TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING

הפקולטה להנדסה על שם איבי ואלדר פליישמן

Depth refinement using content aware neural network filters

A Project

submitted in partial fulfilment

of the requirements for the Degree

of

Master of Science

by

Shai Weisman

I.D 300106358

Tel Aviv University

June 2020

Contents

Figures

Abstract

Depth estimation is an important yet challenging problem in computer vision. Since both stereo and monocular methods tend to fail in texture less, repetitive and\or caged regions, the final depth estimation generally suffers from artifacts that affect the quality of the result.

The goal of the current project is to study the use of neural networks for depth maps improvement using content aware neural network filters. Two main approaches are tested: Joint-filter network (JF-Net), a CNN which selectively transfer salient structures that are consistent with both guidance and target images, and Fast Deformable Kernel Network, a DNN which defines a set of neighbors and the corresponding weights (based on both guidance and target images) adaptively for each pixel . Both solutions are testes with several architecture variants and different loss functions.

After analysing all the different solutions, we achieved a reduction of 20% in the L1 error using the FDKN adaptive loss network on the validation set. However, improvement on the real-world images was smaller.

This work presents the potential of content aware filters in error reduction, and hopefully with additional tools will help to bridge the domain gap between synthetic and real-world datasets.

This project is based on the research of Mr. Shay Elmalem under the supervision of Dr. Raja Giryes.

I. Introduction

Depth map estimation is a core task in many computer vision applications. It can be used for recovery of 3D shapes and scene understanding. Obtaining a precise and accurate depth map is a key step for the success of those applications.

With that stated, many of the current algorithems which try to predict depth , either using stereo approaches or using a single image [1], suffer from various issues in predicting depth accurately.

the root cause of the error could be from many sources such as opacity of objects, occlusion, repetive texrure and more.

in order to resolve this issue , many depth algorithem use joint image filters for denoising. The basic concept behind joint image filtering is to transfer structural details from the guidance image to the target one, typically by estimating spatially-variant kernels from the guidance. By that a weighted averaging kernel is defined, which allows edge preserving denoising when applied on the depth image.

Classical approaches to joint image filtering mainly focus on designing the filter kernels **W** and the set of neighbors **N** (i.e., sampling locations q). They use hand-crafted kernels and sets of neighbors without learning. For example, the bilateral filter [1] uses spatially-variant Gaussian kernels to encode local structures from the guidance image.

This method is efficent and suffient in may applications, however, difficult to manually adapt the kernels to new tasks, and these methods may transfer erroneous structures to the target image and by that creating the opposite effect and “noisng” the image.

In the recent years, with the introduction of neural network to the set of applicable tools to solve imaging problems, several researchers tackled the image denoising problem.
Untill recently, the SOTA solution to this problem was the deep joint filter net [13], in which a CNN-based methods an encoder-decoder architecture is used to learn features from the target and guidance images, and the filtering output is then regressed directly from the network. This Learning-based techniques require a large number of ground-truth images for training and still uses a uniformic grid for feature extraction.

In 2020, the Deformabel kernel network (DKN) was introduced as the current state of the art, it diffrentiated from previous approaches by applying a different sampling grid per pixel and allowing a better weighted average per pixel.

in addition, in CVPR 2019, a new adaptive loss was introduced [15] to solve a rising question in neural network training, “how do you decice on the best loss fuction for your network?”.
the solution that was proposed is an adaptive fuction that base on the distribution of the error in each epoch the fuction changes from L2(noisy) to L1 (less noisy), by that we allow the loss to be based on the data of the network and not prior asumptions.

In this work, we tested both the fast implementation of the DKN (FDKN) and 2 implementations of the JF-Net, one as designed in the original paper, and one with an activation function of ELU, in order to test if changing the activation function will allow reaching better results, on the

MonSter depth output. In addition, we tested all networks both with L1 loss and adaptive loss functions.

The result showed a 18% improvement with FDKN using L1 loss and 20% with adaptive loss, on the validation set based on synthetic outputs of the network. When observing the results of real-world depth estimation, the output of the network showed limited added value compared to the input.

Background

Monocular Depth Estimation Using Aperture Phase-Coding

In this work we will enhance Monocular Depth Estimation Using Aperture Phase-Coding as presented in “MonSter: Awakening the Mono in Stereo” [1]

Passive depth estimation is among the most long-studied fields in computer vision. The most common methods for passive depth estimation are either a stereo or a monocular system. Using the former requires an accurate calibration process and has a limited effective range. The latter, which does not require extrinsic calibration but generally achieves inferior depth accuracy, can be tuned to achieve better results in part of the depth range.

Monocular depth estimation aims at finding depth cues, which can be either global (such as perspective and shadows) or local (focus/out-of-focus).

One method for monocular depth estimation based on a learned phase coded mask, presented in [Haim et al., 2], which embeds depth related cues in an acquired optical image. These cues are in turn extracted by a Convolutional Neural Network (CNN), trained to estimate the scene depth according to those cues and enable a more accurate monocular depth estimation. The accuracy of phase-encoded monocular depth estimation depends on the ability to detect a change in focus for different depth ranges, thus, it is most effective around its focus point. Knowing the focus point of the camera, one can obtain an absolute depth map, with real-world distance units.

aperture phase-coding has the advantage of having a very-high light efficiency, with almost no loss. In particular, we use the technique proposed by Haim et al. [2] They suggest using an aperture phase-coded mask in the image acquisition process. Using the phase-coded mask, a depth and color dependent point spread function (PSF) is generated, such that each of the image's RGB channels can be thought of as being optimally focused on a different depth in the scene.

Using the focus and out-of-focus color-depth cues embedded in the image, a neural network can be used to predict the defocus condition (labeled as ψ) at each pixel. Assuming the lens parameters and focus point(s) are known, the absolute depth can be easily derived from ψ, which is given by:

$$(1)\ \psi = \frac{\Pi R^2}{\lambda}\left(\frac{1}{z_0} + \frac{1}{z_{img}} - \frac{1}{f}\right) = \frac{\Pi R^2}{\lambda}\left(\frac{1}{z_{img}} - \frac{1}{z_i}\right) = \frac{\Pi R^2}{\lambda}\left(\frac{1}{z_{img}} - \frac{1}{z_n}\right)$$

where R is the radius of the exit pupil (assuming a circular aperture), λ is the illumination wavelength, zimg is the sensor plane location for an object in the nominal position zn, and zi is the ideal image plane location for an object located at zo. As |ψ| increases, the image contrast decreases, hence the contrast is at maximum for ψ=0 (the in-focus position). The mask and method presented in [Haim et al., 2] are designed for optimal depth estimation in the range of ψ=−4..10.

Bilateral filter

Bilateral filter [3] was firstly presented by Tomasi and Manduchi in 1998. The bilateral filter takes a weighted sum of the pixels in a local neighborhood; the weights depend on both the spatial distance and the intensity distance. In this way, edges are preserved well while noise is averaged out.

Mathematically, at a pixel location x, the output of a bilateral filter is calculated as follows:

$$(2)\quad \tilde{I}(x) = \frac{1}{C} \sum_{y \in N(x)} I(y) e^{\frac{-\|y-x\|^2}{2\sigma_d^2}} e^{\frac{-\|I(x)-I(y)\|^2}{2\sigma_r^2}}$$

where σ_d and σ_r are parameters controlling the fall-off of weights in spatial and intensity domains, respectively, N x( )is a spatial neighborhood of pixel I x( ), and C is the normalization constant:

$$(3)\ \ C = \sum_{y \in N(x)} e^{\frac{-\|y-x\|^2}{2\sigma_d^2}} e^{\frac{-\|I(x)-I(y)\|^2}{2\sigma_r^2}}$$

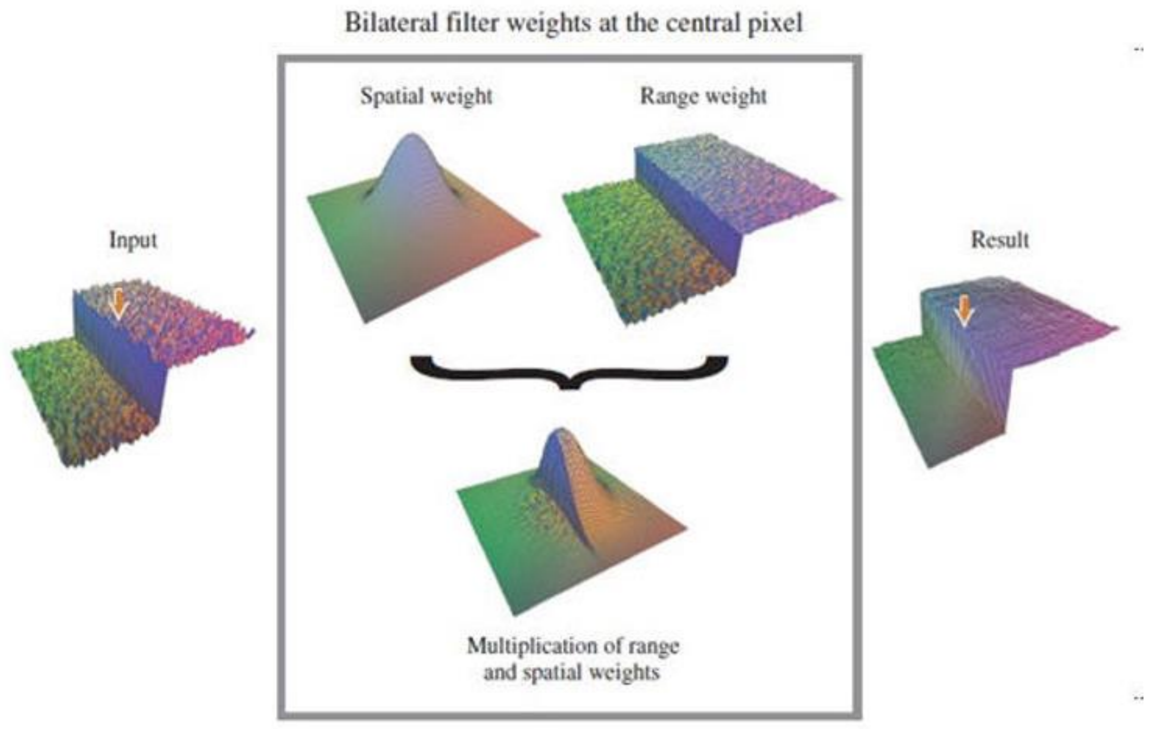

Bilateral filter weights at the central pixel
Spatial weight
Range weight
Input
Result
Multiplication of range and spatial weights

Figure 1 Bilateral filter weights at the central pixel

Figure 1 shows the illustration of the process of a bilateral filter. on the left we can see the input, a noisy step function, in the center we can note the spatial and range kernel and their multiplication, and on the right the output image which can be seen as edge preserving with noise reduction properties.

One improvement to the bilateral filter is the joint bilateral filter in which the range kernel is calculated on a guide image and not the target, by that allowing edge and texture preserving

$$(4) \quad \tilde{I}(x) = \frac{1}{\sum_{y \in N(x)} e^{\frac{-\|y-x\|^2}{2\sigma_d^2}} e^{\frac{-\|I_G(x)-I_G(y)\|^2}{2\sigma_r^2}}} \sum_{y \in N(x)} I(y) e^{\frac{-\|y-x\|^2}{2\sigma_d^2}} e^{\frac{-\|I_G(x)-I_G(y)\|^2}{2\sigma_r^2}}$$

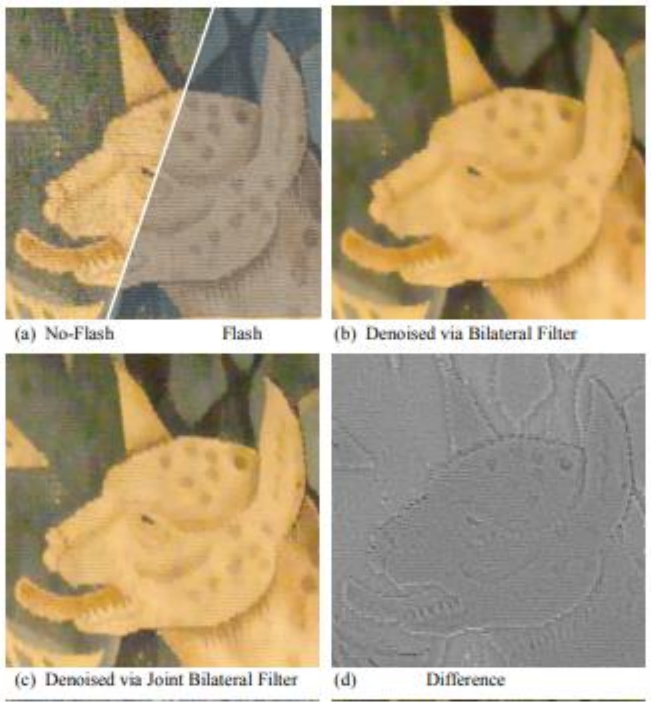

(a) No-Flash
Flash
(b) Denoised via Bilateral Filter
(c) Denoised via Joint Bilateral Filter
(d) Difference

Figure 2 Joint bilateral Filter example
(a) A close-up of a flash/no-flash image pair of a Belgian tapestry. The no-flash image is especially noisy in the darker regions and does not show the threads as well as the flash image. (b) Basic bilateral filtering preserves strong edges, but blurs away most of the threads. (c) Joint bilateral filtering smoothes the noise while also retaining more thread detail than the basic bilateral filter. (d) The difference image between the basic and joint bilateral filtered images

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 and require fine parameter configuration including the size of the kernel and the size of the sigma, which can cause over\under fitting and need to be adjusted per task.

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 [3,4] and guided filters [5] 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. There are several approaches to 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, content aware NN learn 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 [6]. In the context of joint depth up sampling, Tai et al. [7] use a multi-scale guidance strategy to improve up sampling performance. Gu et al. [8] adjust the original guidance dynamically to account for the iterative updates of the filtering results. However, these methods [7,8] are limited to the application of depth map up sampling. In contrast, in recent years generic joint filters are used for various applications using target/guidance image pairs in different visual domains by deconstructing the target image and

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[9] 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.

RELU & ELU

Rectified Linear Unit or **ReLU**,[10] is the default choice for most deep learning projects today, And can be represented as follows:

$$f\,(x) = \begin{cases} x & if\ x > 0 \\ 0 & else \end{cases}$$

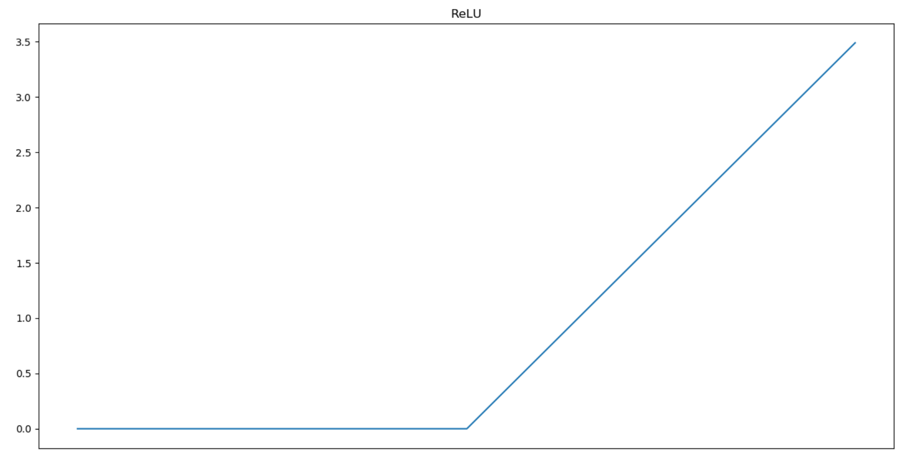


Figure 3 Relu Activation function

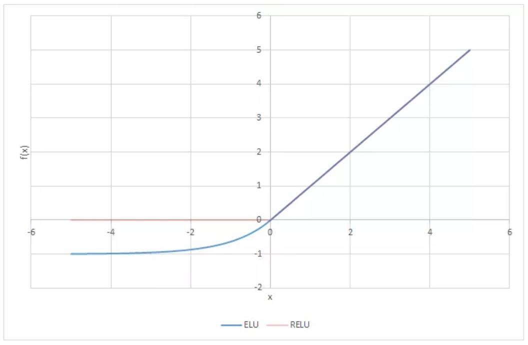
Hence, it produces a zero output for all inputs smaller than zero; and x for all other inputs.

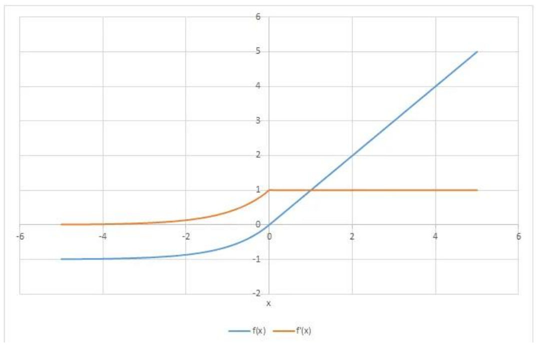
the Relu has several advantages:

- Sparsity- his benefits sparsity substantially: in almost half the cases, now, the neuron doesn't fire anymore. This way, neurons can be made silent if they are not too important anymore in terms of their contribution to the model's predictive power.
- Fewer vanishing gradients- It also reduces the impact of vanishing gradients, because the gradient is always a constant: the derivative of f(x)=0 is 0 while the derivative of f(x)=x is 1. Models hence learn faster and more evenly.
- Computational requirements- ReLU does need much fewer computational resources than the Sigmoid and Tanh functions (Jaideep, n.d.). The function that essentially needs to be executed to arrive at ReLU is a max function: max(0,x) produces 0 when x<0 and x when x>=0.

With that said, the Relu comes with several challenges, in which the most important one is the *dying ReLU problem* (Jaideep, n.d.). If a neuron's weights are moved towards the zero output, it may be the case that they eventually will no longer be capable of recovering from this. They will then continually output zeros. This is especially the case when your network is poorly initialized, or when your data is poorly normalized, because the first rounds of optimization will produce large weight swings. When too many neurons output zero, you end up with a dead neural network.

To overcome this issue, several activation function were proposed, one of the is the ELU(exponential linear unit) [11].





$$ELU\ (x) = \begin{cases} x & if\ x > 0 \\ \alpha(e^x - 1) & if\ x < 0 \end{cases} \qquad ELU'(x) = \begin{cases} 1 & if\ x > 0 \\ ELU(x) + \alpha & if\ x < 0 \end{cases}$$

Figure 4 ELU Activation function

The y-value you get depends both on your x-value input, but also on a parameter alpha

A (default 0.1-0.3), which you can adjust as needed. Furthermore, it uses an exponential operation, which means the ELU is more computationally expensive than the ReLU.

We can note, that for x>0 the ELU is equivalent to the Relu, but for values of x<0 we get a value slightly below zero, this allows to avoid the dying relu problem.

Transfer Learning

A technique that is vastly used on recent researches. It allows using a network that was designed and trained to solve one problem and fine-tune it so it can be fit to solve a different problem (the actual problem that the user needs to solve).

It was experimentally showed on many occasions that such a technique can significantly improve results and save training time [12], and by that may resolve inherent issues that most of the users have:

- Training a network is a Sisyphean and time-consuming task and usually requires a lot of processing power.
- To properly train a network, one must collect a very large number of examples (can reach millions for a modern deep network). This is something that most of the users cannot afford (or don’t have enough time to manually collect it).

However, recent studies claim that in some cases transfer learning can only save training time, but best performance is only achieved with ‘from-scratch’ training. It seems that this issue is still an open question, as can be seen both in the literature and as a conclusion from this work.

II. Proposed Solution

In this work, a new scheme for enhancing the depth map from a mono image is proposed. in this scheme we will inset the mono depth map and the RGB image (as guide) to a neural network which will perform content aware noise filtering and smoothing of the depth image.

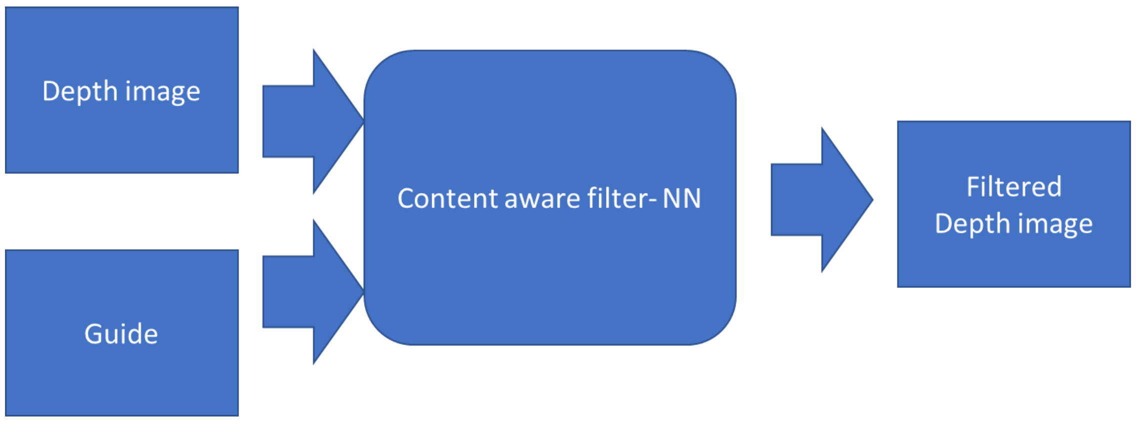


Figure 5 content aware filter general flow

To improve the final depth image, this work will examine 2 different NN for the content aware filtering:

- **'Joint Image Filtering with Deep Convolutional Networks' (JF-Net)[13]-** a network which uses both images based on feature maps and enforces consistency implicitly through learning from examples. Considered as SOTA untill June 19'.
- **Fast Deformable Kernel Network (FDKN)[14]**- Inspired by classical approches, the network outputs for each pixel in the target image sets of sampling points $N(p)$ and

their corresponding weights W, which are learned from combined learned features extracted from the guide and the target images.

Compare between the use of ELU vs RELU in the JF -NET and test if it gives any added value. in addition, we will compare (in all methods) between using L1 loss (which is the common loss in depth NN) and adaptive robust loss which enables the training of neural networks in which the robustness of the loss automatically adapts itself during training (changing between Charbonnier/pseudo-Huber/L1-L2, and L2 loss functions).

In addition, this project will examine the added value of re-training only the final 2 layers of the FDKN vs all the network.

Model	**Loss**	**Weights trained**
JF-Net (Relu)	L1	all
JF-Net (Relu)	Adaptive loss	all
JF-Net (ELU)	L1	all
JF-Net (ELU)	Adaptive loss	all
FDKN	L1	all
FDKN	Adaptive loss	all
FDKN	L1	Only final 2 layers
FDKN	Adaptive loss	Only final 2 layers

Figure 6 table of models ,losses and layers trained
from left to right : the DNN model (JF-Net and the main activation function of FDKN),the loss function used for training and on which layers the weights were trained

Related work

Joint Image filter CNN

JointFilterNet consists of three sub-networks: the target network $\mathrm{CNN_T}$, the guidance network $\mathrm{CNN_G}$, and the filter network $\mathrm{CNN_F}$ as shown in Figure 3. First, the sub-network $\mathrm{CNN_T}$ takes the target image as input and extracts a feature map. Second, like $\mathrm{CNN_T}$, the sub-network $\mathrm{CNN_G}$ extracts a feature map from the guidance image. Third, the sub network $\mathrm{CNN_F}$ takes the concatenated feature responses from the subnetworks $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ 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 $\mathrm{CNN_T}$ and $\mathrm{CNN_G}$ are to serve as nonlinear feature extractors that capture the local structural details in the respective target and guidance images. The sub-network $\mathrm{CNN_F}$ 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 are simultaneously considered when predicting the final filtered result. Such a design allows to selectively transfer structures and avoid texture-copying artifacts.

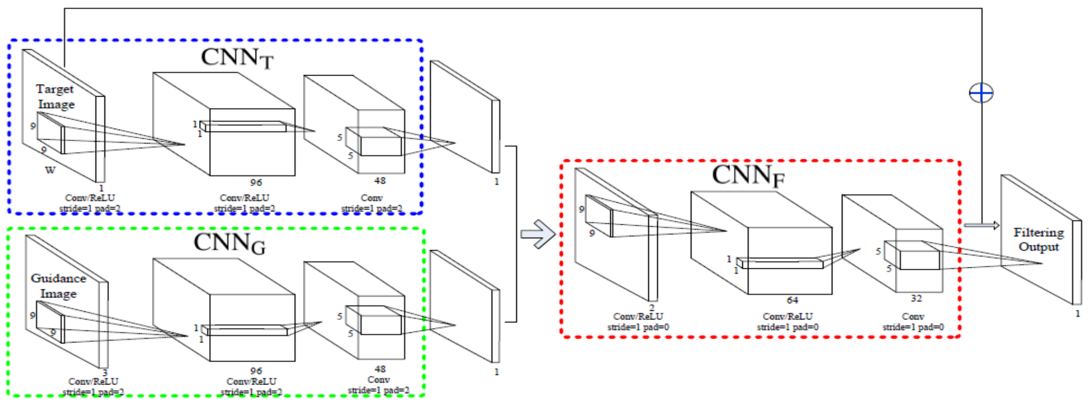


Figure 7 Joint Filter Net Model

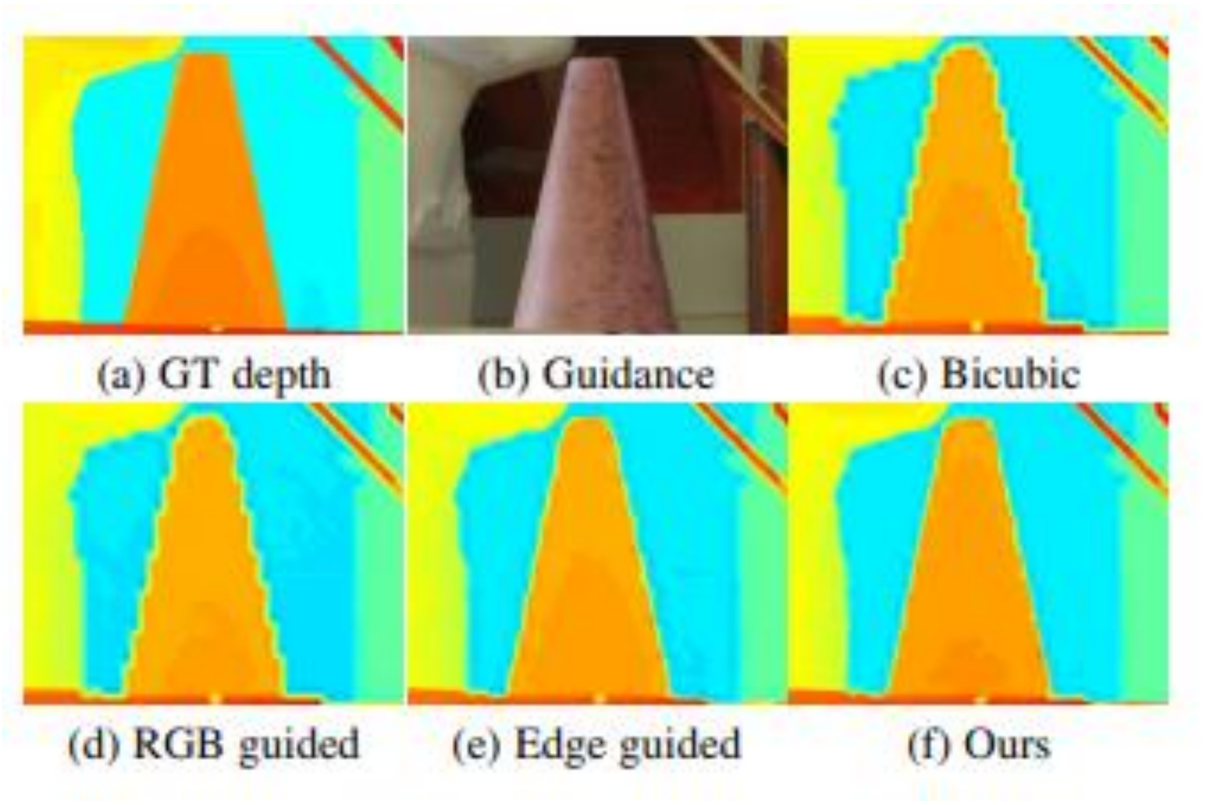


Figure 8 comparison between classic joint filtering methods
(a) Depth Ground truth ,(b) Guidance image,(c) bicubic interpolation,(d) RGB Guide filter,(e) edge guided filter and(f)

Fast Deformable Kernel Network (FDKN)

FDKN structure, consists of two parts: a learning part and a weighted average process to obtain high-resolution outputs.

The learning part is done by a two-stream sub-networks with the same structure for the guidance and the target images. It starts with feature extraction maps from each image and continues to a weights and offsets regression by combining the feature maps from each stream with convolution layers and sigmoid activation function for the weight regression.

The weighted average process is done using the map of weights with size of k^2 and a map of offsets (x, y) with size of $2k^2$ for each pixel in the target image. **Error! Reference source not found.** illustrates the weighting average process with a demonstration of the deformable sampling locations denoted by $s(q)$.

The architecture suggests using a residual connection between the target image and the weighted average, since the filtering result is highly correlated with the target image:

$$(4)\ \hat{f}_p = f_p + \sum_{q \in N(p)} K_{pq}(f, g) f_{s(q)}$$

where $f_p, \hat{f}_p$ are the target and filtered images, respectively. K_{pq} is the spatially variant kernel for the pixel p, $f_{s(q)}$ is the sampled image value at offset Δq from the location q.

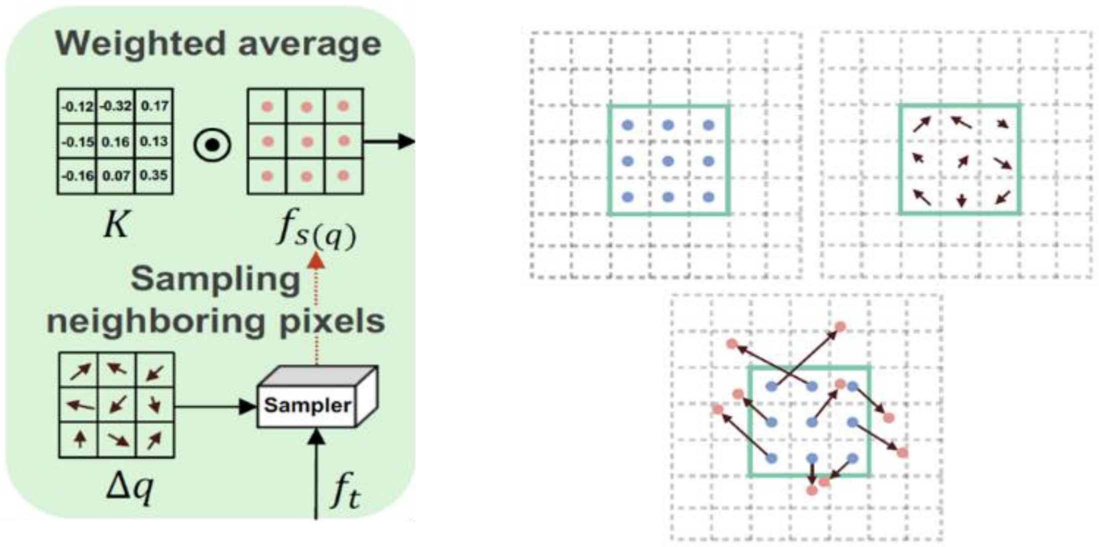


Figure 9 - Weighted average process. On the left: learned kernel K weights multiplied by sampled target image deformable learned locations. On the right: First row: regular sampling q on discrete grid and learned offsets. Second row: deformable sampling

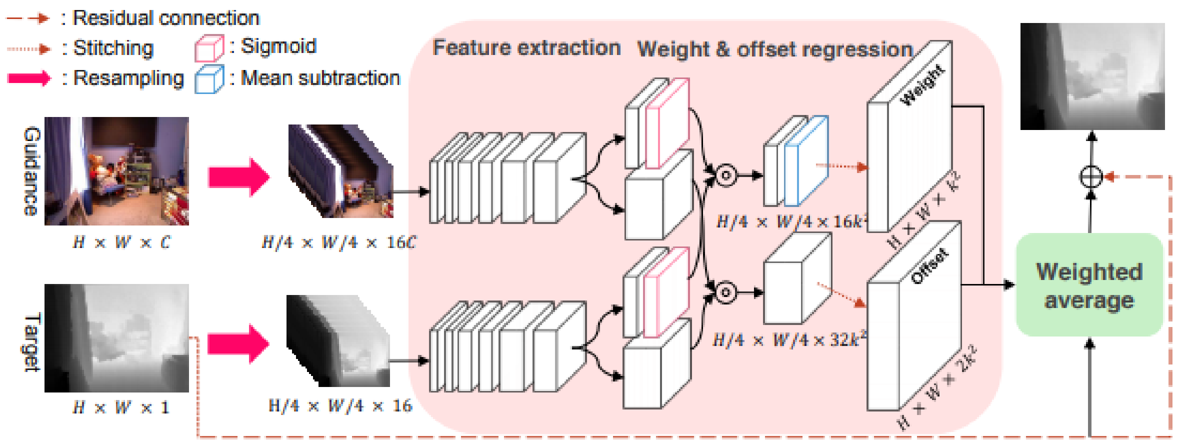


Figure 10 FDKN model architecture

To speed up the network, both the Guidance and the target are subsampled and shifted parts as before, but this time stack them into new target and guidance images (Fig. 5), with 16 channels for the former, and 16C channels for the latter, e.g., C = 3 when the RGB image is used.

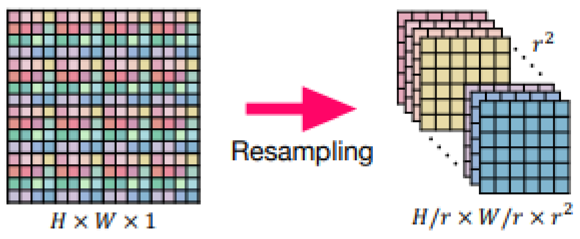


Figure 11 FDKN resampling method

The FDKN is considered the fast and equivalent version of the SOTA content aware filter

Neural network

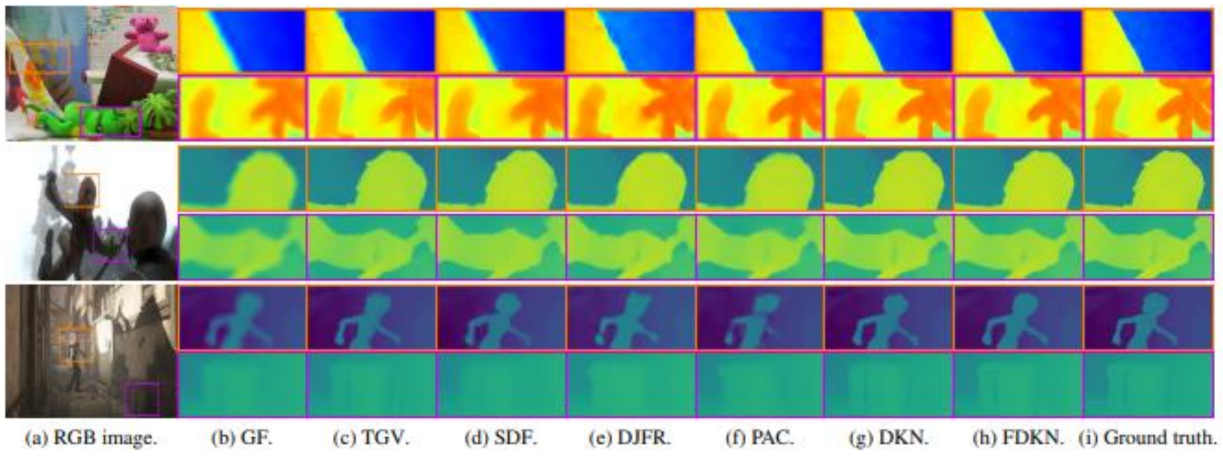


Figure 12 Comparison between different guide filters and FDKN

Adaptive robust loss function

One of the most common challenges in tuning a neural network is defining the loss function.

manual testing and assessing a loss function is tiresome and data specific.

therefore the Adaptive robust loss function[15] suggests the following :

$$f(x,\alpha) = \frac{|\alpha-2|}{\alpha}\left(\left(\frac{x^2}{|\alpha-2|}\right)^{\frac{\alpha}{2}} - 1\right)$$

Several values of α reproduce existing loss functions: L2 loss (α = 2), Charbonnier loss (α = 1),

Cauchy loss (α = 0), Geman-McClure loss (α = −2), and Welsch loss (α = −∞)

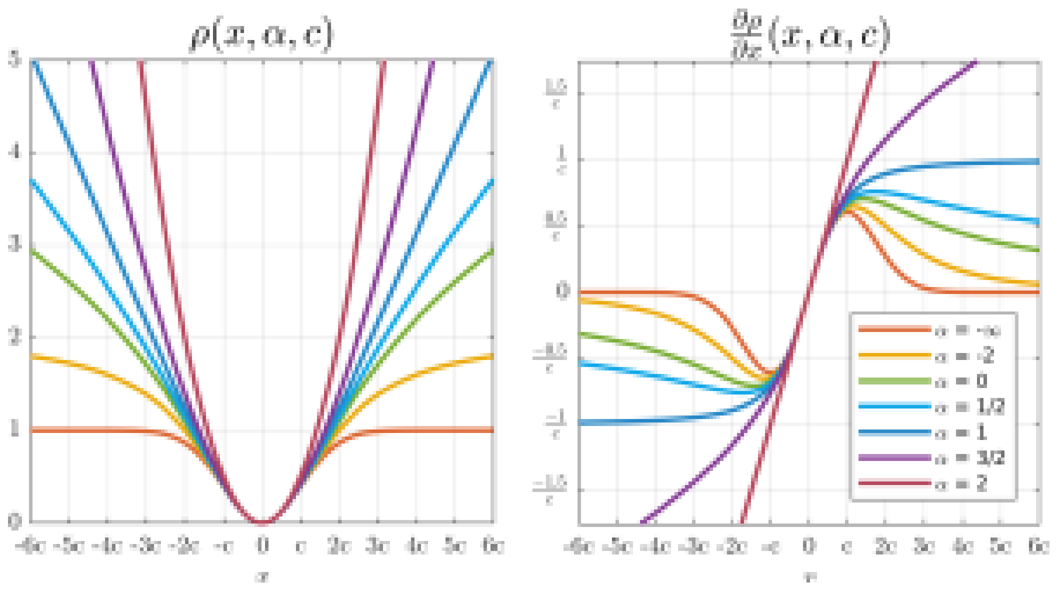


Figure 13 general loss function (left) and its gradient (right)

To allow alpha to adapt per changes in the data during training , the paper suggests minimize the negative log of the likelihood of the probability function:

$$-\log p(x|\alpha) = p(x|\alpha) + logZ(\alpha)$$

and by that not allowing the adaptive to cheat, cause if it wants to give discount to outlier it needs to pay an extra penalty on inliers (and vice versa)

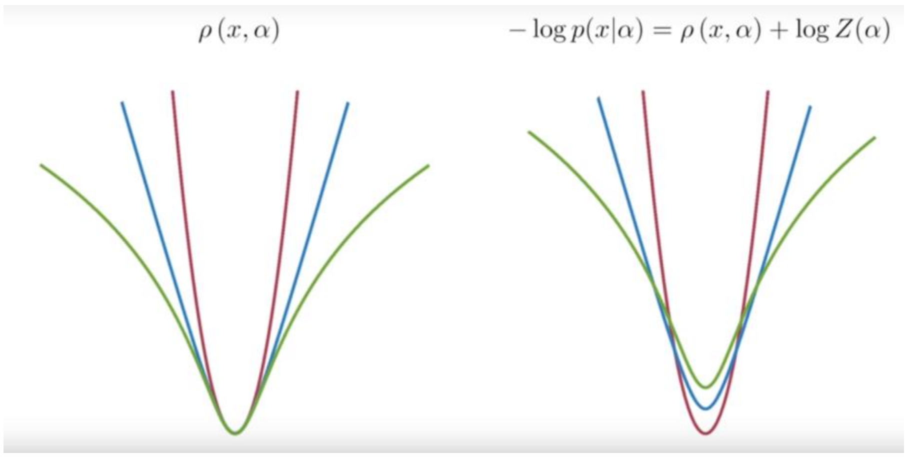


Figure 14 Adaptive loss function

This allows the loss to define the α (and the loss function) per progress in the training.

as can be seen in the figure below, it shows promising results on mono depth results.

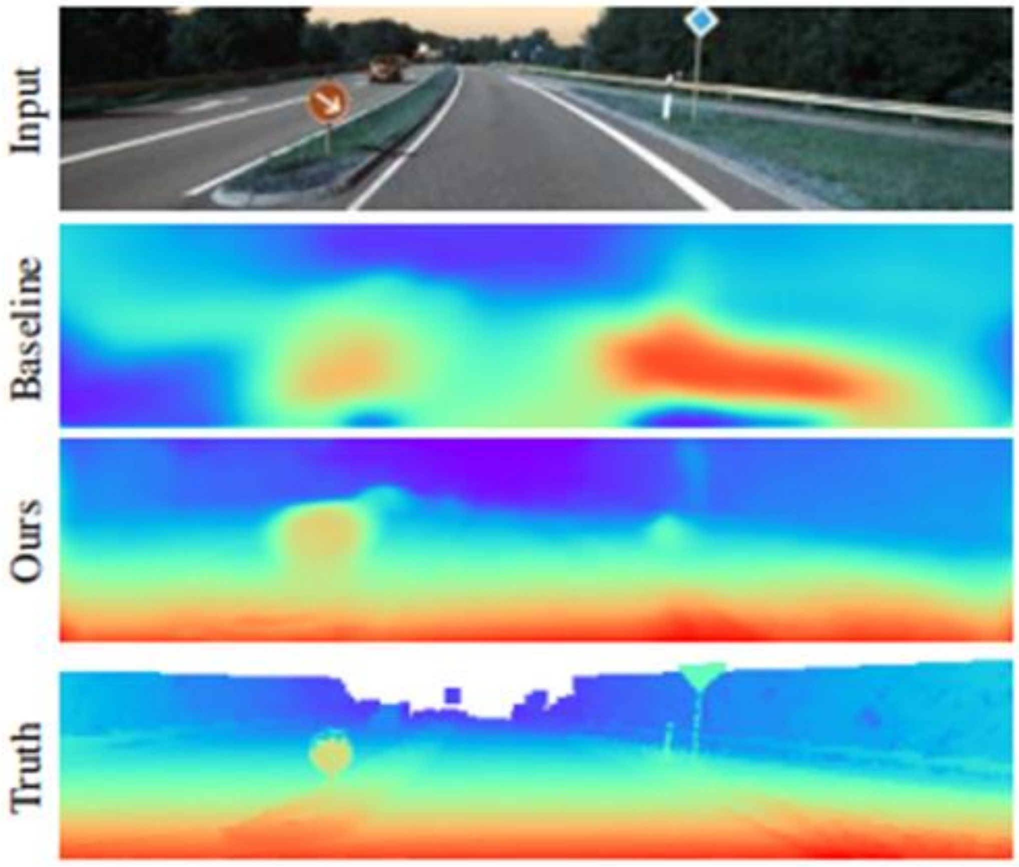


Figure 15 results on mono depth result when replacing only the loss to adaptive loss

III. Experimental Analysis

In order to assess the potential in the proposed method, several tests were carried. Throughout this work, all the tests were conducted on the Sentinel datasets.

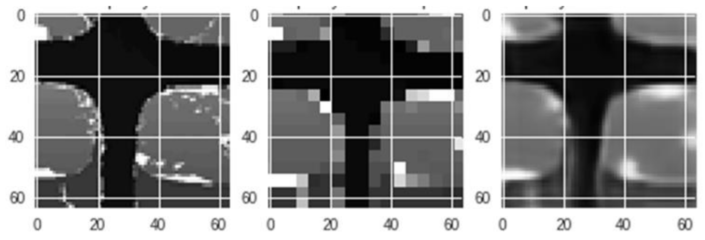
For simplicity, the following attributes were used for all experiments:

- Chosen optimizer is *ADA*M
- On each created dataset:
  - 80% randomly chosen examples were used for training.
  - The rest 20% were used for validation.
- Mini-batch size is 12.

1.JF -Net and activation function assessment

Due to the fact that the JointFilterNet has no implementation in pytorch, a verification of the model was done using patch images to ensure the network is capable to reach similar outputs as the original implementation. The network was trained using the stereo depth dataset MONKAA[16], using a resized image (T-Target) by down sample (x4) and up sample the Ground Truth Depth image from and used the corresponding Left image as the guidance image (G), under the assumption that down sampling and up sampling will add noise and quantize the contours in the original image. 50K patches of 64*64 were used for training and 10K for test.

The results can be shown from this sanity test in Figure 7, is that the JointFilterNet net successfully reconstructed the contour of the disparity image and smoothed the surface, similar to a joint bilateral filter.



Ground Truth *Quantified image* *Joint Filter Net Output*

Figure 16 Ground truth, Resized image, final map after JointFilterNet

Afterwards we used the same network to test the added value of changing the activation functions of the JF-net from RELU to ELU. Training on 10 Epochs with MSE loss, we can see from the plot in figure 8 that the ELU showed a loss lower by 25% from the Relu.

With that said, we must remember that the noise regime in this example is quite simple (quantization noise) and this result mostly justify that there is potential for changing activation functions.

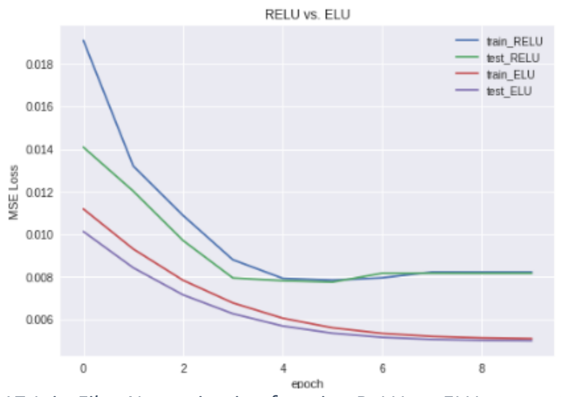


Figure 17 JointFilterNet activation function ReLU vs. ELU comparison

After the verification that the model is equip for testing, 2 networks were trained from scratch using the synthesized Mono -depth Sentinel dataset, when the only difference in training was the activation function.

It can be seen from the plots in figure 10 that in the early epochs, the loss of the ELU drops much faster, but after 100 epochs the ELU best validation loss is 11.7K while the Relu is 11.1K. proving that in this specific network the value of the ELU function is minimal.

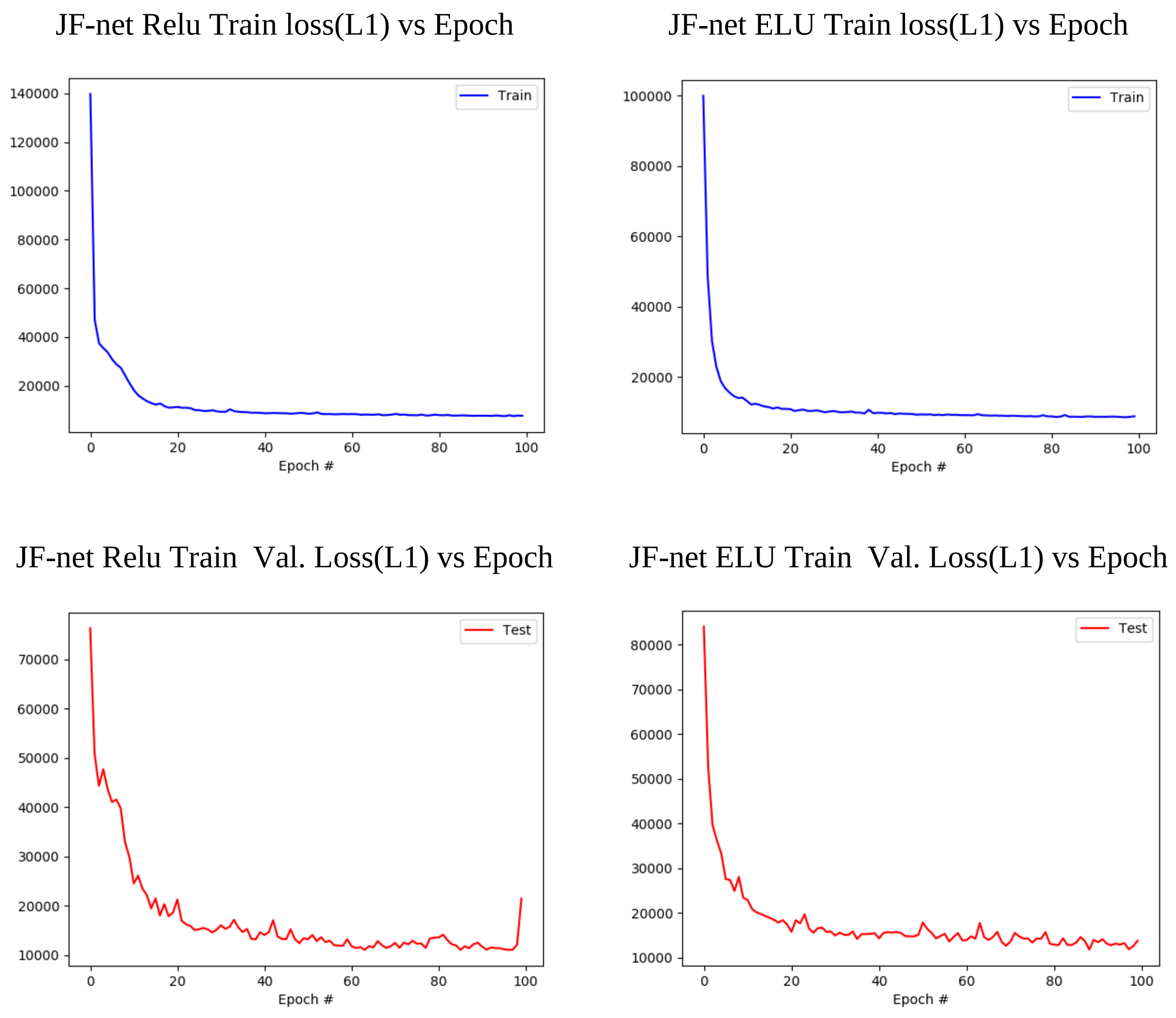
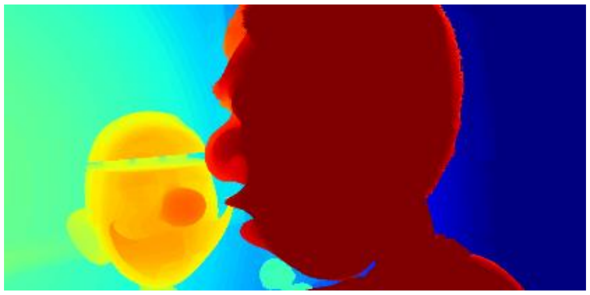


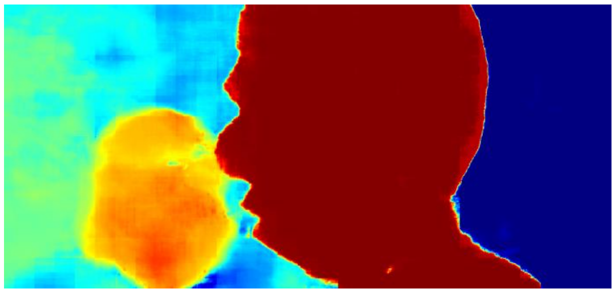
Figure 18 ELV vs RELU Sentinel Dataset

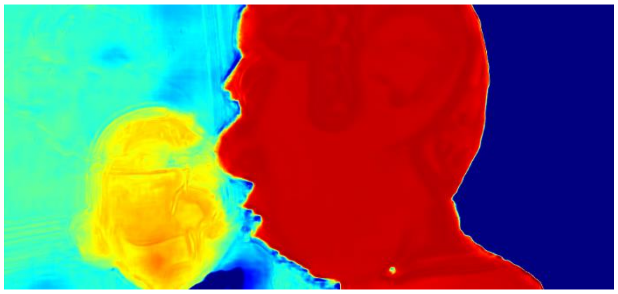
Method	L1 loss per image	L1 loss per pxl	output_loss/ input_loss
input validation data	2008442	4.670	1
JF_ReluL1 output	5460072	12.695	2.72
JF_ELU_L1 output	3005474	6.988	1.50

Figure 19 loss results JF-net

When observing the output images of the network it can be noted that both networks have sever artifacts, and none of them gives a sufficient solution for the targeted problem.


GT


Input image


JF-Net-Relu output

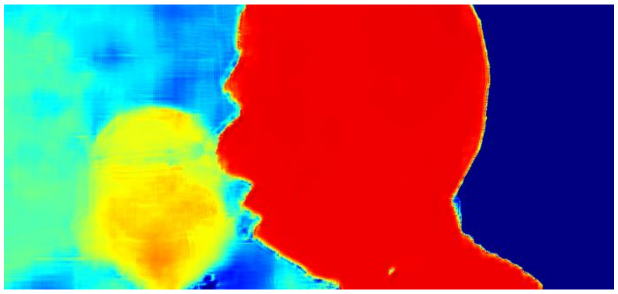

JF-Net-ELU output

Figure 20depth images of JF Net Relu vs Elu

Although the ELU showed promising potential, it also raises a flag if the amount of training data is sufficient in order to reach a NN without overfitting.

2. NN Content aware filter comparison (full training and only weights)

[The Deformable kernel network [DKN] (And it's fast implementation, FDKN) was tested if it can meet the requirements. The network had several advantages:

1. The network was trained on sufficient data and showed promising results on dataset which it wasn't trained on.
2. The low number of parameters and structure made it fast for retraining
3. It's novel approach which allowed smart weighting kernel with smart sampling allowed to reach SOTA results compared to other solutions, including the FJ-Net.

To test the capability of the FDKN , we preformed 100 epochs once in which all the weights were trained and one where all the weight are frozen except the two last layers, in which the weights and the offsets are calculated, this allows the network to use the feature extraction based on weights that learned on a large data set while preforming transfer learning to this images domain.

From assessing the loss of both approaches, it can be seen that training the full network reaches lower loss values and the general error decreases, while the training on the last 2 layers doesn't improve results.

When we observe the image, we can note that the overall training improves the results, but still seems that there are artifacts.

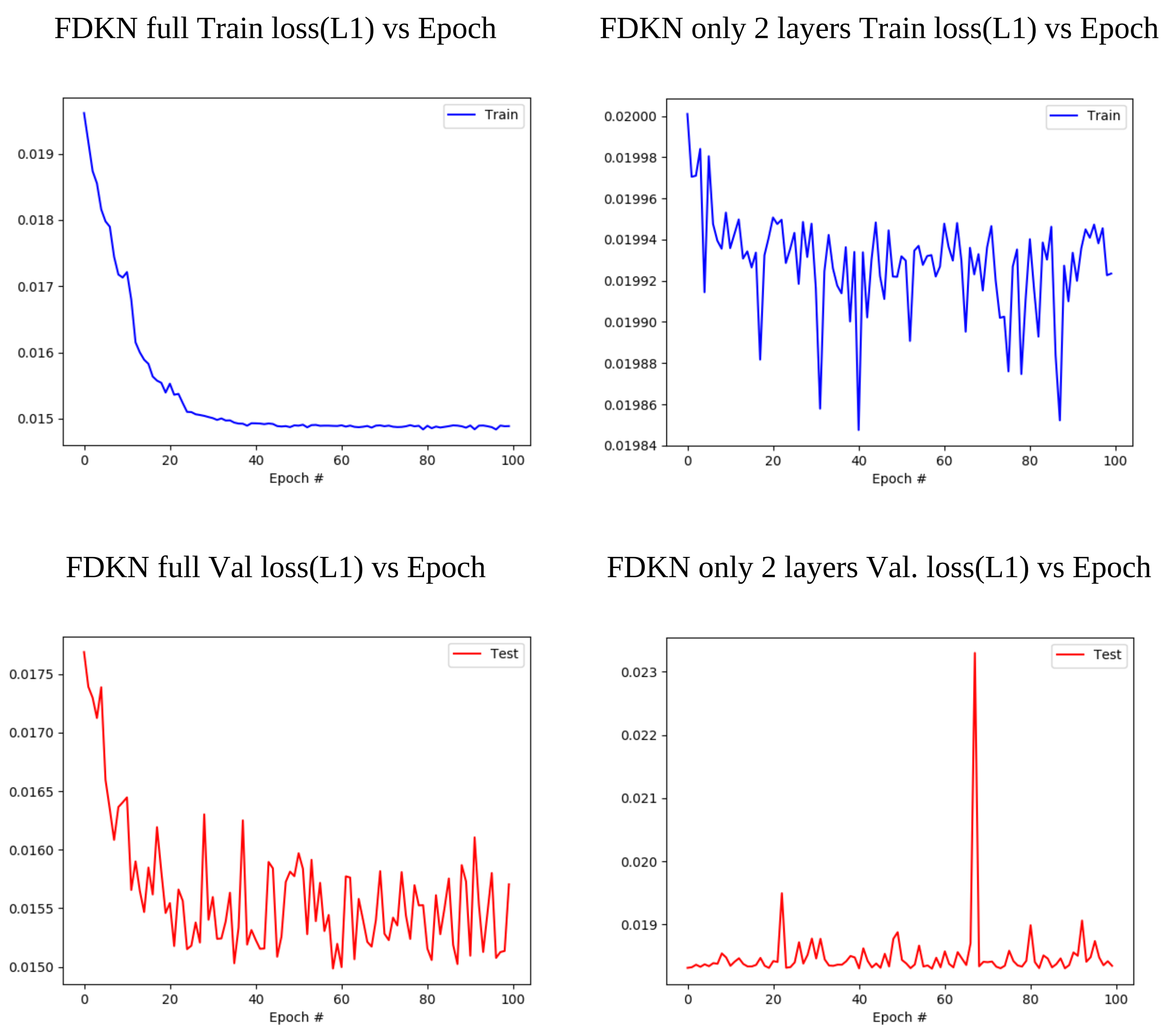


Figure 21 FDKN full train vs only last 2 layers Sentinel Dataset

Method	L1 loss per image	L1 loss per pxl	output_loss/ input_loss
input validation data	2008442	4.670	1
JF_ReluL1 output	5460072	12.695	2.72
JF_ELU_L1 output	3005474	6.988	1.50
FDKN_only last 2 layers L1 loss output	2007699	4.668	1.00
FDKN_All layer train L1 loss output	1643387	3.821	0.82

Figure 22 Results of FDKN vs Relu

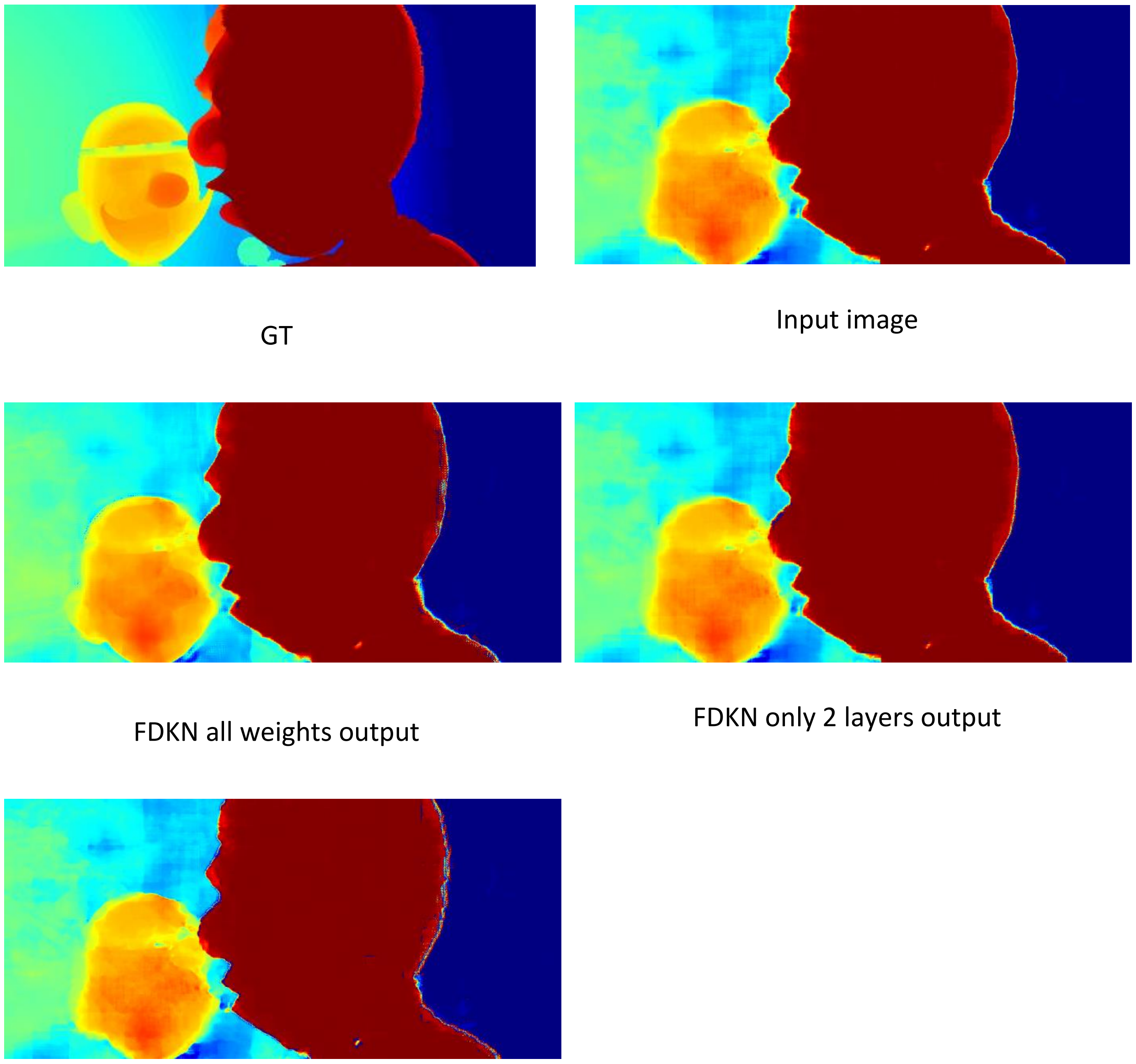


Figure 23 depth output comparison of FDKN

this leads us to the question, if we would a better optimized loss function, will the results improve or is the issue is in the network structure or data.

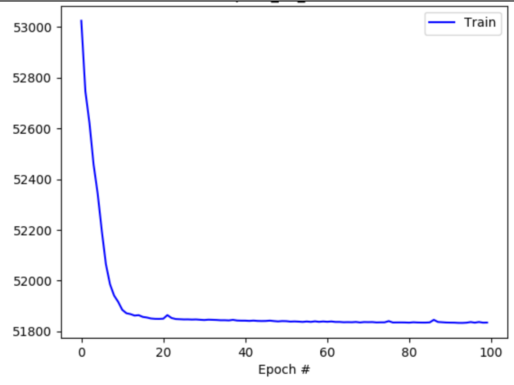
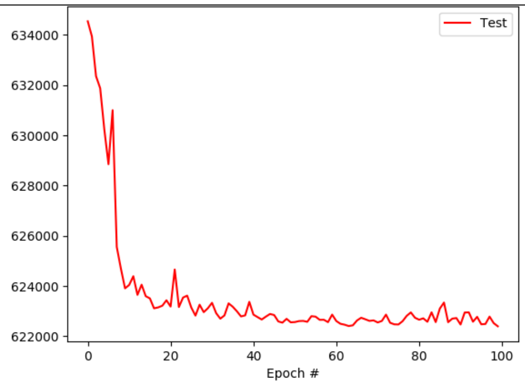
3. Adaptive loss assessment

Defining an effective loss function that allows representation of the error while taking consideration of the data is a key component in optimizing neural networks.

due to the fact that there most depth network use L1 or MSE loss without any explanatory version, we decided to use adaptive loss based on the distribution of the data.

Although we expected that the alpha value would change during the training, it stayed almost constant in the value on alpha equals 2 (which is equivalent to MSE loss).

in order to verify that all previous assessment have not changed, all considered networks were trained with adaptive loss

NN	Train (adaptive loss vs Epoch)	Test (adaptive loss vs Epoch)
JF-Net Relu		

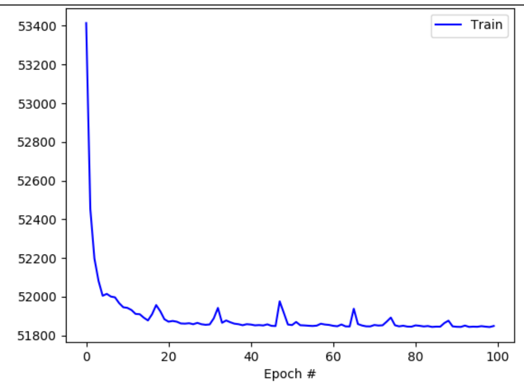
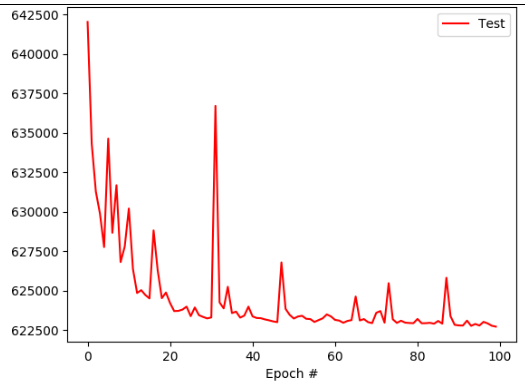
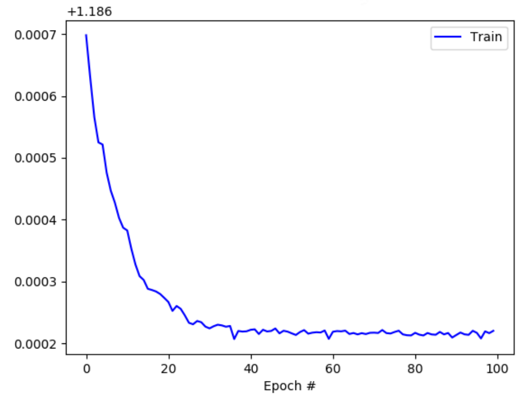
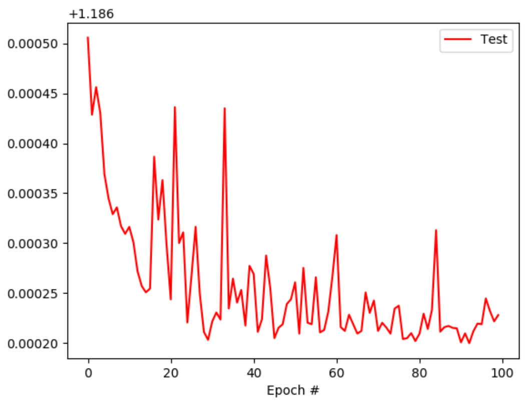
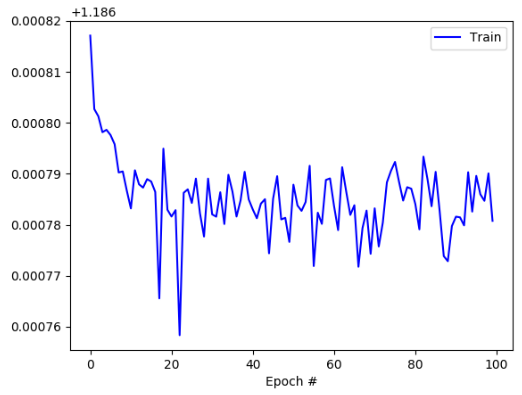
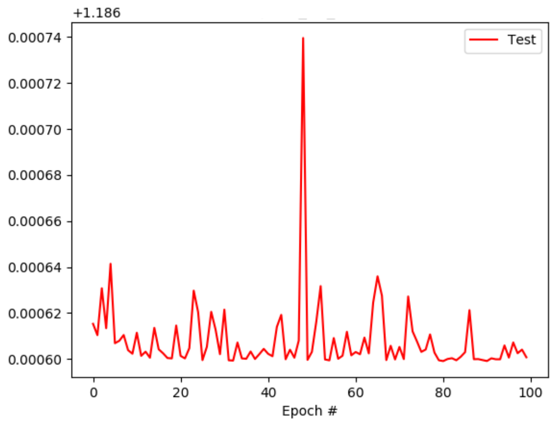
JF-Net ELU		
FDKN full		
FDKN only last 2 layers		

Figure 24 loss plot of adaptive loss functions

When preforming a full comparison between the L1 error of all batch images, we can see that the L1 error improved in ~ 5% compared to original depth output. This improvement shows potential of this method but puts in question it’s value in the overall.

it can also be seen that the results meet the results presented in the DFKN paper in which it has better results than the JF-Net.

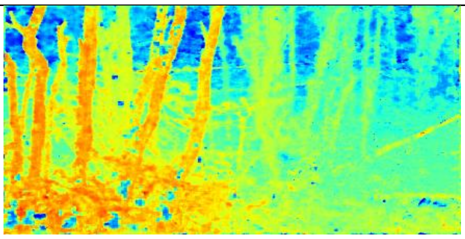
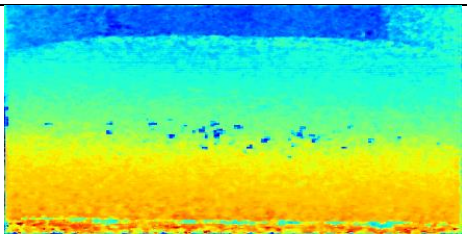
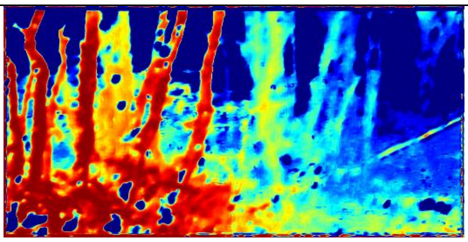
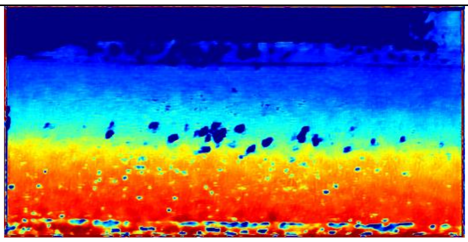
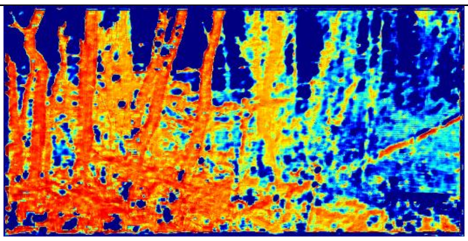
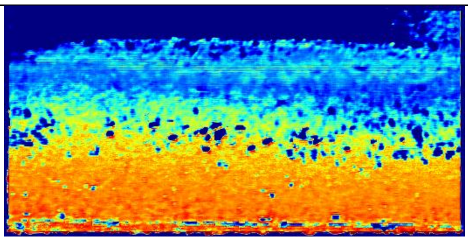
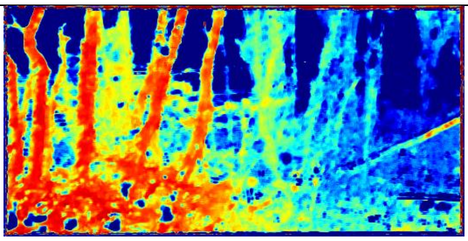
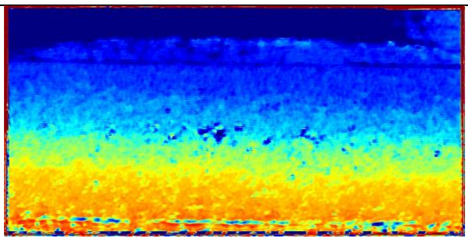
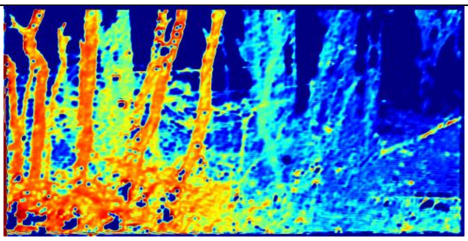
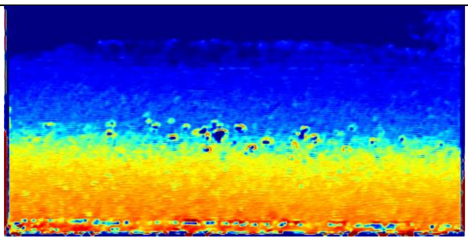
Method	L1 loss per image	L1 loss per pxl	output_loss/ input_loss
input validation data	2008442	4.670	1
JF_ReluL1 output	5460072	12.695	2.72
JF_relu_adaptive output	3236990	7.526	1.61
JF_ELU_L1 output	3005474	6.988	1.50
JF_ELU_Adaptive loss output	4070635	9.465	2.03
FDKN_All layer train adaptive loss output	**1606170**	**3.735**	**0.80**
FDKN_only last 2 layers adaptive loss output	2009825	4.673	1.00
FDKN_only last 2 layers L1 loss output	2007699	4.668	1.00
FDKN_All layer train L1 loss output	1643387	3.821	0.82

Figure 25 Results of different tested content aware neural networks

In addition , when observing the results based on the mono depth images , which don't have any ground truth, it can be seen that the FDKN in which transfer learning was preformed has a better imaging results than the FDKN in which had a complete train cycle.

this puts into consideration that the training of the dataset isn't sufficient to overcome the domain gap.

	13	16
RGB image		

Orig iput		
JF-Net_Relu- L1		
JF-Net_ELU- L1		
JF-Net_Relu- Adaptive loss		
JF-Net_ELU- Adaptive loss		

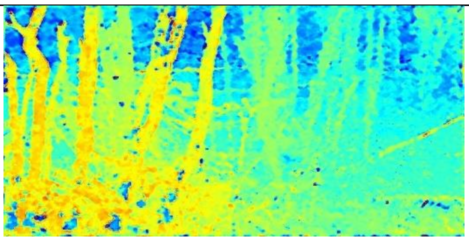
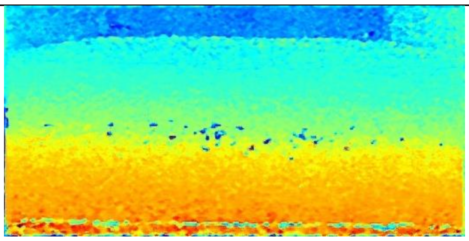
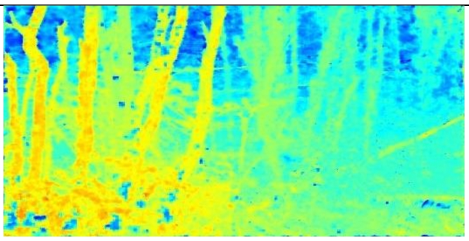
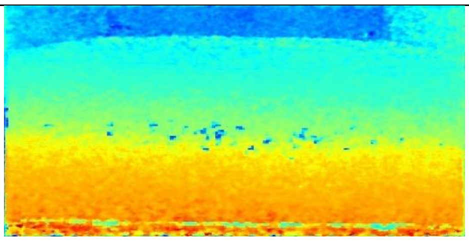
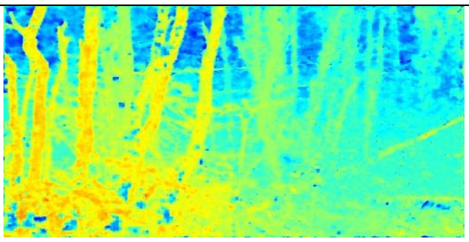
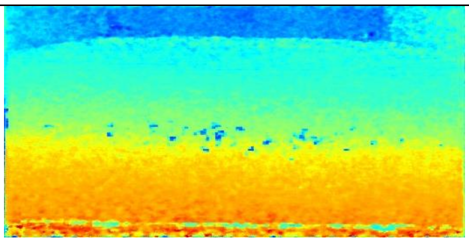
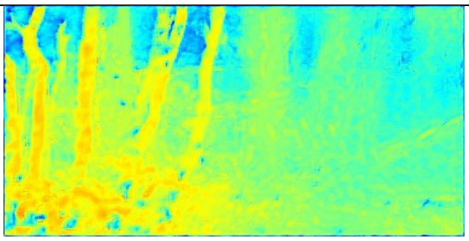
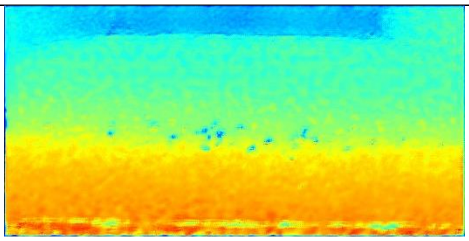
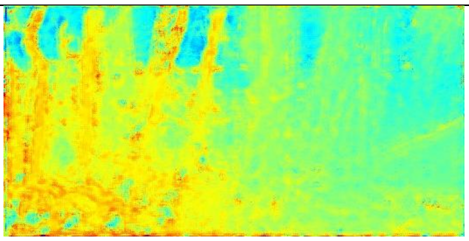
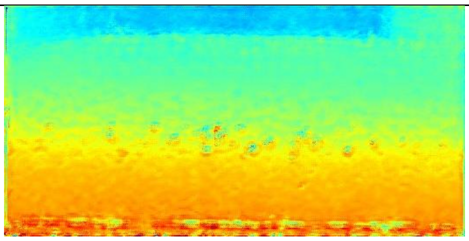
FDKN- original weights		
FDKN only last 2 layers- L1		
FDKN only last 2 layers- Adaptive		
FDKN L1		
FDKN - Adaptive loss		

Figure 26 depth images of real world mono images

From observation of all the data , it can be concluded that although there is potential in this method, it seems that in order to reach results which will be sufficient for real world problems require a large dataset , and although this method showed potential on the training and validation data reaching an improvement of 80% of the initial error, it still has not met the desired effect on real world captured images.

IV. Summary and Conclusions

The project was defined by three main stages as described in this report. First impending the joint filter network and testing the ELU vs the RELU activation function. Second, evaluating the Deformable kernel network (the current state of the art in content aware filters) , and assessing the add value of preforming transfer learning (training only the last 2 layers) or performing a full train of all layers in network. And finally, evaluating the use of a novel loss function, adaptive loss, which adapts the loss function based on the distribution of the error.

In the end, when assessing this project by its end goal of improving the depth estimation of an output neural network using an content aware filter, the project showed improvement of 80% of the initial error on synthetic dataset in which it was trained but couldn't pass the domain gap between real world data and synthetic data, reaching similar results to the input.

With that said, the project was able to show several conclusions which may assist and support projects, first and foremost is the project was able to show the potential of using of content aware filters in order to reduce error in depth images. Second is to address the importance of using sufficient data in order to minimize the possibility of overfitting and to ease the crossover between different data domains, and finally examine the option of use smarter loss function that adapt during training, although it didn't give the desired effect in this project, it has the potential of assisting in smarter and training based on data and not assumption.

V. References

[1] Gil, Y., Elmalem, S., Haim, H., Marom, E., & Giryes, R. (2019). MonSter: Awakening the Mono in Stereo. arXiv preprint arXiv:1910.13708.

[2] H. Haim, S. Elmalem, R. Giryes, A. Bronstein, and E. Marom (2018) Depth Estimation from a Single Image using Deep Learned Phase Coded Mask. IEEE Transactions on Computational Imaging, pp. 298 – 310

[3] Tomasi, Carlo, and Roberto Manduchi. "Bilateral filtering for gray and color images." Sixth international conference on computer vision (IEEE Cat. No. 98CH36271). IEEE, 1998.

[4] J. Kopf, M. F. Cohen, M. F. Cohen and M. Uyttendaele, "Joint Bilateral Upsampling," 2004.

[5] J. S. a. X. T. K. He, "Guided image filtering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, 2013.

[6] 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.

[7] 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.

[8] 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.

[9] X. Z. S. R. a. J. S. K. He, "Deep residual learning for image recognition," in EEE Conference on Computer Vision and Pattern Recognition, 2016.

[10] Dahl, George E., Tara N. Sainath, and Geoffrey E. Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout." 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013.

[11] Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." arXiv preprint arXiv:1511.07289 (2015).

[12] Hertel, Lars, et al. "Deep convolutional neural networks as generic feature extractors." 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015.

[13] Li, Yijun, et al. "Joint image filtering with deep convolutional networks." IEEE transactions on pattern analysis and machine intelligence 41.8 (2019): 1909-1923.

[14] Kim, Beomjun, Jean Ponce, and Bumsub Ham. "Deformable Kernel Networks for Joint Image Filtering." arXiv preprint arXiv:1910.08373 (2019).

[15] Barron, Jonathan T. "A general and adaptive robust loss function." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[16] Mayer, Nikolaus, et al. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.