Neural Wave Time-of-Flight: A novel depth recovery procedure for time of flight

Zhicheng Gu[b], Zhenyu Zhang[b], Nikhil Nakhate[a], Felipe Gutierrez-Barragan[a]

[a] *1st year graduate students*
[b] *2st year graduate students*

Abstract

Time of Flight (ToF) refer to the time that a light pulse takes to travel from a source to a target scene back to a detector. This technique is often used to recover scene depth and geometry. Continuous-Wave ToF (CW-ToF) is one particular low-cost ToF setup where the light source intensity and sensor exposure are temporally modulated by a modulation ($M(t)$) and demodulation ($D(t)$) function, respectively. In this scenario, multiple brightness measurements are obtained using a set of K modulation and demodulation functions. In conventional CW-ToF setups, where $M(t)$ and $D(t)$ are sinusoidal functions there exists analytical expressions to solve for scene depths. For arbitrary $M(t)$ and $D(t)$ functions, however, an analytical expression might not be available so a lookup brightness table for candidate depths is constructed and searched over. This is a computationally expensive procedure where the construction needs to be done per scene and the search is done for every pixel. In this report, we formulate the depth estimation procedure ToF brightness measurements as a regression problem. We propose two light-weight neural network architectures to solve this regression problem. We report scenarios where both architectures are able to match and sometimes outperform the lookup table/analytical approach.

Keywords: Time of Flight, Neural Networks, Fully Convolutional Networks

1. Introduction

In CW-ToF systems the amount of light sent to the scene is modulated by a modulation function, M(t). Further, the amount of light that is allowed into the sensor is determined by a demodulation or exposure function, D(t). Conventional CW-ToF systems use sinusoidal functions for M(t) and D(t), which are suboptimal. Recent work has shed some light on the design of optimal sets of functions that achieve higher depth resolutions. In practice, there is a resolution-speed trade-off when utilizing simple sinusoidal coding functions with low depth resolution vs. more complex coding functions that achieve higher resolutions. To compute depth from sensor/brightness measurements in a sinusoidal coding scheme there exists a few known analytical expressions. Once we introduce different frequencies and more complicated waveforms this analytical relation disappears. In these scenarios, depth is computed by searching a large lookup table that relates depths to brightness measurements. Constructing the table needs to be done per scene, and a lookup is performed per pixel. The computational burden imposed by these lookups renders these novel techniques not practical.

Recent work by Adam et al. (2017) formulates the depth recovery process from brightness measurements as a generative probabilistic model. By reformulating the problem as a regression problem they are able to adhere to the tight computational constraints of real world applications without sacrificing improved depth resolution.

Similar to Adam et al. (2017) we formulate the depth recovery process from brightness measurements as a regression problem. We present two neural network architectures that are able to accurately and efficiently compute scene depths from brightness measurements. The first architecture is composed of fully connected layers and performs per-pixel inference, in other words, given the brightness measurements for a single pixel it predicts the depth. The second architecture is a Fully Convolutional Network (FCN) which performs full depth map inference for a whole scene. In other words, the FCN takes as input the brightness measurements for each pixel in the scene and outputs the depth map.

In order to train the neural networks we developed a physically accurate ToF simulator. We leverage this simulator to generate a 100,000 sample dataset for the single-pixel network, and a 5,000 sample dataset for the FCN.

2. Methods

2.1. Preliminaries

In CW-ToF the light source and sensor exposure are temporally modulated. The radiant intensity (i.e intensity of light) emitted by the source is determined by a periodic *modulation function*, $M(t)$. The sensor exposure is modulated by a periodic *demodulation function*, $D(t)$. The light reflected from the scene and incident on a sensor pixel (p) will be a scaled, phase shifted, and vertically shifted version of the $M(t)$ denoted as the incident radiance, $L(p,t)$ (see equation 1. The brightness, $B(p)$, measured at a sensor pixel is the temporal correlation between $L(p,t)$ and $D(t)$. Note that the scene depth, Γ, is encoded in the temporal shift (phase shift, $\phi = \frac{2\Gamma}{c}$) of in $L(p,t)$. This process is illustrated by Figure 1.

$$L(p,t) = \beta(p)M(t - \frac{2\Gamma}{c}) + L_a \tag{1}$$

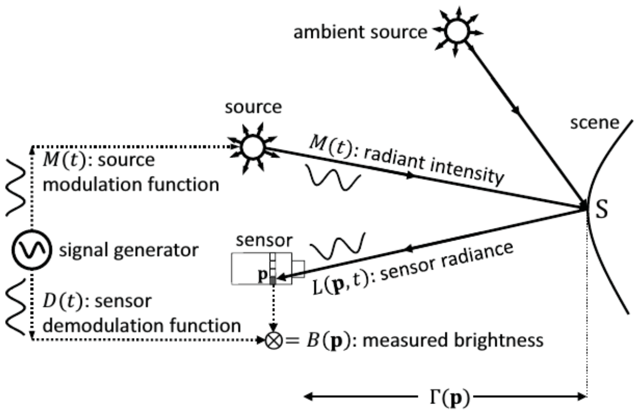

ambient source
source
scene
$M(t)$: source modulation function
$M(t)$: radiant intensity
S
signal generator
sensor
p
$L(\mathbf{p},t)$: sensor radiance
$D(t)$: sensor demodulation function
$\otimes = B(\mathbf{p})$: measured brightness
$\Gamma(\mathbf{p})$

Figure 1: Illustration of a CW-ToF setup.

The brightness measured by the sensor can be expressed as shown in equation 2, where $\beta(p)$ is the scene albedo (light absorbed), L_a is the ambient illumination (assumed constant), and $\gamma_i = \int_0^\tau D(t)dt$. To determine the unknowns (β, Γ, L_a) we need at least 3 brightness measurements.

$$B_i(p,\beta,\Gamma,L_a) = \int_0^\tau D_i(t)L_i(p,t)dt = \beta(p)\int_0^\tau D_i(t)(M_i(t-\frac{2\Gamma}{c}))dt + L_a\gamma_i \quad (2)$$

$$1 \le i \le K, \quad K \ge 3 \quad (3)$$

2.2. Problem Definition

Given K brightness measurements of a given pixel p we want to recover its depth. If $M_i(t)$ and $D_i(t)$ $(1 \le i \le K)$ are simple functions such as sinusoids with the same frequencies, one can simplify equation 2 and obtain a system of K equations from which depth can be solved for. However, if $M_i(t)$ and $D_i(t)$ are more complex functions then there is no direct way to solve for depth from the set of K equations. The common procedure in this case is to construct a large lookup table that maps the set of K brightness measurements $(B_i(p,\beta,\Gamma,L_a))$ to a phase shift which is associated to depth. Searching such table for thousands of pixels is a computationally expensive procedure which renders many of the coding functions that achieve high depth resolution not practical for real world applications. We propose an alternate method to compute depth. We formulate the problem as a regression problem and leverage the neural network architectures described in the following section to learn how to compute depth from brightness measurements.

2.3. Algorithm

The first algorithm is predicting the depth of single pixel based on K brightness measurement. In this part we use fully connected network. The network will predict a single depth based on the brightness input of the pixel. The number of brightness measurement K could be any number greater than or equals to 3. A larger number of K will require more memory space and computing time, so we set $K = 3$ for the convenience. The network structure is shown in Figure 2. The input to the network is obtained from CW-ToF simulator and the output is the depth.

The second algorithm is predicting the depth of a whole image at the same time. In this case we are performing per pixel depth inference of a full image. Fully convolutional networks, originally introduced by Long et al. (2015), perform exactly this i.e output one or multiple values for each element of the input. The advantage of this algorithm is that when predicting the depth of one pixel, the network will take the input of its neighbors into account. This will reduce the impact of noise. In this case, the input to the network will be the brightness measurement of the whole scene and the output is the estimate depth of the whole scene. The input map is a three dimensional matrix with the three dimensionals to be the width, height and K. We implement FCN network to do this job since the output size is the same as the input size in FCN network. The architecture of the network is shown in Figure 3.

3. Experimental Setup

3.1. Evaluation Criteria

Since we are the measuring the depth of a scene, the label is the depth which we set to be between $0-10000$ millimeters. We want the loss value to reflect the real error of the prediction. This will make it easy to compare compare against analytical and ground truth results. Hence, we use the mean absolute error but not the mean squared error as the loss function.

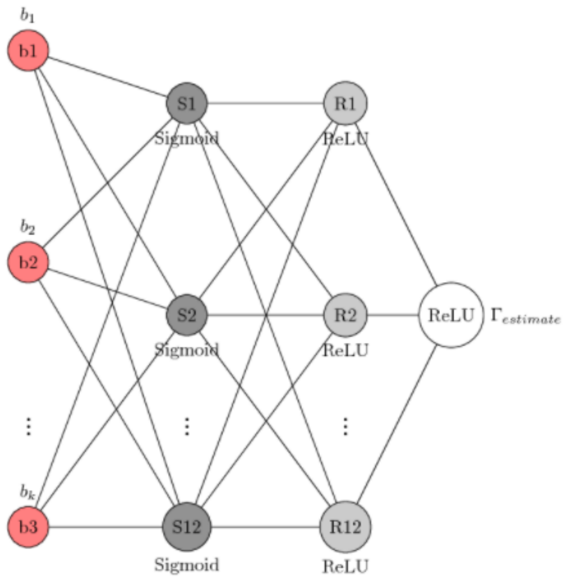


Figure 2: Fully connected network structure

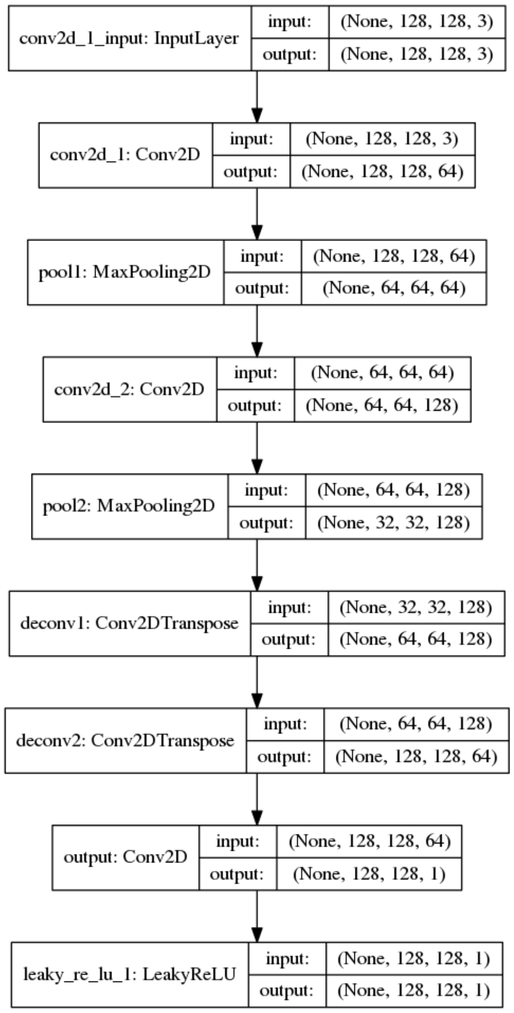

conv2d_1_input: InputLayer
input: (None, 128, 128, 3)
output: (None, 128, 128, 3)
conv2d_1: Conv2D
input: (None, 128, 128, 3)
output: (None, 128, 128, 64)
pool1: MaxPooling2D
input: (None, 128, 128, 64)
output: (None, 64, 64, 64)
conv2d_2: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 128)
pool2: MaxPooling2D
input: (None, 64, 64, 128)
output: (None, 32, 32, 128)
deconv1: Conv2DTranspose
input: (None, 32, 32, 128)
output: (None, 64, 64, 128)
deconv2: Conv2DTranspose
input: (None, 64, 64, 128)
output: (None, 128, 128, 64)
output: Conv2D
input: (None, 128, 128, 64)
output: (None, 128, 128, 1)
leaky_re_lu_1: LeakyReLU
input: (None, 128, 128, 1)
output: (None, 128, 128, 1)

Figure 3: Simple FCN network structure

3.2. Simulated Datasets

We developed a physically accurate CW-ToF simulator to generate the datasets used for training. The simulator takes as input: the scene's true depth map (2D matrix), the ambient light magnitude, the read noise variance of the ToF sensor, and the modulation/demodulation functions. Then for each point/pixel in the depth map it simulates K brightness measurements. Using this program we generated the following two datasets to train each network:

1. **100,000 Sample - Single Pixel Dataset:** This dataset is composed of brightness measurements for a single pixel (as opposed to full scenes). To generate this dataset we draw a depth from a uniform distribution between $0 - 10000$, draw the ambient illumination parameter from a uniform distribution between $1x - 5x$ (make ambient $1 - 5$ times stronger than the modulated light), draw the read noise parameter from a uniform distribution between $1 - 100$. Given those parameters we obtain the brightness measurements for that given point and repeated the process 100000 times.

2. **5,000 Sample - 5 Scene Dataset:** Each sample in this dataset corresponds to the set of brightness measurements obtained from each pixel in a $128x128$ scene. We used 5 basic

scenes:

(a) Wall: All pixels have same depth.
(b) 2 stairs: There is a total of 2 depths in the scene, half of the pixels are at one depth and the other half at the other.
(c) 4 stairs: Same as 2 stairs but with 4 depths.
(d) Ramp: An inclined wall.
(e) Half Sphere: A half sphere with a wall behind it.

A visualization of each scene is shown in 10. Similar to the Single Pixel Dataset to generate a sample in this dataset we draw the parameters from similar uniform distribution. Except the depth that is drawn from the distribution acts as an offset to the scene and now ranges between $1000 - 9000$ because the maximum width of any of these 5 scenes is 1000.

3.3. Environment Setup

We have used the Keras deep learning library which runs Tensorflow on the backend to test all of the network designd. All evaluation runs of the FCN reported in this report were performed on a single machine with NVIDIA 1070x GPU, while the single pixel network was run on a dual-core Intel i5 processor.

3.4. Hyperparameter Tuning for Single Pixel Network

In order to find a relatively good set of hyperparameters for single pixel network, several parameter space is explored:

- **Number of Hidden Layers:** one, two and three layers for single pixel are experimented.
- **Activation function:** In order to add nonlinearity to the network, activation of first hidden layer is fixed to be the sigmoid function. To make output scaled with a big range instead of 0 to 1, linear activation and leaky ReLu with different coefficient as activation functions for the later layer are explored.
- **Number of Hidden Units:** We experiment with several hidden units of different layers. These can be inferred from the first column of table 1.
- **Learning Rate:** 0.001, 0.01, 0.05, 0.1 and dynamic option are tested.
- **Batch Size:** 128, 64, 32 are tested in the experiment.
- **Weight Initialization method:** Random uniform initialization and random normal initialization are experimented.

3.5. Hyperparameter Tuning for FCN

Here the input is a 128x128x3 matrix (brightness image of the scene) and the output is a 2D depth map of the scene. While the same architecture was maintained, the hyperparameters that were tuned, were the following:

- **Patience:** $10, 20, 40$
- **Learning Rate:** $10^{-3}, 10^{-4}, 10^{-5}$

Here patience is the tune period. The tune period is the number of epochs of having tuneset error greater than or equal to the current least tuneset error. We also tried using average pooling instead of max pooling. We observed that there was more stability in the run, but only a marginal increase in accuracy.

4. Results

4.1. Result for single pixel network

The result is shown in Table 1. We use early stopping and a default maximum epochs of 10000. The last row gives us the best network parameters.

Table 1: Hyperparameter Search Result for single pixel network

Hidden Layer Structure	Activation function	Learning rate	Batch Size	Weight Initialization	Test Set Loss
[12, 12]	Leaky ReLu, α = 0.3	0.01, no dynamic rate.	128	Random uniform	95.42430879
[12, 12]	Leaky ReLu, α=0.1	0.01, no dynamic rate.	128	Random uniform	100.3975776
[12,12]	Leaky ReLu,α = 0.3	0.05, no dynamic rate.	128	Random uniform	196.6624763
[12, 12]	Leaky ReLu,α = 0.3	0.1, dynamic rate.	128	Random uniform	82.31620815
[12, 12]	Linear	0.01, no dynamic rate.	128	Random uniform	119.7730264
[12, 12]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random Normal	139.6405088
[12, 12]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	64	Random uniform	131.4798378
[12, 12]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	32	Random uniform	74.56125828
[10]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	1242.294179
[50]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	104.1866253
[200]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	99.54450011
[100,50]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	83.10440374
[400,20]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	82.40132151
[40,20,10]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	112.9198268
[200,100,50]	Leaky ReLu,α = 0.3	0.01, no dynamic rate.	128	Random uniform	107.2272492
[12,12]	Leaky ReLu,α = 0.3	0.001, no dynamic rate.	32	Random uniform	**68.45357662**

Figure 4 illustrates the early stopping curve using parameters which gives us the best test set loss. It is shown in the figure that the tuning set accuracy would not dropping after two hundred epochs. Setting ‘patience’ of early stopping helps us reduce the risk of over-fitting and saves us time.

Figure 5 gives the learning curve using parameters which gives us the best test set loss. It achieves almost the same test set loss when using 75% of data, which indicates the data in our experiment is of reasonable size.

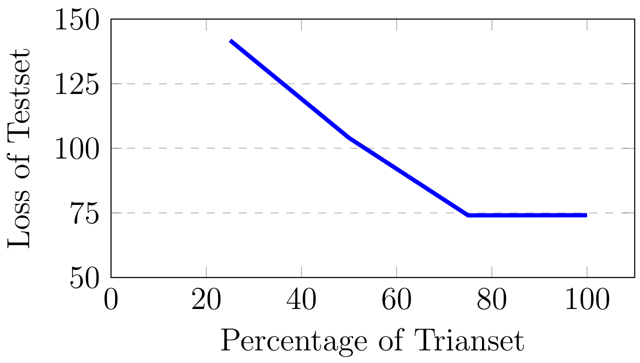


Figure 4: Learing curve of single pixel network

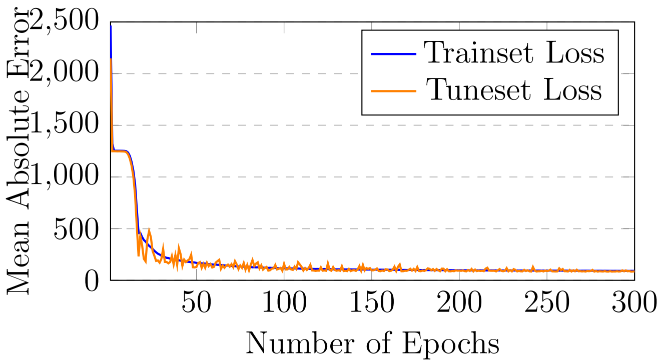


Figure 5: Loss of train&test set in training process

4.2. Result for single FCN network

. **Testing Loss.** Table 2 illustrates the testset errors at the different values of the hyperparameters - patience and learning rate. Testset error decreases with increase in patience and reduction in the learning rate.

. **Convergence/Tuning Training Loss.** Figures 6,7 and 9 illustrate the convergence of the trainset and tuneset losses as a function of the number of epochs that the network runs for. The number of epochs is determined by the learning rate and the patience of the model.

. **Learning Curve.** Figure 8 demonstrates that we achieve better accuracy as we increase the size of the dataset.

Table 2: Hyperparameter Search Result for FCN network

Patience \ Learning Rate	10^{-3}	10^{-4}	10^{-5}
10	995.744321289	197.823323975	348.555881348
20	183.465506592	223.862288818	186.36887085
40	151.612488403	143.844888916	**77.2267715454**

4.3. Depth Map Visualizations

Figure 10 contains some of the recovered depth maps using different methods. The first column correspond to the ground truth scenes. The second column corresponds to the results using the computationally expensive lookup table. The third column were obtained by iterating through each pixel in the depth map obtaining the brightness measurements and giving them as input to the single pixel network. The fourth column corresponds to the depth map calculated by the FCN.

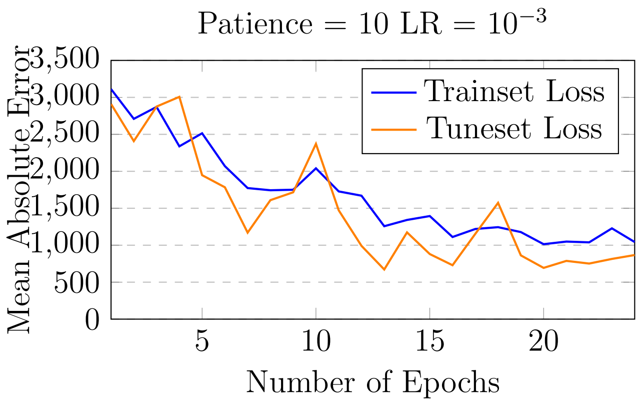


Figure 6: Loss of train&test set in training process

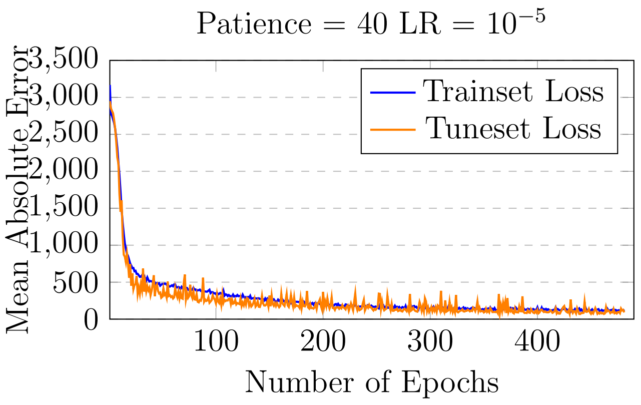


Figure 7: Loss of train&test set in training process

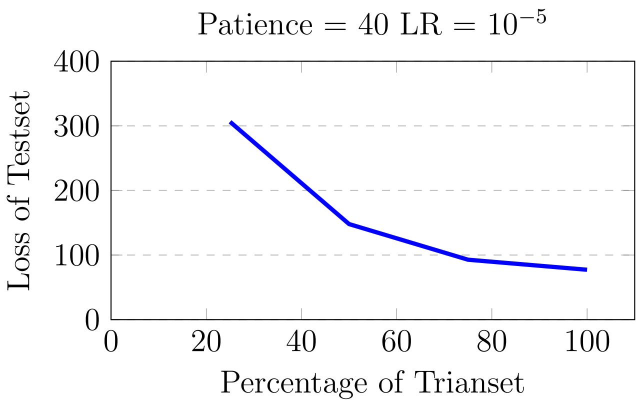


Figure 8: Learning curve of FCN network

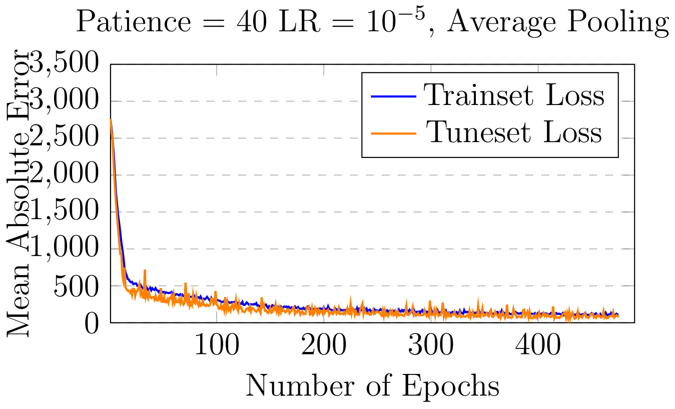


Figure 9: Loss of train&test set in training process

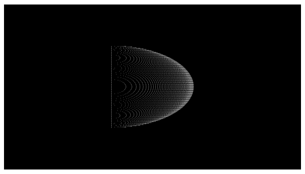
5. Discussion

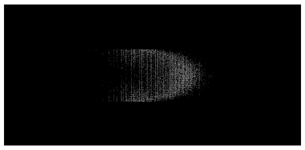
The single pixel network is able to generate comparable depth estimates to the analytical depth estimates. Table 1 shows that increasing the number of hidden units and layers did not have a significant impact on the test set error. A linear output activation function is typically used in neural network-based regression, however, we find that a leaky ReLU was a better choice for the output. The best results are obtained using a small batch size, 2 layers (12 HUs each), with a leaky ReLU. Finally, the learning curve in 4 shows that the impact of more training samples would be minimal.

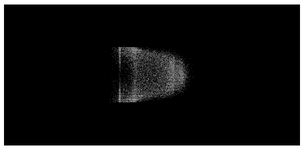
The FCN is able to outperform the analytical mean absolute error (MAE=170 mm) in the testing set. Table 2 shows that by increasing the patience of the model benefits the model. This could be attributed to the fact that inject noise at the time of the creation of the simulated dataset. Since the noise is random and in addition to this we shuffle the dataset before each epoch, the tuneset loss does not steadily decrease. But if we have a sufficient enough patience, we get results that are better than the analytical or the single pixel network. Overall, we can observe from 2 and figure 6, 7, and 9 could be improved even further from training from a larger number of epochs without the risk of overfitting.

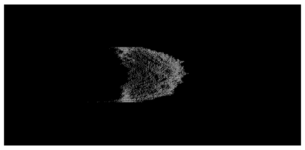
We also observe that as we reduce the learning rate, we get better accuracy. Empirically, it is observed that the 'ADAM' optimizer that we have used in our model, gives the best results with learning rates of the order of 10^{-4} to 10^{-5}. This is evident from the results.

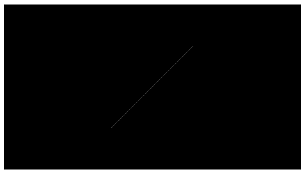
Depth Maps

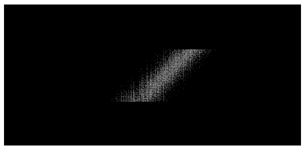

(a) Truth, **MAE=0**

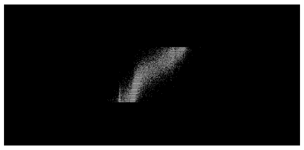

(b) Analytical, **MAE=267.70**

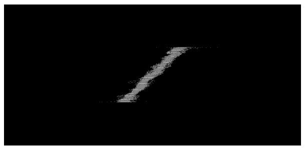

(c) Single Px Network, **MAE=129.74**

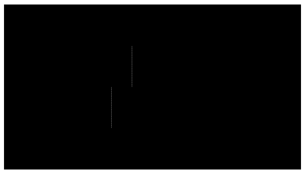

(d) FCN, **MAE=1182.07**

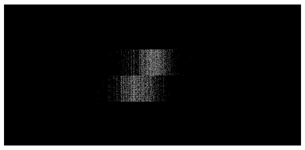

(e) Truth, **MAE=0**

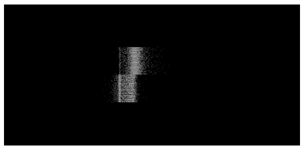

(f) Analytical, **MAE=197.93**

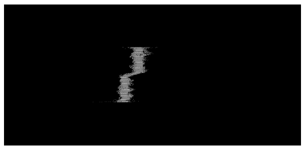

(g) Single Px Network, **MAE=130.44**

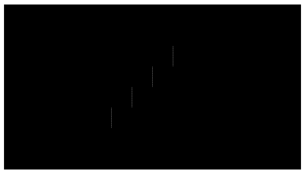

(h) FCN, **MAE=44.19**

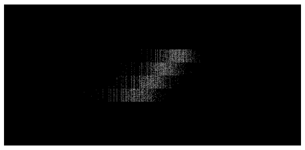

(i) Truth, **MAE=0**

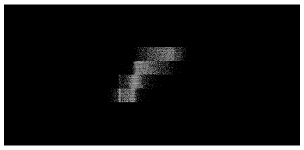

(j) Analytical, **MAE=134.33**

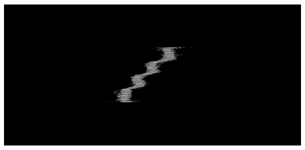

(k) Single Px Network, **MAE=99.13**

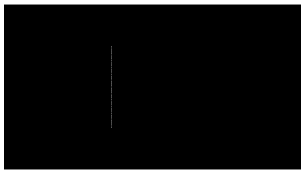

(l) FCN, **MAE=96.71**

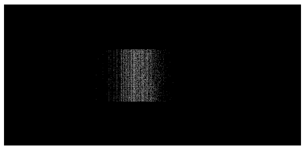

(m) Truth, **MAE=0**

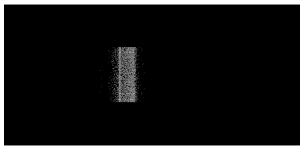

(n) Analytical, **MAE=58.34**


(o) Single Px Network, **MAE=131.76**


(p) FCN, **MAE=639.60**


(q) Truth, **MAE=0**


(r) Analytical, **MAE=172.48**


(s) Single Px Network, **MAE=98.68**

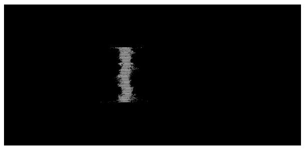

(t) FCN, **MAE=48.27**

Figure 10: Visualization for the depth maps of all 5 scene types. Row 1 is a half sphere, row 2 is a ramp, row 3 are two walls at different depths, row 4 are 4 walls at different depths, and row 5 is a wall at a single depth. The noise parameters added to the brightness measurements were the average noise added in the dataset.

As opposed to the single pixel network, the learning curve 8 shows that the FCN model would benefit from a larger dataset. As we discuss in future work increasing the dataset size and the diversity of scenes within the dataset is one of the immediate next steps.

One particular point that is not illustrated in the results is the fact that the FCN is able to handle the phase wrapping issue encountered in the analytical and single pixel networks models at small and large depths (near 0 and 10000). We believe that this can be attributed to the convolution steps in the FCN which take into account a local neighborhood information. This helps mitigate the effect of the outliers (brightness measurements with extremely large magnitudes due to noise). The analytical result and the single pixel network on the other hand do not consider the neighborhood and hence their performance is worse in these scenarios.

6. Individual Contributions

- **Felipe:** Wrote ToF simulator. Generated datasets. Generated analytical results. Wrote FCN code.
- **Zhicheng:** Build and run experiment on the fully connected network for single pixel.
- **Zhenyu:** Run the fully connected network experiments and processing result data.
- **Nikhil:** Run the experiments for the FCN network. Set up the GPU system.

7. Future Work

We have demonstrated that neural networks can effectively map ToF brightness measurements to depth. Knowing this there are multiple ways to extent this idea, in particular for the FCN model.

- **More data:** The learning curve for the FCN model shows that a larger dataset can benefit the performance of the current model. Furthermore, we are only using 5 sample scenes under many noise configurations to generate the dataset. Therefore, in the future it would be interesting in creating a dataset with thousands of scenes combined with the multiple noise configurations and see if the model becomes more robust.
- **Introducing new scenes:** We performed a preliminary study of what would happen if we train the model on 4 scenes and then gave a new scene as input. We are still interpretting these results and due to lack of space have not included them in the report.
- **Deeper and more complex network:** Pathak et al. (2015) and Hazirbas et al. (2017) have succesfully leveraged FCN type networks as regression models to estimate albedos from images and depth from focus, respectively. Their state of the art results are obtained from much deeper neural networks. Furthermore, Hazirbas et al. (2017), whose network also outputs depth (from different measurements), concatenates some of the initial convolution layer activations in the future deconvolution layers. This seems as an essential step of their model which we believe could also benefit ours.

8. Conclusion

In this report, we proposed two neural network to solve the problem of estimating the depth based on ToF brightness measurements. The fully connected neural network takes the brightness measurement of single pixel as input and predict the depth of the pixel. The input of FCN network is the brightness measurement of all pixels in the image and the output is the depth of all pixels. We used data generated from CW-ToF simulator to test our networks, and the results show that our networks work better than analytics results in some cases.

References

Adam, A., Dann, C., Yair, O., Mazor, S., Nowozin, S., 2017. Bayesian time-of-flight for realtime shape, illumination and albedo. IEEE transactions on pattern analysis and machine intelligence 39 (5), 851–864.

Hazirbas, C., Leal-Taixé, L., Cremers, D., 2017. Deep depth from focus. arXiv preprint arXiv:1704.01085.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440.

Pathak, D., Krähenbühl, P., Yu, S. X., Darrell, T., 2015. Constrained structured regression with convolutional neural networks. arXiv preprint arXiv:1511.07497.