J. Pokojski et al. (Eds.)

doi:10.3233/ATDE200110

Customized Synthetic Dataset for Deep Learning Noise Filtering for Time-of-Flight Indoor Navigation Applications

Vinícius da Silva RAMALHO, Rômulo Francisco Lepinsk LOPES, Ricardo Luhm SILVA and Marcelo RUDEK
Pontifícia Universidade do Paraná (PUCPR), Brazil

Abstract. Synthetic datasets have been used to train 2D and 3D image-based deep learning models, and they serve as also as performance benchmarking. Although some authors already use 3D models for the development of navigation systems, their applications do not consider noise sources, which affects 3D sensors. Time-of-Flight sensors are susceptible to noise and conventional filters have limitations depending on the scenario it will be applied. On the other hand, deep learning filters can be more invariant to changes and take into consideration contextual information to attenuate noise. However, to train a deep learning filter a noiseless ground truth is required, but highly accurate hardware would be need. Synthetic datasets are provided with ground truth data, and similar noise can be applied to it, creating a noisy dataset for a deep learning approach. This research explores the training of a noise removal application using deep learning trained only with the Flying Things synthetic dataset with ground truth data and applying random noise to it. The trained model is validated with the Middlebury dataset which contains real-world data. The research results show that training the deep learning architecture for noise removal with only a synthetic dataset is capable to achieve near state of art performance, and the proposed model is able to process 12bit resolution depth images instead of 8bit images. Future studies will evaluate the algorithm performance regarding real-time noise removal to allow embedded applications.

Keywords. Synthetic Dataset, Noise Removal, Time-of-Flight Image

Introduction

Human beings rely on their eyes to navigate through complex scenarios and avoid obstacles. When someone suffers from low-vision or blindness, their navigation ability gets compromised. Machines, on the other hand, uses a camera and other types of sensors to be able to perform autonomous navigation tasks. Differently from our eyes, sensors are not adaptable to complex lighting. Due to manufacturing limitations these sensors interpret some of these complex conditions as noise, causing autonomous navigation to work similarly to blindness or low-light vision.

Autonomous navigation applications require devices that are capable to extract data from the environment and transform them into a valid path. There are three main technologies that use computer vision to perform 3D reconstruction of the environment: stereo vision, Time-of-flight (ToF) camera, and LIDAR. ToF and LIDAR use logic similar to a sonar, but with a modulated near-infrared light. The stereo vision uses a pair of calibrated cameras and through epipolar constraints measure distance by matching the corresponding pixels from one image to the other. Stereo vision is usually cheaper, but is a more software-intensive task and requires great computational power to process the depth image, with the most accurate algorithms taking up to minutes to process a single pair of images and faster algorithms perform poorly on textureless regions. On the other hand, ToF cameras process everything in hardware, needing less computational power makes it a more attractive solution for embedded applications. Although LIDARs are more accurate and data processing is considerably fast, the acquisition cost is a huge limitation for several indoor applications or when a product requires production scalability.

One of the challenges of these technologies is to evaluate their depth measurement from each equipment and compare them with ground truth data. Ground truth with real data depth images is harder to obtain because ToF and Stereo normally contain a high noise level and LIDAR requires intensive scanning to generate dense accurate data, otherwise it will sparse. Since LIDAR has the most reliable performance, it is used to generate ground truth information, but due to LIDAR technical limitations, the dataset is usually sparse. Kitti dataset[1] is one example of a sparse ground-truth dataset.

Synthetic datasets become a very attractive solution to approach the lack of ground truth data, as the depth map generated is noiseless and dense, like on the Flying Things dataset [2] used for train/test stereo matching methods. Deep learning models are usually trained with real-world data or a combination of real-world and synthetic data. Since ToF real-world data is hard to obtain, this paper proposes an architecture that is trained only with the Flying Things synthetic dataset and the addition of synthetic Gaussian noise. The trained model is validated with the Middlebury dataset [3] which is a real-world source of the dataset.

1. Technological background

A ToF camera is an active sensor that uses the speed of light to calculate distance. The ToF camera mechanism measures the phase shift between the emitted modulated light and the received light which should be proportional to the distance between the emitter and the reflective object.

1.1. ToF noise and common filtering techniques

Like any other camera sensor, ToF camera is susceptible to undesirable effects like artifacts, unrealistic edges, blurred objects and disturbs background scenes caused by noise [4]. The source of noise can vary, as its statistical behavior, from the thermal vibration of the atom and discrete nature of radiation to the amplitude quantization process [5].

To attenuate the effect of noise, several traditional computer vision filters can be divided into two classes, spatial and temporal. Spatial filters use neighboring pixel values to filter noise. The most common are the mean, the median [6], and the bilateral filter [7]. However, temporal filters use the actual and past values of the same pixel to filter noise. The mean, the median, and the impulse response filter are the most common temporal filters.

Their effectiveness depends on the type of noise, but they all have disadvantages. Spatial filters are known to decrease image sharpness and temporal filters cause motion blur when objects are moving fast.

Combined computer vision algorithms to filter noise can achieve good results and still maintain some image structural integrity, but they are tailor-made for specific environments and they lack the ability to adapt to other scenarios. Recent deep learning architectures that use an encoder-decoder module are being applied to image reconstruction and image denoising tasks. These architectures can achieve great denoising performance while keeping image integrity [8].

1.2. Deep Learning

The base of all deep learning theory comes from the artificial neural networks which are made by a number of interconnected artificial neurons. The artificial neuron, as shown in Figure 1, perform a set of linear operations, equations, and then apply a non-linear behavior at the output signal using an activation function.

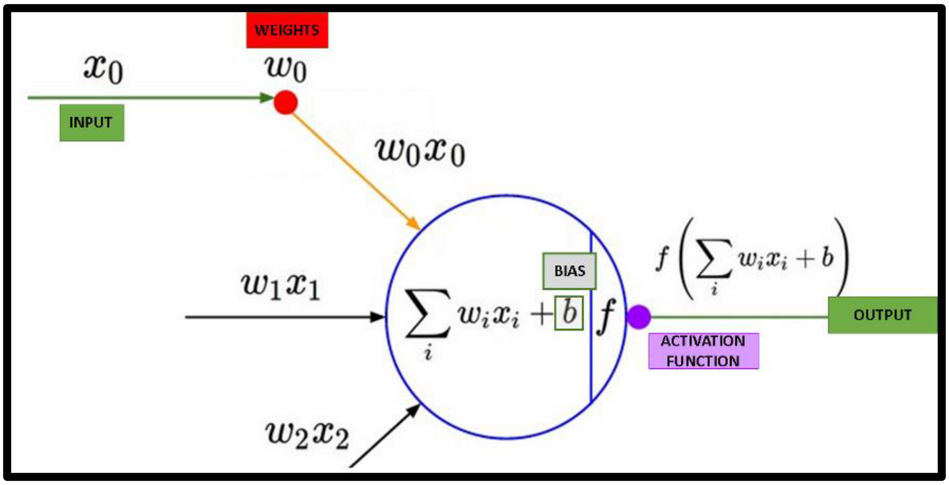

Figure 1. Example of an artificial neuron.

A simple artificial neural network is a supervised algorithm, which needs a dataset with known inputs and desired outputs. Any network must have an input and output layer and a variable number of hidden layers, which will depend on the selected architecture. Through backpropagation and optimizing technics, like gradient descent, to adjust the weights and minimize the error between the output and the desired value. The deep learning term refers to a neural network which has a great number of hidden layers, and Convolutional Neural Network (CNN) is one specific type of deep learning architecture.

CNNs refers to a neural network that uses convolution operations to extract features. The kernels used in the convolutions have the weights parameters embedded, in other words, the CNN learns which are the weight values for each kernel that extract features best.

A large number of architectures were developed for different types of applications: segmentation, object detection, stereo matching, image completion, image creation, image denoising, and many others. The encoder-decoder architecture is commonly used to image denoising. U-Net[9] is an example that uses the encoder to compresses the image into a latent space while reserving the primary components of objects in the image, thus eliminating noise. The decoder reconstructs the image from the latent space.

One of the very first encode-decoder networks used for image restoration was RED-Net [10] which showed that a deep learning model using downsampling and upsampling philosophy with convolutions, deconvolutions (transposed convolution) and skip connections can attenuate noise and still keep image detail. Some architectures also use shared parameters between convolutions and transposed convolutions of the same level [11, 12].

1.3. Synthetic Noise and Quality metric

Deep Learning models are most likely to be trained with real colored/greyscale images with noise added artificially, usually, Gaussian distributed noise with the mean value equals zero. The same principle can be applied to depth images, however, instead of 8bit per channel depth maps are normally higher. As deep learning is normally a supervised algorithm, the ground truth is needed, that is why the synthetic data for depth measurement is necessary as one of the most feasible ways to acquire noiseless images and also the reason why noise is added artificially.

To measure an algorithm’s performance Peak Signal to Noise Ratio (PSNR) is a very common metric to validate the filter quality. PSNR (in dB) is defined as follows

$$\text{PSNR}=10*\log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right) \tag{1}$$

Where MAX is the maximum possible intensity value.MSE is the mean squared error. The MSE function is also defined by:

$$\text{MSE}=\frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \tag{2}$$

Where I(x,y) is the intensity value at pixel (x,y) in the noiseless image, k(x,y) is the intensity value at pixel (x,y) in the noisy image.

2. Methodology

One of the most important parts to train a deep learning model is the dataset selection. The desired aspects of the dataset are to be large, noiseless information, and a dense depth map, meaning that all pixels have a depth value. The dataset that checked all the aspects is the Flying things dataset which is selected for training and to test/validate the model the Middlebury dataset is selected. As those datasets are used in stereo matching tasks the ground truth is a disparity map but can be converted into depth map by using the following equation.

To simulate real hardware, the distances are then normalized between 0 and 4095 (where 4095 is 18m) to imitate a 12bit analog to digital converter. The equation to convert disparity values extracted from stereo matching datasets is:

$$Z = f\frac{T}{d} \tag{3}$$

Z is depth, f is focal length, T is the distance between cameras, and d is disparity value.

Like many papers did, the noise is added artificially by summing a random image, Gaussian distributed, with zero mean and 255 standard deviations. The Gaussian noise was kept to test and validate the approach, but if another type of noise distribution is detected in future studies, the proposed model can be adapted.

3. Proposed Architecture

In high-quality traditional stereo matching algorithms after the matching step, the computed disparity is further processed by optimizing it by minimizing an energy function. The equivalent method for a CNN implementation developed for stereo matching tasks is the regularization steps which, usually, consist of encoder-decoder architectures to enhance and refine the disparity map. Supposing that the image from a TOF camera can be enhanced with the same method we propose the usage of stacked UNets with skip connections similar to PSMNet [13], but with 2D convolutions and residual bottleneck blocks. The proposed architecture, Table 1, is built upon a BasicConv which is a sequence of 2D convolution, batch normalization, and Rectified Linear Unit (ReLU) activation function.

Table 1. Proposed architecture.

LAYER/BLOCK	INPUT	KERNEL
BasicConv	WxHx1	Size=3x3,Kernels=128, Stride=1
BasicConv	WxHx128	Size=3x3, Kernels=128, Stride=2
BasicConv	W/2xH/2x128	Size=3x3, Kernels=128, Stride=1
Res_Unet	W/2xH/2x128	Stride=2
Res_Unet	W/2xH/2x128	Stride=2
Res_Unet	W/2xH/2x128	Stride=2
BasicConv	W/2xH/2x128	Size=3x3, Kernels=128, Stride=1
BasicConv	W/2xH/2x128	Size=3x3, Kernels=128, Stride=1
BasicDeconv	W/2xH/2x128	Size=3x3, Kernels=128, Stride=2

3.1. Implementation details

All codes were developed in python 3.7 in a Ubuntu 16.04 machine, OpenCV 4.1 was used to load/manipulate images, and Pytorch 1.0.1 was used to build and train the CNN. The proposed network was trained with logcosh loss, adam optimizer, a learning rate of 0.0001, and a batch size of 16. The loss function is given below:

$$Loss(y, y^p) = \sum_{i=1}^{n} \log\left(cosh\left(y_i^p - y_i\right)\right) \tag{4}$$

Where y is the ground truth value and y_p is the predicted value.

As the GPU memory is limited the images were cropped into smaller 92x92 patches to build the training batches. Also, all images were divided by 4095 to normalize them to values between 0 and 1.

The training section was not enough to complete an epoch (going through the whole dataset), as the dataset contains more than 20000 960x540 images it would take too long, and was trained for 13920 steps (about 4450 images). All training and validation were implemented on a Core i7 8750H, GTX 1050Ti 4GB with 16GB of RAM laptop.

4. Results and discussion

As described before, the Middlebury dataset was selected to validate the model performance. A total of 21 images were used for validation, similarly to the training step noise is added artificially by summing a random image following a gaussian distribution, however this time a seed was used to generate the same numbers, noisy images, making it fair to compare between checkpoints and to validate the model improvements during train season. Table 2 contains all the results with mean absolute error (MAE) in millimeters, MAE without considering the edges given in millimeters, PSNR in dB, and PSNR not considering the edges in dB.

Table 2. Validation performance results.

Image number	MAE [mm]	MAE without edges[mm]	PSNR[dB]	PSNR without edges [dB]
0	**757.8**	**553.0**	**23.5999**	**27.7506**
1	**664.2**	**548.1**	**25.7641**	**28.8376**
2	**604.8**	**527.7**	**27.1901**	**29.1806**
3	**901.8**	**674.5**	**21.7431**	**25.2678**
4	**606.6**	**487.4**	**25.6854**	**29.3118**
5	**581.4**	**444.4**	**25.3539**	**28.9093**
6	**603.0**	**576.1**	**27.9423**	**28.6548**
7	**684.0**	**544.9**	**23.5617**	**26.5834**
8	**610.2**	**520.8**	**25.7491**	**28.3964**
9	**608.4**	**506.4**	**26.3715**	**29.1668**
10	**831.6**	**523.2**	**21.9359**	**27.7333**
11	**770.4**	**550.4**	**23.7021**	**27.9718**
12	**781.2**	**586.3**	**24.1689**	**28.0893**
13	**648.0**	**546.0**	**26.0417**	**28.8634**
14	**676.8**	**569.0**	**25.0692**	**27.8267**
15	**536.4**	**440.3**	**25.7491**	**28.8105**
16	**583.2**	**499.3**	**27.0577**	**29.5701**
17	**770.4**	**602.2**	**23.4323**	**27.3206**
18	**835.2**	**646.2**	**22.9760**	**26.7951**
19	**603.0**	**484.1**	**25.7382**	**29.6685**
20	**680.4**	**546.4**	**25.2831**	**28.5535**

The general PSNR is 24.64dB and the general PSNR without edges is 28.11dB which performance is bellow when compared to other image denoise papers that achieve 30dB, however, it is a preliminary result to validate the concept of using deep learning to denoise depth images with 12bit resolution. Figure 2 illustrates, the CNN does filter noise at the cost of the image's structural integrity and shape.

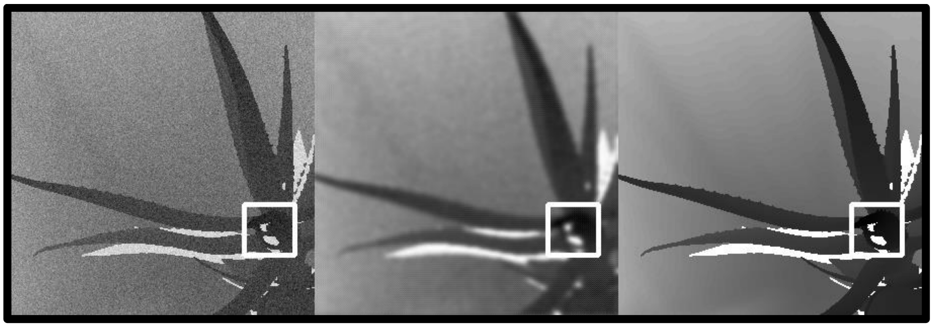

Figure 2. Noisy Image Input (left), Filtered Image (middle), Ground Truth Image (Right).

The decreased image output sharpness is probably caused by downsampling the resolution every time a convolution layer with stride 2 is used. This process reduces the required information for the reconstruction when a transposed convolution is operated, causing the blur effect in the output. The blur-like effect can be better observed in Figure 3, which is the detailed image of the white box in Figure 2.

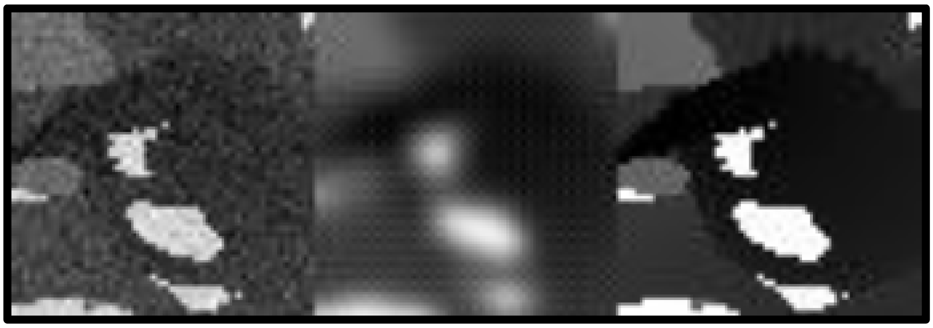

Figure 3. Detailed Images: Noisy Image Input (left), Filtered Image (middle), Ground Truth (right).

Subtracting the filtered image from the ground truth image and taking the absolute value of each pixel an error mask is made. Converting the error mask to depth error a heat map image is created to highlight regions with higher error. Figure 4 shows the depth error heat map results with and without edges.

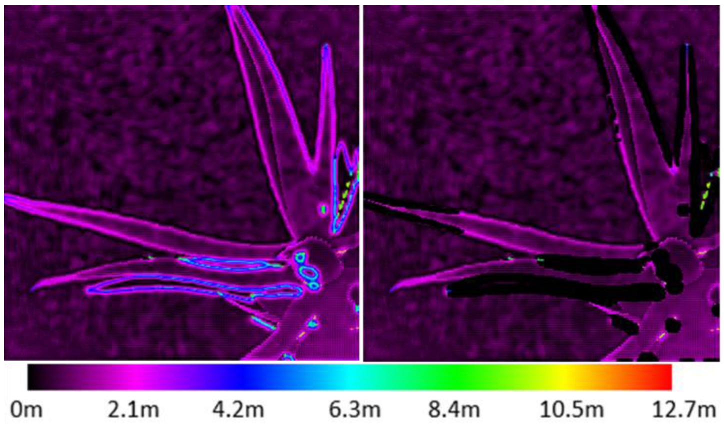

Figure 4. Depth error heat map image (Left), Depth error heat map image without edges(Right).

Figure 5 shows the depth error heat map of the same crop considered in Figure 3.

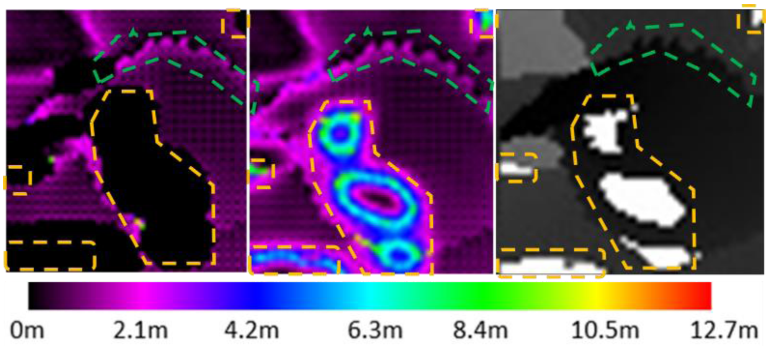

Figure 5. Depth error heat map image without edges (Left) Annotated heat map with edges (Middle), Annotated ground truth (Right).

The regions with dotted yellow lines annotation are the regions where the model performed worst, with errors ranging between 2,1m (purple), 4,2m (blue), and 8,4m (green). This type of event occurred in the transition from white parts, which depth correspond do 18 meters, to darker colors (which depth is near 0m) when analyzing the ground truth image. The error in the green region was smaller, ranging from 0 (black) to 2,1m. The model had the worst performance in regions with the transition to white spots. Because the Middlebury is a dataset extracted from stereo vision data, the white color is considered as a region where the matching algorithm could properly calculate the true depth, therefore considering the depth as the maximum value, which is 18 meters.

5. Conclusion and future works

In this work, the problem of ToF noisy depth images and the difficulty to acquire accurately noiseless depth images were approached, as a solution, a methodology using synthetic dataset and convolutional neural networks was proposed. The novelty of this work was to adapt the denoising approach which is commonly used for an 8bit image and apply it to 12-bit depth images. It was possible to simulate real ToF hardware by converting disparity generated maps to depth maps, and then normalizing the output to 12bit images. A CNN was trained with artificial images Gaussian noise using and then used for reducing noise in a real-world dataset with known ground truth.

The preliminary results pointed out that filtering performance did not achieve the state of the art performance and image output tends to blur. Spatial information seems to be lost during convolutions with stride 2, so a larger kernel to capture more spatial information could be a good option to implement and potentially reduce the blur. The worst performance was detected on the transition of objects where the depth gradient was high, and this hypothesis was confirmed by the removal of the information on the edges, which increased the performance of the final solution. Modifying the network do semantic segmentation, to detect these transition regions can be an option to improve the filtering capability.

The proposed architecture was able to mitigate some of the noise but it can be improved by optimizing hyperparameters and integrating semantic segmentation. The results also confirm the possibility of using only synthetic generated depth maps. Future works will also include the test of the model in a real scenario with a rear ToF camera to validate the proposed architecture. This will increase its applicability to ToF equipment for complex navigation scenarios, either indoor or outdoor.

References

[1] A. Geiger et al. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 2013, Vol. 32, n. 11, pp. 1231-1237.

[2] N. Mayer et al. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 4040-4048.

[3] H. Hirschmuller, D. Scharstein, Evaluation of cost functions for stereo matching. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2007. pp. 1-8.

[4] V. Rani, A brief study of various noise model and filtering techniques. *Journal of global research in computer science*, 2013, Vol. 4, n. 4, pp. 166-171.

[5] A.K. BOYAT, B.K. Joshi, A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.

[6] P. Patidar et al. Image de-noising by various filters for different noise. *International journal of computer applications*, 2010*,* Vol. 9, n. 4, p. 45-50.

[7] A. Gadde, S.K. Narang, A. Ortega, Bilateral filter: Graph spectral interpretation and extensions. In: *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 1222-1226.

[8] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention.* Springer, Cham, 2015. p. 234-241.

[9] Y. Zhang et al. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* arXiv:1812.10477, 2020.

[10] X. Mao, C. Shen, Y.-B. Yang, Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: *Advances in neural information processing systems.* 2016, pp. 2802-2810.

[11] S. Lefkimmiatis, Non-local color image denoising with convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3587-3596.

[12] S. Lefkimmiatis, Universal denoising networks: a novel CNN architecture for image denoising. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3204-3213.

[13] J.-R. Chang, Y.-S. Chen, Pyramid stereo matching network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410-5418.