationary object such as a chessboard and using rotation and projection matrices to rectify and calibrate the camera.

Once the images are obtained from the stereo cameras, they then need to be processed accordingly to obtain a disparity map. A disparity map calculates the apparent motion present for the same pixel in

the two stereo images. This is achieved through algorithms such as Census Transform, Sum of Absolute Differences, and Multi-Block Matching. The depth map can then further be calculated using the formula:-

$$depth = ((focal\ length) * baseline)/disparity \quad (1)$$

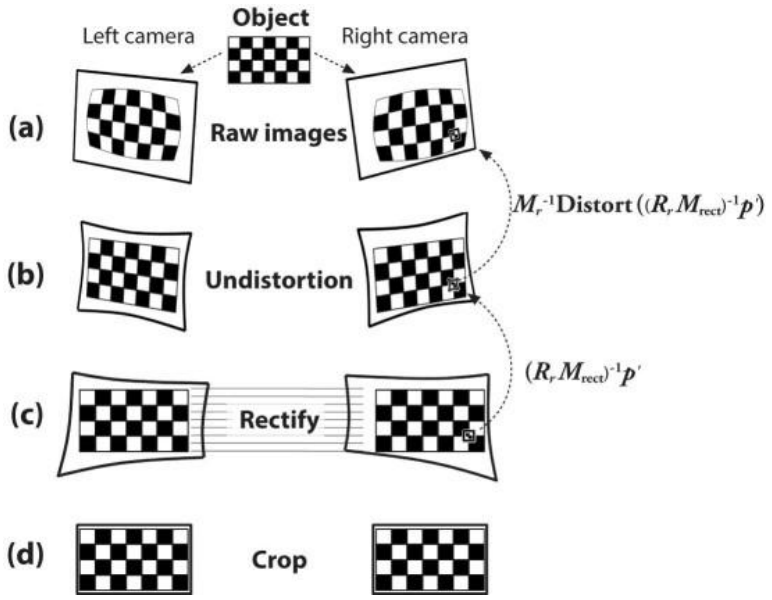


Fig. 1. Stereo camera rectification

The disparity maps further need to be converted into a 3D point cloud. A point cloud is a collection of pixels in a 3-dimensional space. The rotation and translation matrices are obtained from the images and can be further applied to point clouds to achieve a complete 3D reconstruction of the environment.

3 Literature Survey

In recent years, various methods have been developed to generate sparse and dense disparity maps. In order to generate a disparity map, for every pixel in the left image we need to find the corresponding pixel in the right image, and for that, there are various methods. Single block matching techniques use a certain number of pixels in the given block size to match the left and right pixels. In this project, we are using deep learning techniques instead of image processing techniques.

In the paper [2] they made a deep convolution architecture which consists of the coarse net and fine net. The Coarse net was used to predict the global depth map and then this global depth was refined using the fine net. The fine net uses the input image and concatenates with the global depth map to extra features and outputs the enhanced depth map. This model is a supervised model which uses only a single RGB image. In the paper [4], they developed an unsupervised method where they use stereo

images to train the model. The model takes the left image and generates the right image as output. The error for this model is calculated by comparing it with the input right image. Similarly, they do it for the right image. In this project, we develop a supervised model which uses a single RGB image as input and the ground truth corresponding to it.

4 DataSet

4.1 NYU Dataset

The NYU-Depth V2 data set is comprised of video sequences from a variety of indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect. It has 1449 densely labeled pairs of aligned RGB and depth images 464 new scenes taken from 3 cities 407,024 new unlabeled frames. The dataset is split into 1024 train, 201 test, and 224 validation. We downsampled the RGB images and ground truth depth by 4 times, from 640x480 to 160x120.

Fig. 2. NYU Dataset. Left is the image and Right ground truth

4.2 KITTI Dataset

The depth completion and depth prediction evaluation are related to work published in Sparsity Invariant CNNs (THREEDV 2017) It contains over 93 thousand depth maps with corresponding raw LiDaR scans and RGB images, aligned with the raw data of the KITTI dataset

Fig. 3. Kitti Dataset Images

5 Related Work

5.1 Previous year model

This Image processing technique was developed by us (Chandravaran and Saikumar) in the sixth semester under the course *Image and Video processing.* This method combines the techniques of SAD (Sum of Absolute Difference) and Census transform. It uses color images, greyscale, and gradient X, Y as inputs into the method. It combined the disparity output by using the Lagrange equation. As disparity can be considered as an error the equation works perfectly. Below are the results:-

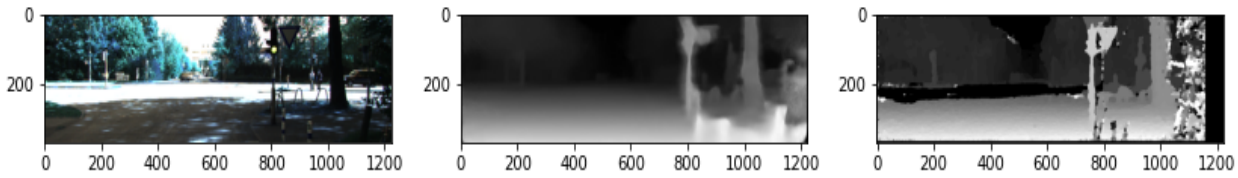


Fig. 4. Image processing technique

As we can see the values are high and it correlates to the very bright area and dark areas in the input images. As we can see in the predicted output, the right side of the image is lost and the interaction that the tree line meets the road is also lost, this is due to the high intensity of the light present in that region. Another huge drawback of this method is the large computation time, which gets worse as the image gets bigger. As there are too many drawbacks to this technique we have decided to develop a CNN model that will perform better.

5.2 Unsupervised Monocular Depth Estimation with Left-Right Consistency

This is an unsupervised method[4] that takes in a stereo camera and uses the consistency between both the images to get the output.

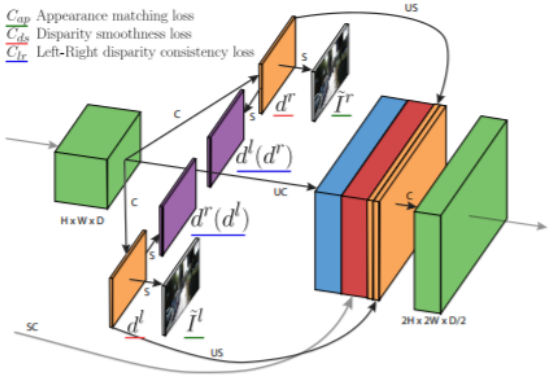


Fig. 5. Left and right consistency matching

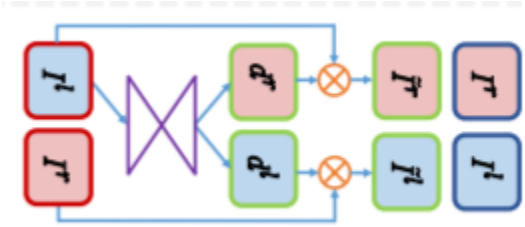
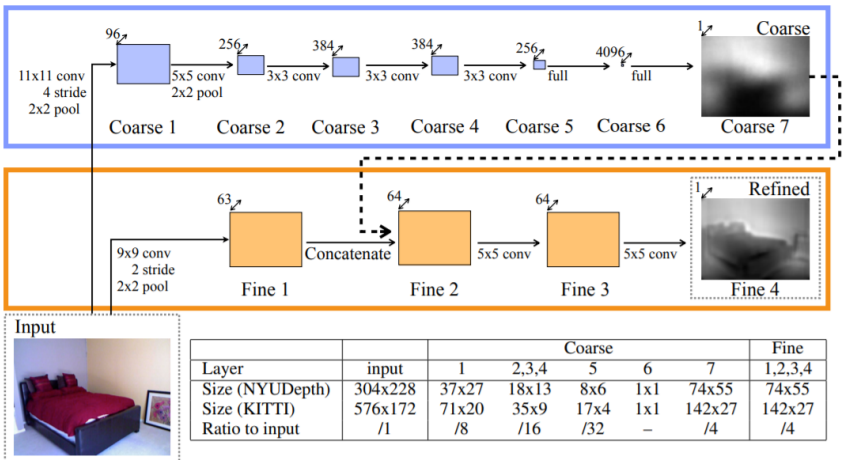


Fig. 6. Full network to get depth image

5.3 Course & Fine net

This network has 2 parts: Coarse net and fine net [2]. The Coarse net part predicts the depth of a scene at a global level. Fine net is used to refine within the local regions. The model can be seen below with the distinctive blocks.



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Fig. 7. Course & Fine net

The top portion of the network which is the coarse net, has continuous convolution layers that can be seen to give the depth information in the global reference as seen above. The coarse network contains 5 convolution and max-pooling layers and then 2 fully connected layers. The output of the course net is input into the second layer of the refine net to get local information and predict the depths. The final output of this model is 1/4th the size of the input. The layers use the ReLu activation function except for the coarse 7 layers, where they used a linear layer.

5.4 U-Net

Unet[3] is a common model used for many segmentation techniques, we can classify depth estimation as a segmentation problem. There are various modifications of this model like unet++ which are used for depth estimation. The model takes an RGB image as the input and at the output, we find scalar invariant error between the ground truth and predicted output. This model consists of two parts,

Encode and Decoder. The encoder part learns what features are there in the image and the decoder part learns where these features correspond to. This model uses skip connections to get the spatial information of the image. These skip connections are used by the decoder to reconstruct the image. Skip connections also help in solving the vanishing gradient problem.

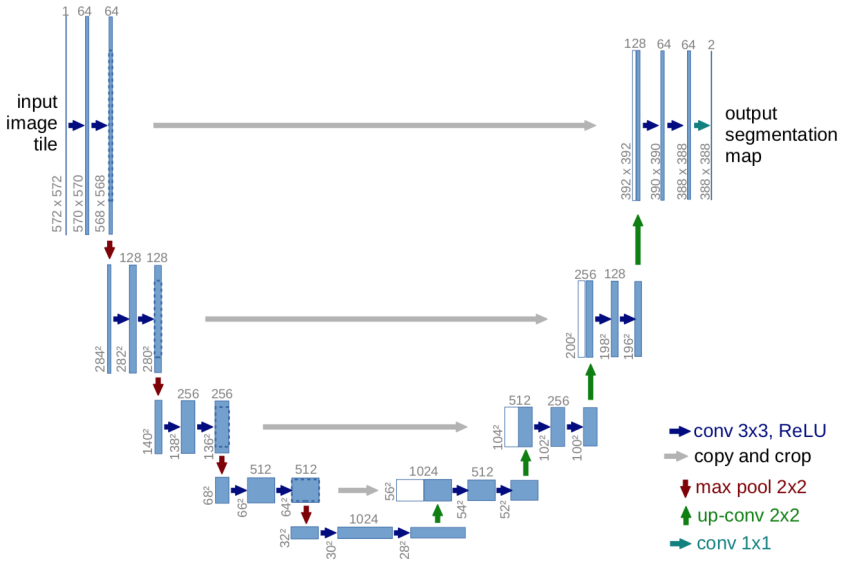


Fig. 8. U-Net

The left portion of the network is the downsampling part that is also called the encoder. This downsampling part consists of 5 double convolution blocks which consist of two 3x3 convolutions. After every double convolution, we apply 2x2 max-pooling of stride 2. At the bottom, we get a total of 1024 features, these are the features extracted by the U-Net. These features are upsampled and recombined with the output of the previous layer. Similar to the downsampling, in upsampling we use 5 double convolution blocks which consists of two 3x3 convolutions. After every double convolution, we apply up-sampling. This continues until the last layer where we finally use a 1x1 convolution to get the output.

6 Proposed Models

6.1 O-net

This model is based on the U-net. From our experience applying gradients for our methods improves the accuracy and gives us better results. But combining the gradients with the RGB image does not allow us to have specific weights for the gradients, we will only get common weights of the image. So to include the gradient images we passed them through a separate U-net So we get encoded data for the gradient images. However, the gradients are mixed with the RGB images in the upsample. This gives us encoded data and allows us to combine it with the RGB image to get a proper distinction of the objects in the image. The model's main flow gets represented as an 'O' so it is called O-net.

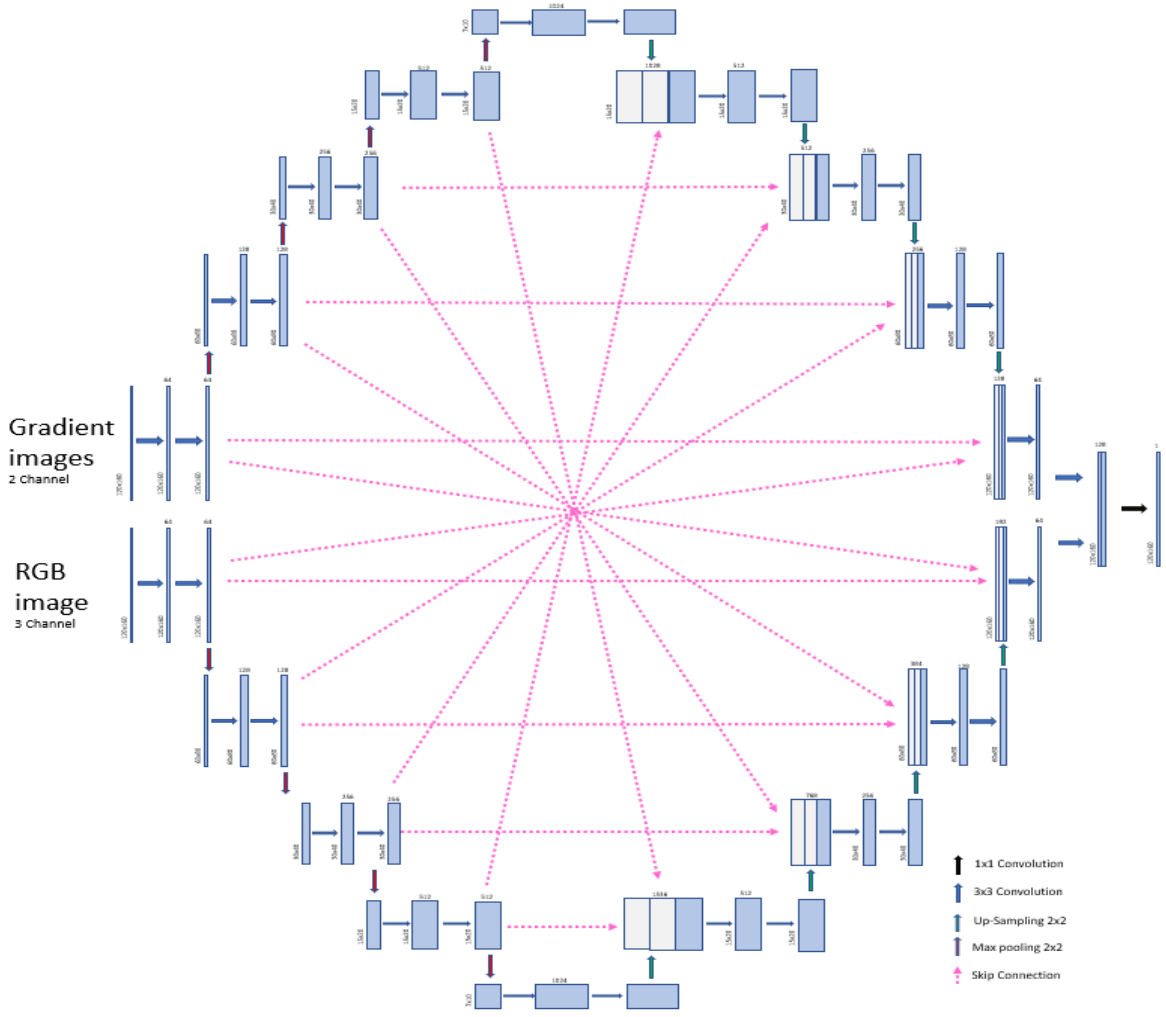


Fig. 9. O-Net

As you can see from the above picture it makes an 'O' shape. There are many skip connections and combinations of features in each layer. The top part is meant for the gradient images and the bottom part for the RGB image. The model summary has trainable parameters of approximately 68 million.

6.2 Y-Net

The reason we went on to make another model was to simplify the model by reducing both side upsampling. So we have 2 encoders one for the RGB images and the other for the gradient images. Then we have skip connections from the layers to the up convolution part. This simplifies the model and improves the execution time.

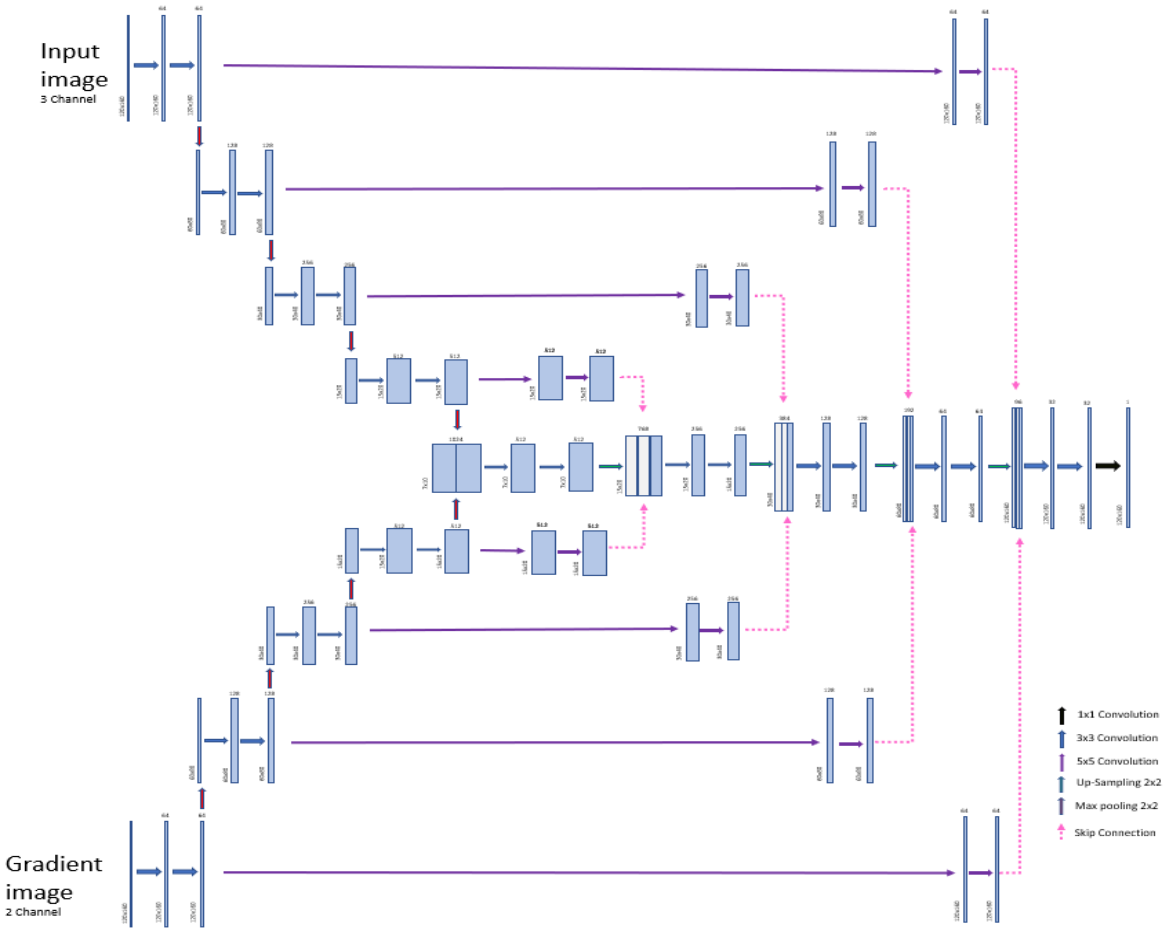


Fig. 10. Y-Net

From the above diagram the main flow of the images is from the branches to the central upsampling part that gives us a Y-shaped structure. We have added convolutions in the skip connections as well to take care of any transforms required before adding them to the decoder network. It has nearly 58 million trainable parameters. Instead of directly connecting skip connections like in Onet, we used residual blocks in between. These residual blocks consisted of two convolutional blocks each followed by a relu activation function. We used 5x5 kernel sizes in the residual block. At the end of the network, we use a 1x1 convolutional block to get the final output. The error is calculated by using the scalar invariant loss function which takes predicted and ground truth as input and this error is backpropogated. Skip connections help in solving the vanishing gradient problem.

6.3 Convolution Block

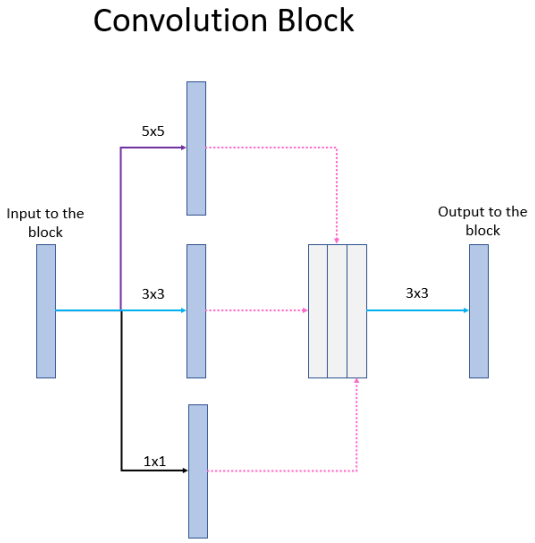


Fig. 11 Proposed Convolution Block

This is the new convolution block that we proposed. We replace the double convolution blocks which we used until now in O-Net and Y-Net with this new block. The idea of the block is taken from the Multiblock matching technique which we used in the image processing technique. In the multi-block matching, we use different block sizes like 3x3, 5x5, and 11x11 and use them to find the correlation between left and right images and then combine those correlations to predict the disparity value. So similarly in this convolution block, we use three different convolutions of sizes 1x1, 3x3, and 5x5, then concatenate the output of those three convolutions and apply a 3x3 convolution for it.

6.4 Scalar invariant loss function

In this project, we use scalar invariant error as a loss function. It measures the relationships between pixels in a given image. Let us consider the ground truth image as y* and predicted depth image as y, each containing n pixels, the scalar invariant mean squared error is defined as

$$D(y, y^*) = 1/2n \sum_{i=1}^{n} (\log y_i - \log y_i^* + \alpha(y, y^*))^2 \quad (2)$$

7 Results

7.1 Multi-block Matching Technique

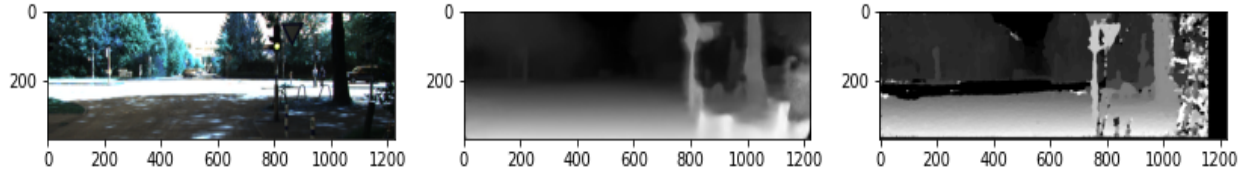


Fig. 12. Input image (left), ground truth (middle), and predicted (right)

In this method we can see that there is a huge problem for the algorithm to work properly when there is a very bright area or when it is very dark, to improve this we are designing a deep convolutional network.

7.2 Course and Fine net

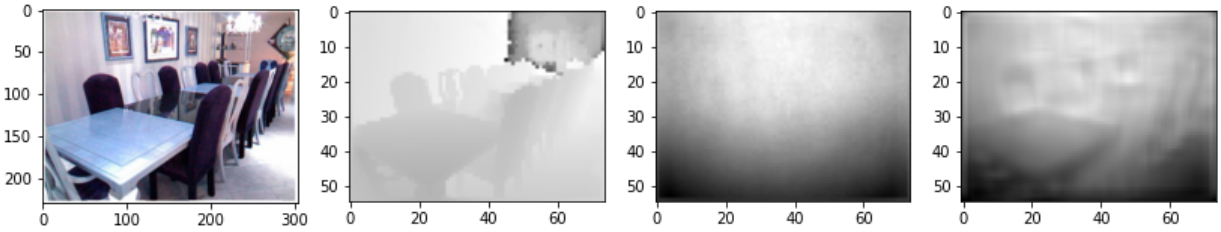


Fig. 13. Input image (left), ground truth (middle), and predicted(right)

The second image is the ground truth, the third image is the course net output, it basically gives us global depth. The fine net is used to give local depth. The output is smudged.

7.3 U-Net

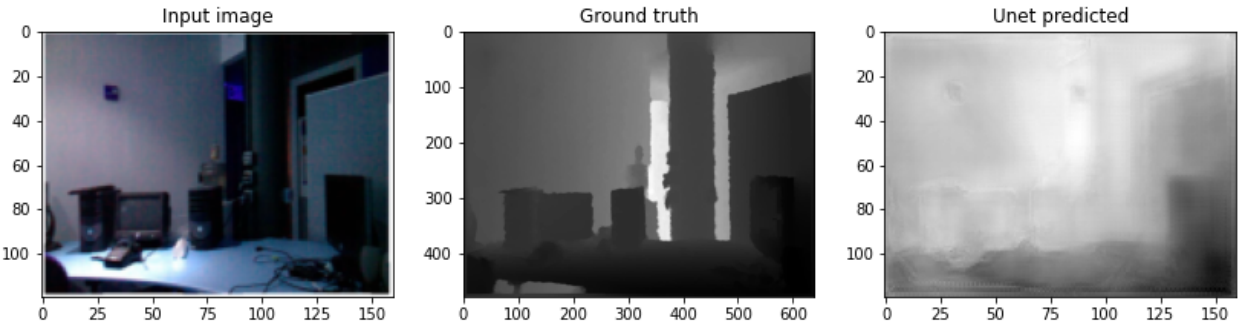


Fig. 14. Input image (left), ground truth (middle), and predicted(right)

As seen above U-net method the objects are not seen well and everything is very smooth, even the output we can't see the outputs.

7.4 O-Net

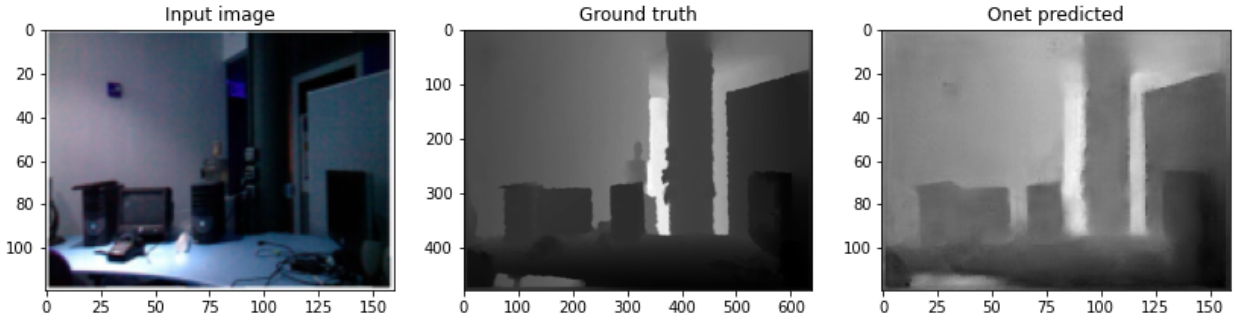


Fig. 15. Input image (left), ground truth (middle), and predicted(right)

This proposed model shows us great improvement compared to the previous models the objects are clearly visible, The laptops are properly distinguishable.

7.5 Y-Net

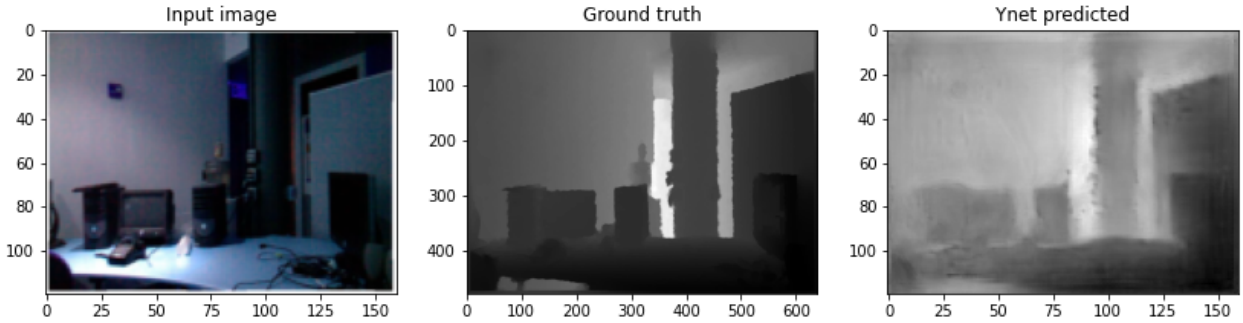


Fig. 16. Input image (left), ground truth (middle), and predicted(right)

This is the faster model it can be seen that the model has a lot of noise but the objects are clearly visible.

7.6 Y-Net + Proposed Convolution Block

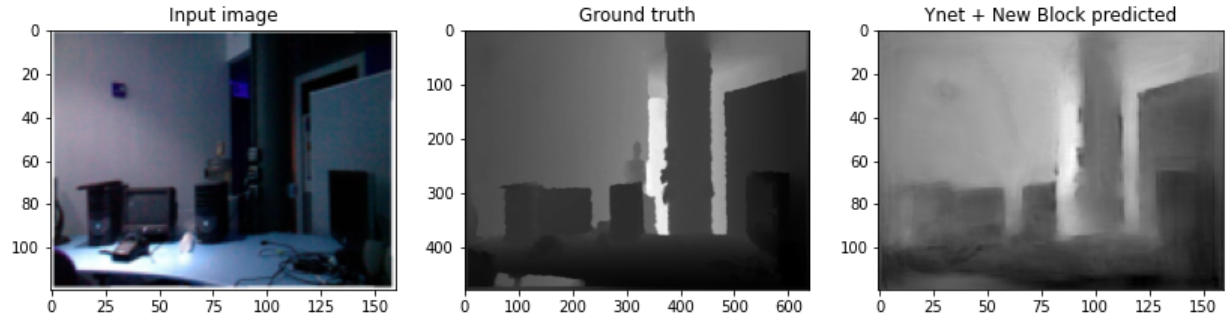


Fig. 17. Input image (left), ground truth (middle), and predicted(right)

This is the model with the new convolutional block the noise has been removed.

7.7 Comparison

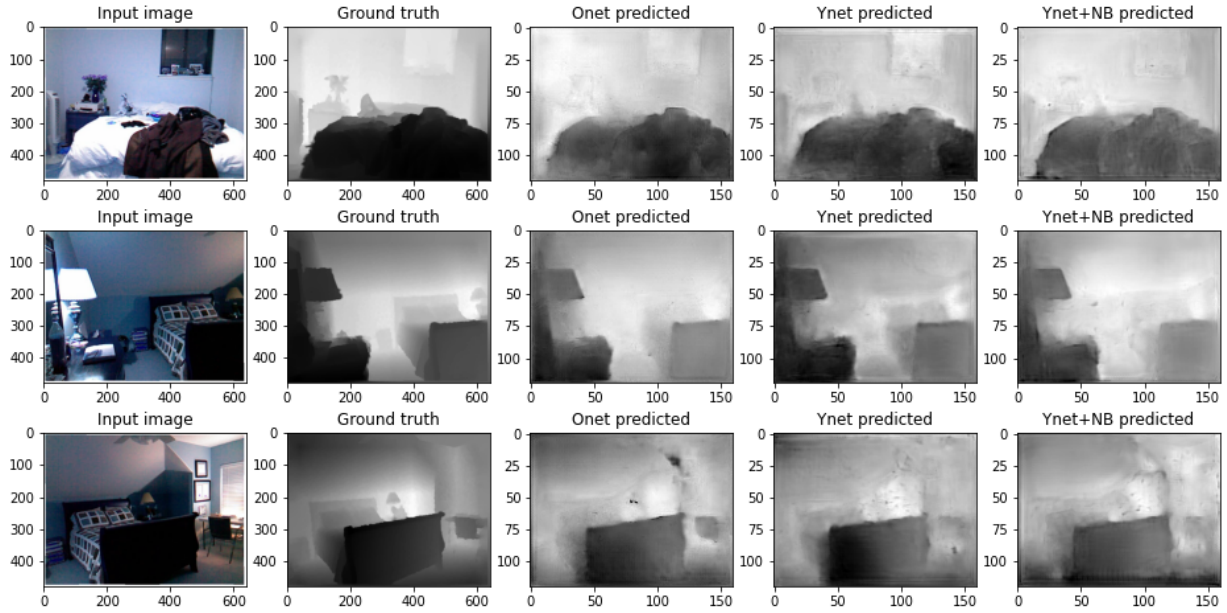


Fig. 18. Comparing results of Onet, Ynet and Ynet + Proposed convolution block

Models	Delta 1 (<1.25)	Delta 2 (<1.25^2)	Delta 3 (<1.25^3)	RMSE Linear	RMSE Log	abs relative	Square relative
Coarse+Fine (Paper)	0.611	0.887	0.971	0.907	0.285	0.215	0.212
Coarse+Fine (Trained)	0.5230	0.8268	0.9470	0.8283	0.1415	0.3831	0.5373
U-net	0.5625	0.8605	0.9571	0.7873	0.1278	0.3582	0.6016
O-net	0.6298	0.8984	0.9704	0.6924	0.1057	0.3125	0.5097
Y-net	0.6560	0.9092	0.9725	0.6652	0.1009	0.3074	0.5118
Y-net + NB	0.6745	0.9119	0.9772	0.6537	0.0985	0.2971	0.4910

Table 1. Results of all the models

From Table 1 we can see that Ynet has better accuracies (delta1, delta2, and delta3) compared to the other models. In Fig 18, we can see that Onet output is much clear and sharper and Y-net output is blurred and there is some noise in the Y-net output. When we used the proposed convolution block in the Y-net, we can see improved better results compared to O-net.

7.8 Computation time for proposed models

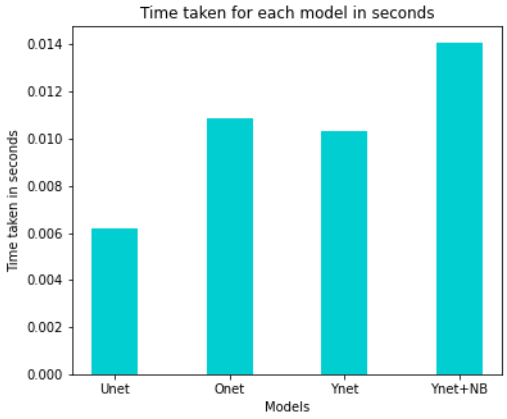


Fig. 19 Time bar graph

From the above graph it can be seen that O-net takes more time compared to Y-net but Y-net gives similar results to O-net. However, Y-net + NB is a little slower but has better results compared to O-net, so based on our requirements we can choose the network to be used.

7.9 The number of Parameters

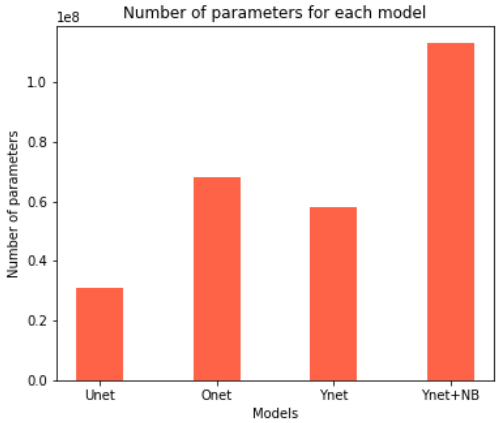


Fig. 20 Parameter bar graph

As expected Y-net has fewer number of parameters compared to O-net as we have reduced one up-convolution side. However, as in the new convolution block, we have increased the number of

convolutions by 3 times the number of parameters for Y-net + NB is more, but the amount of computation time is not huge.

8 Conclusion and Future Work

We plan to improve the model by using atrous convolution as part of the new-convolution block. We also want to use the full image size for predicting the depth if we have enough computational power for it. We will also try to deploy this in a real-life scenario using a webcam, also add preprocessing for real-world images.

9 References

[1] Dataset: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

[2] David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth map prediction from a single image using a multi-scale deep network. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 2366–2374

[3] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[4] Godard, Clement & Aodha, Oisin & Gabriel, Jourdan. (2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. 10.1109/CVPR.2017.699.