Optimized loss function in deep learning profilometry for improved prediction performance

IOP Publishing J. Phys. Photonics 3 (2021) 024014 https://doi.org/10.1088/2515-7647/abf030

Journal of Physics: Photonics

OPEN ACCESS

RECEIVED
18 December 2020

REVISED
4 March 2021

ACCEPTED FOR PUBLICATION
18 March 2021

PUBLISHED
16 April 2021

PAPER

Optimized loss function in deep learning profilometry for improved prediction performance

Sam Van Der Jeught[1,*], **Pieter G G Muyshondt**[1] **and Ivan Lobato**[2]

[1] Laboratory of Biomedical Physics, University of Antwerp, Groenenborgerlaan 171, B-2020 Antwerp, Belgium
[2] EMAT, University of Antwerp, Groenenborgerlaan 171, B-2020 Antwerp, Belgium
[*] Author to whom any correspondence should be addressed.

E-mail: sam.vanderjeught@uantwerp.be

Keywords: profilometry, deep learning, structured light, Loss function

Abstract

Single-shot structured light profilometry (SLP) aims at reconstructing the 3D height map of an object from a single deformed fringe pattern and has long been the ultimate goal in fringe projection profilometry. Recently, deep learning was introduced into SLP setups to replace the task-specific algorithm of fringe demodulation with a dedicated neural network. Research on deep learning-based profilometry has made considerable progress in a short amount of time due to the rapid development of general neural network strategies and to the transferrable nature of deep learning techniques to a wide array of application fields. The selection of the employed loss function has received very little to no attention in the recently reported deep learning-based SLP setups. In this paper, we demonstrate the significant impact of loss function selection on height map prediction accuracy, we evaluate the performance of a range of commonly used loss functions and we propose a new mixed gradient loss function that yields a higher 3D surface reconstruction accuracy than any previously used loss functions.

1. Introduction

Structured light profilometry (SLP) is the technique of reconstructing the 3D surface map of an object by projecting predefined fringe patterns onto its surface and by observing it under an angle [1, 2]. From the recorded deformed fringe pattern, the full-field height map of the object can be extracted using various triangulation techniques that can be subdivided into different classes depending on the specific demodulation strategy that is employed. Since the recorded intensity map I_R of the deformed fringe pattern is a linear equation of background illumination I_B, intensity profile modulation I_M and phase map $\varphi(x,y)$ (equation (1)), a set of at least three different intensity maps I_R is needed per 3D measurement to extract the phase map $\varphi(x,y)$ analytically:

$$I_R(x,y) = I_B(x,y) + I_M(x,y)\cos(\varphi(x,y)). \tag{1}$$

To solve equation (1), phase shifting profilometry (PSP) techniques permute the modulation of I_M by integer multiples of $2\pi/n$ (with $n \geqslant 3$) between successive recordings of I_R. A larger number of phase shifts n per 3D measurement improves the measurement resolution, but also increases the measurement of 3D recordings, limiting the overall 3D recording speed. This results in the typical trade-off between speed and measurement accuracy.

Even though modern PSP systems have been reported to acquire high-resolution 3D measurements at real-time speeds [3, 4], several drawbacks to phase-shifting systems remain. For example, the 3D frame rate of PSP techniques is ultimately limited to $1/n$th of the camera frame rate which leads to an inevitable degree of motion artifacts to be present in the 3D measurements. Furthermore, PSP setups require custom synchronization between the digital light projector and the camera. Therefore, additional triggering hardware and onboard (flash) projector memory is needed to display a constant loop of fringe patterns. In contrast, single-shot profilometry techniques are designed to extract height information from a single deformed fringe

pattern. Generally speaking, all single-shot strategies are either adapted versions of Takeda's Fourier transform profilometry (FTP) [5] or employ additional color channels to superimpose differently modulated intensity patterns onto a single color map [6, 7]. While both approaches have been highly successful and are implemented into a variety of application fields, they each have specific drawbacks. For instance, FTP techniques depend on the correct selection of the originally projected carrier frequency, which is stretched in Fourier space through modulation of the object's surface shape. When the carrier frequency band overlaps with the other frequency components in Fourier space, it cannot be retrieved unambiguously, leading to loss of information especially near sharp edges. Color-sensitive approaches, on the other hand, assume that the object surface reflects different colors uniformly and that there is no crosstalk between adjacent color channels. When this is not the case, the quality of the resulting depth map deteriorates significantly.

Recently, we reported a new technique to obtain object height information directly from only a single deformed fringe pattern [8]. The proposed technique is based entirely on deep learning and uses a neural network to extract full-field height information from deformed fringe patterns with high accuracy. In addition, intermediate data processing steps (such as background masking, noise reduction, and phase unwrapping) that otherwise must be implemented explicitly in classic demodulation algorithms were learned directly as part of the network's mapping function. Since the publication of this proof-of-principle paper, several implementations of deep learning-based SLP systems have been reported, exploring adaptations to the network architecture, training dataset and model hyperparameters. None of them, however, addresses the choice of loss function that is used to train the model. In fact, nearly all models employ the L_2 or mean squared error (MSE)-loss [9–15], one uses smooth L_1-loss [16] and some do not mention the employed loss function in the training of their model at all [17–19]. According to Wang *et al* [20], the popularity of the L_2-norm in the training of neural networks can be explained by several arguments: first, it is easy and computationally inexpensive to calculate. Second, it has a clear physical meaning—it is the natural way to define the energy of the error signal. Third, it is convex, symmetric and differentiable, making it an excellent metric in the context of optimization problems. Finally, and perhaps most importantly, it is readily implemented in nearly every existing deep learning framework as the default loss function. As we will see, the L_2-norm also possesses some properties that make it a questionable candidate to be implemented as loss function of choice in some contexts. For example, several pre-assumptions are made when the L_2-norm is used: signal fidelity is assumed to be independent of spatial or temporal relationships between samples, independent of the signs of the error signal and all signal samples are to be equally important to signal fidelity. In any case, the loss function is one of the key components of any neural network: it is the metric that defines the prediction error and its gradient is used to update the network weights with each pass over the training dataset. In this work, we demonstrate that the choice of loss function in a deep learning-based SLP setup has a significant impact on prediction accuracy, we evaluate the performance of several common loss functions and we propose a custom mixed gradient loss function that yields a higher prediction accuracy than any of the other investigated loss functions.

2. Methods

2.1. Data set

We use a dataset of simulated height maps with corresponding deformed fringe patterns as data couples to train the neural network. To automate the process of constructing a large training dataset, a random surface map generator is designed. Input parameters such as boundary limits, number of peaks, and sharpness or smoothness of the edges can all be set and are randomly varied within predefined limits to produce a set number of randomly varying 3D height maps. Reflectivity is assumed to be uniform and no additional noise is added to the simulated fringe maps. After the creation of a set of randomly fluctuating height maps, a predefined fringe pattern is projected virtually onto each height map and the resulting surface modulation is observed under a fixed angle α. Using straightforward triangulation, the modulated fringe pattern is sampled on a Cartesian grid. The complete process of data-couple generation is illustrated in figure 1.

The simulated dataset used in this work aims to mimic images obtained with real SLP setups. Consequently, realistic values for fringe pattern frequency (six periods per image width), the angle between projection and observation axes ($\alpha = 30°$), and maximum width to height ratio (4:1) are chosen in the generation of our dataset. Z-values were confined to $z \in \mathbb{R}^3\,[0, 32]$. Note that the maximum z-value of 32 does not necessarily need to be attained for every surface map; it is simply the upper limit. The 4:1 ratio between image width and maximum surface height reduces the amount of shadows that may arise during the fringe modulation process. Deformed fringe patterns and height maps are each sampled on a grid of 128×128 pixels. After fringe modulation, volumetric boundary limits of the surface maps in our dataset are normalized to $x, y, z \in \mathbb{R}^3\,[0, 1]$. For a more detailed description of the training data set generation, we refer the reader to [8].

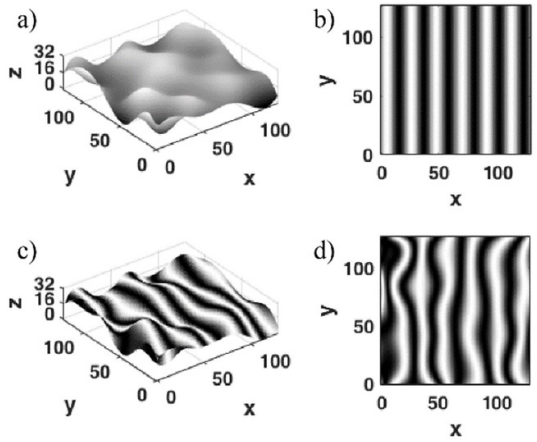

Figure 1. Fringe modulation process. First, a randomly generated height map is generated (a). Next, a predefined sinusoidally modulated fringe pattern (b) is projected onto it, after which the resulting surface image (c) is observed under an angle α (here $\alpha = 30°$). Finally, the deformed fringe pattern (d) is sampled onto a Cartesian grid and stored.

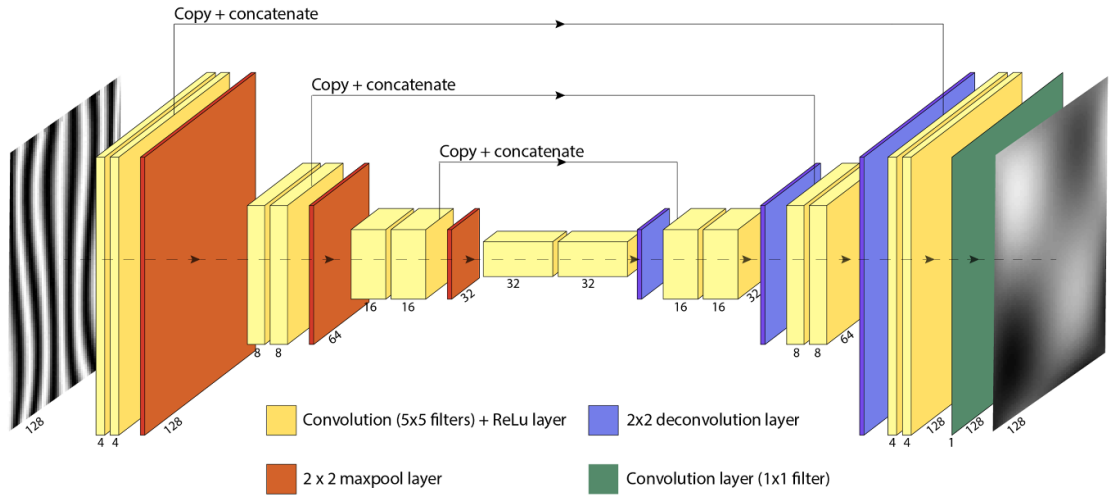

Figure 2. Network architecture of the modified U-net. Three downsampling and three upsampling blocks are placed symmetrically between input and output images. Downsampling blocks consist of two convolutional layers followed by a 2×2 max-pooling layer, upsampling blocks consist of two convolutional layers preceded by a deconvolutional layer. ReLu activation functions are used after each convolutional layer, except for the final single-channel convolutional layer.

2.2. Network architecture

The neural network employed in this work is a modified version of the U-net proposed by Ronneberger *et al* [21]. A schematic representation of its architecture is shown in figure 2. The modified U-net can be divided symmetrically into a contracting path (encoder) and an expansive path (decoder). The contracting path consists of a succession of convolutional layers with 5×5 filters, each followed by a rectified linear unit (ReLu) activation function. After every two successive convolutional layers, a 2×2 max-pooling operation is applied. Together, two convolutional layers and one max-pooling layer form a downsampling block. Each downsampling block halves the spatial size of the input representation but doubles the number of feature maps, starting with four feature maps for the first block, eight for the second, and so on. The purpose of the contracting path is to capture the context of the input image by increasing the receptive field of the feature maps. By doubling the number of feature maps with each downscaling block, the architecture can detect the complex structures that are present in the input image more efficiently [22]. The expansive path is symmetrically identical to the contracting path, except that the max-pooling operations are replaced by a 2×2 deconvolutional layer that precedes the convolutional layers in each upsampling block. The deconvolutional (or transposed convolutional) layers upsample the input feature map by a factor of 2. With each deconvolutional operation, the number of feature maps is halved until finally a single convolutional layer with a 1×1 filter is applied to the output of the last upsampling block. By using a symmetric network architecture, the modified U-net produces an output image of identical size as the input image.

Note that each upsampling block has access to the output feature map of its corresponding downsampling counterpart through concatenated skip connections between them. This allows the expansive path to reuse the previously learned local features, allowing information to be retained directly from previous layers of the network. In total, our model contains 78 997 free parameters.

2.3. Loss functions

2.3.1. L_1 and L_2

Single-shot deep learning-based profilometry aims to learn a mapping function $\hat{Y} = f(X)$ that transforms a 2D modulated fringe map (X) to a continuous 3D height distribution ($\hat{Y}$) without any additional intermediary processing steps. The loss function is the metric that defines how close the predicted height map $\hat{Y}$ is to the ground truth Y. Common metrics that are used as loss function in the training of neural networks are L_1 or mean absolute error (MAE) and L_2 or MSE:

$$L_1(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |\widehat{y_i} - y_i| \tag{2}$$

$$L_2(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} (\widehat{y_i} - y_i)^2 \tag{3}$$

with n the number of data points. From their definitions, some properties of L_1- and L_2-norms can be deducted when they are used as loss function in deep learning models. First, it can be seen that L_2 is more sensitive to outliers in the dataset compared to L_1 since the differences between prediction and ground truth are squared. As a result, L_2 will adjust the model to minimize the error caused by these single outlier cases at the expense of many other data points if they contribute only little to the general error. On the other hand, L_1-norms produce less stable solutions, meaning that for a small horizontal adjustment of a single data point the slope of the regression line between data samples may jump a large amount. Selection between L_1 and L_2-norms as loss function of choice in a deep learning model is made primarily based on this trade-off between robustness and stability and which of these properties is more desirable for the given minimization problem.

2.3.2. SSIM and MS-SSIM

The structural similarity index (SSIM) is a metric that is frequently used as loss function in single-image super-resolution models [23] and other tasks that involve human perceptual judgment. The SSIM and its multi-scale variant MS-SSIM are loss functions that, unlike L_1 and L_2, are based on the human visual system as they aim to match the luminance I, contrast C, and structural information S between images. The SSIM is defined as:

$$\mathrm{SSIM}(y, \hat{y}) = I(y, \hat{y})^{\alpha} C(y, \hat{y})^{\beta} S(y, \hat{y})^{\gamma} \tag{4}$$

with

$$I(y, \hat{y}) = \frac{2\mu_y \mu_{\hat{y}} + C_1}{{\mu_y}^2 + {\mu_{\hat{y}}}^2 + C_1}$$

$$C(y, \hat{y}) = \frac{2\sigma_y \sigma_{\hat{y}} + C_2}{{\sigma_y}^2 + {\sigma_{\hat{y}}}^2 + C_2}$$

$$S(y, \hat{y}) = \frac{\sigma_{y\hat{y}} + C_3}{\sigma_y \sigma_{\hat{y}} + C_3}.$$

The variables μ_y, $\mu_{\hat{y}}$, σ_y and $\sigma_{\hat{y}}$ denote mean pixel intensity and standard deviations of pixel intensity in a local image patch at either y or $\hat{y}$, respectively. In the following, we choose a square region of 3 pixels on either side of y or $\hat{y}$, which results in patches of 7×7 pixels. The variable $\sigma_{y\hat{y}}$ denotes the sample correlation coefficient between corresponding pixels in the patches centered at y and $\hat{y}$. The constants C_1, C_2 and C_3 are small constant values that are added for numerical stability and α, β and γ define the weights given to each component of the SSIM.

The SSIM metric assumes a fixed image sampling density and is a single-scale index. An extension to this metric is the multi-scale variant MS-SSIM, which can operate at multiple scales simultaneously. The ground truth image y and its candidate approximation image $\hat{y}$ are both iteratively downsampled by a factor of 2. The contrast C and structural S components are applied at all scales. The luminance component I, however, is

only applied at the coarsest scale, denoted M. Again, the weighting of the different components is allowed at each scale, leading to the following definition of the MS-SSIM:

$$\text{MS-SSIM}\,(y,\hat{y}) = I_{\text{M}}(y,\hat{y})^{\alpha}\prod_{j=1}^{M} C_j(y,\hat{y})^{\beta_j} S_j(y,\hat{y})^{\gamma_j}. \tag{5}$$

Note that both SSIM and MS-SSIM are normalized to $[0,1]$ where 1 indicates a full correspondence between y and $\hat{y}$. Since the loss function employed in a neural network is to be minimized, we use 1–SSIM and 1–MS-SSIM instead.

2.3.3. Mean gradient error (MGE) and mixed gradient error (MixGE)
In SLP, we are mainly concerned about relative phase differences. Two surface shapes can be assumed to be identical if their pixel-wise phase difference is a constant value: subtracting the minimum, mean or maximum value of the surface map from each pixel value suffices to unwrap the phase map of dynamically moving surface shapes temporally. It, therefore, makes sense to incorporate the local gradient of the prediction ($\hat{Y}$) and the ground truth (Y) surface maps directly into the loss function. Inspired by [24], we introduce classic gradients to the loss function using Sobel operators:

$$G_x = Y * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = Y * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

where $*$ denotes the convolution operator. Together, the gradients G_x and G_y are combined into the general pixel-wise gradient map $G = \sqrt{{G_x}^2 + {G_y}^2}$. Likewise, the gradient map $\hat{G}$ of the predicted surface map $\hat{Y}$ can be calculated. The MGE can then be defined as:

$$\text{MGE} = \frac{1}{n}\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(G(i,j) - \hat{G}(i,j)\right)^2.$$

Using the MGE directly as a stand-alone loss function of the neural network would result in divergence of the model since the DC component is completely removed from the solution. The MGE can, however, be used as an auxiliary component by adding it to the standard L_1-norm in the creation of a MixGE:

$$\text{MixGE} = (1-\lambda)\,L_1 + \lambda\,\text{MGE} \tag{6}$$

where $\lambda \in [0,1]$ is the gradient error weight factor that needs to be optimized in function of the employed model architecture and dataset.

2.4. Training the network

To train our network, a dataset of $N_{\text{total}} = 12\,500$ simulated data couples is created using the automated surface generator. The number of local surface maxima and minima that are present in the simulated height maps varies randomly between 0 (flat planes of random height) and 15 (highly fluctuating landscapes). Either linear or spline interpolation is used randomly to sample the simulated surfaces on a 128×128 grid between the minima and maxima, resulting in a mix of surface maps containing both sharp and smooth edges, respectively. A training dataset of $N_{\text{train}} = 10\,000$ simulated input-output pairs is used to train the network and a validation dataset of $N_{\text{val}} = 2500$ simulated input-output pairs is used to test the performance of the model on previously unseen data.

The Tensorflow framework [25] is used to train the modified U-net. The batch size is set at 4 and roughly 100 full passes (or epochs) through the training dataset are completed, depending on the individual convergence of the network trained with each employed loss function. We use Adam optimization with an initial learning rate of 10^{-4}. Before training, the initial values of the filter weights are set using Xavier initialization to ensure equal variance between successive layers. To regularize training and to reduce overfitting, L_2 weight decay is used with an initial setting of 10^{-3}. After every 50 000 iterations, the learning rate is divided by 5, and L_2 penalties are divided by 10. During the final set of iterations, L_2 penalties are set to 0. The number of training iterations was chosen so that for each loss function investigated in this paper

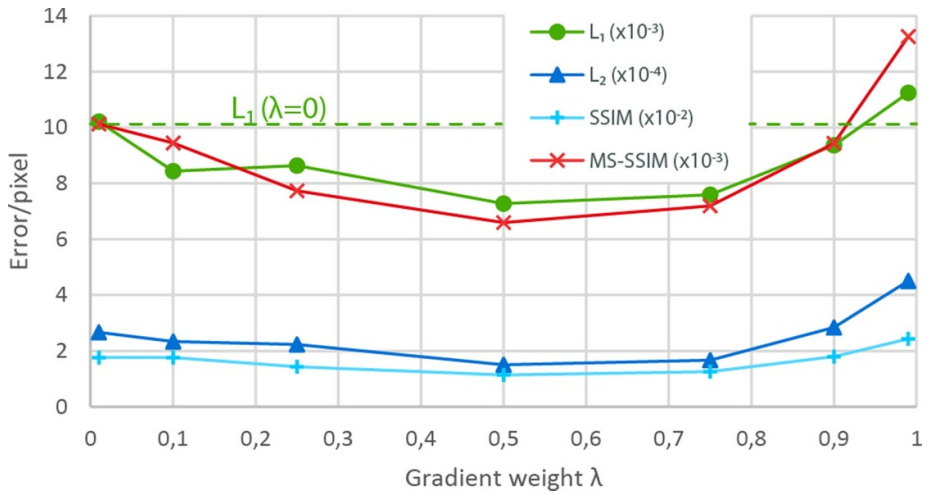

Figure 3. Optimization of the gradient weight λ shows an optimal region between $\lambda = 0.4$ and $\lambda = 0.6$ for which the error per pixel of the model trained with the mixed gradient error reaches the minimum for various metrics.

(L_1, L_2, SSIM, MS-SSIM, and MixGE), the validation error reached convergence. Convergence is defined here at the point where the validation error does not improve for five consecutive epochs. In addition, validation curves are monitored to confirm the general converging behavior of the network and to ensure that no faulty divergence is occurring. Our implementation uses cuDNN and is optimized for parallel execution. The training time of a single model varies depending on the employed loss function. In total, the training time for a single model ranges between 34 min (MAE), 35 min (MSE), 36 min (MixGE), 42 min (SSIM), and 62 min (MS-SSIM) when deployed on a Titan X Pascal GPU. These convergence times correspond to a total number of training epochs of 98, 102, 92, 106, and 119 for networks trained using L_1, L_2, MixGE, SSIM, and MS-SSIM, respectively.

3. Results

3.1. Optimization of gradient weight λ

First, the modified U-net model was trained using various ratios of MGE/L_1 to find the optimal contribution of the MGE component to the total MixGE. Several values of λ ranging from 0 to 0.99 were used in equation (6) to modify the impact of the MGE component. Each resulting MixGE loss function was used to train the model to convergence on a fixed dataset. Upon convergence, the model performance was evaluated on the validation data set using the L_1,L_2, SSIM, and MS-SSIM metrics. The result is shown in figure 3. The models trained with the MixGE outperform those trained with standard L_1 loss up to a value of $\lambda \approx 0.95$. Beyond this point, the MGE component becomes too dominant over the DC component provided by L_1, resulting in divergence of the model and a steep increase in prediction error. Although the various metrics provide different evaluations of the models trained using the differently weighted MixGE loss functions, they all reach a minimum error per pixel when λ is chosen between $\lambda = 0.40$ and $\lambda = 0.60$. In the following, we will use a gradient weight of $\lambda = 0.50$ in the MixGE loss function.

3.2. Performance of various loss functions

The modified U-net was trained to convergence using each of the L_1, L_2, SSIM, MS-SSIM, and MixGE loss functions. The performance of each trained model was evaluated by comparing the height map predictions ($\hat{Y}$) of the samples in the validation set to their respective ground truth (Y) height maps using each of the corresponding metrics of L_1, L_2, SSIM, MS-SSIM, and MixGE. The resulting error values were averaged over the entire validation set and are included in table 1.

The top part of table 1 (General) represents the performance matrix over the entire validation dataset, containing 2500 simulated height map samples with a random mix of both smooth (spline-interpolated) and sharp (linearly interpolated) edges. The MixGE loss function outperforms the other loss functions on this general population of height maps by a significant margin, even when the evaluation metric is the same as the one that was used as loss function in the training of the model. Only when the MS-SSIM metric is used, the model trained with the MixGE loss function is scored second best after the MS-SSIM loss function itself. Top performing metric is underlined.

Table 1. Evaluation of the modified U-net trained using different loss functions. Top: the general average of the model performance on the entire validation dataset. Middle: model performance on a validation subset composed of entirely spline-interpolated surface maps. Bottom: model performance on a validation subset composed of entirely linearly interpolated surface maps.

Loss function Metric	L_1	L_2	SSIM	MS-SSIM	MixGE
General					
L_1 ($\times 10^{-3}$)	10.01	10.52	13.40	11.21	7.27
L_2 ($\times 10^{-4}$)	2.45	1.62	1.81	1.74	1.49
SSIM ($\times 10^{-2}$)	1.83	1.85	1.25	1.27	1.13
MS-SSIM ($\times 10^{-3}$)	10.03	8.52	6.84	6.49	6.59
MixGE ($\times 10^{-3}$)	2.41	2.56	2.45	2.28	2.09
Spline					
L_1 ($\times 10^{-3}$)	10.04	10.21	12.93	11.44	7.35
L_2 ($\times 10^{-4}$)	2.50	1.56	1.77	1.76	1.46
SSIM ($\times 10^{-2}$)	1.81	1.72	1.24	1.26	1.17
MS-SSIM ($\times 10^{-3}$)	10.44	8.14	6.87	6.29	6.65
MixGE ($\times 10^{-3}$)	2.40	2.46	2.40	2.31	2.11
Linear					
L_1 ($\times 10^{-3}$)	10.01	11.46	13.52	11.11	7.15
L_2 ($\times 10^{-4}$)	2.41	1.69	1.78	1.73	1.57
SSIM ($\times 10^{-2}$)	1.87	2.02	1.25	1.27	1.08
MS-SSIM ($\times 10^{-3}$)	9.89	8.98	6.89	6.60	6.52
MixGE ($\times 10^{-3}$)	2.48	2.88	2.46	2.23	2.08

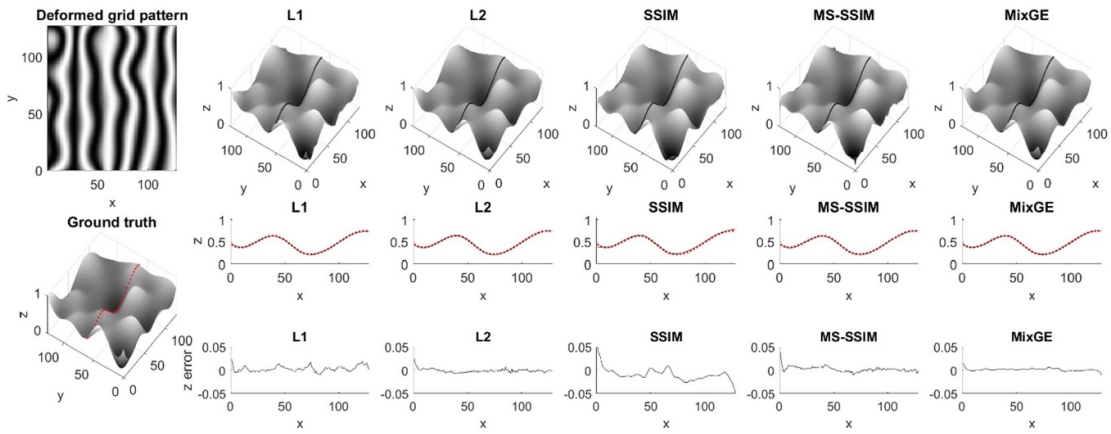

Figure 4. Sample from the validation set composed of spline-interpolated height maps. X- and Y-coordinates are displayed as pixel numbers on a 128 × 128 grid; Z-values are normalized to [0, 1].

To further investigate the effect of surface shape variation on loss function performance, two additional validation datasets of each $N = 1000$ data couples were created using the random surface map generator: one in which only spline interpolation was used to connect the peaks and troughs present in the height map, and one in which only linear interpolation was used. From the middle and bottom parts of table 1, a large discrepancy can be observed between L_2-performance on samples with smooth edges (Spline) and those with sharp edges (Linear). For all metrics, L_2 performs significantly worse on surfaces with sharp edges than it does on surfaces with smooth edges. This behavior is reflected in the samples illustrated in figures 5 and 6. Whereas models trained with L_1, SSIM, MS-SSIM, and MixGE loss functions perform similarly on spline-interpolated height maps (figure 4) as on linearly interpolated height maps (figure 5), the model trained with L_2-loss produces a larger error when applied to predict height maps with sharp edges than when applied to predict height maps with smooth, curved edges.

To highlight the superior performance of MixGE over the other loss functions, we include the measurement of the surface shape of a mannequin doll head in figure 6. The ground truth 3D surface measurement was obtained using four-step phase-shifting profilometry. Parts of the surface map where shadows, saturated reflections or any other regions of invalid data corrupted the 3D measurement were interpolated in post-processing. It can be seen that the model trained with MixGE does a significantly better job of predicting the mannequin doll surface shape than the models trained with the other loss functions. It should be noted that a clear linear error is present across all prediction maps, including the one

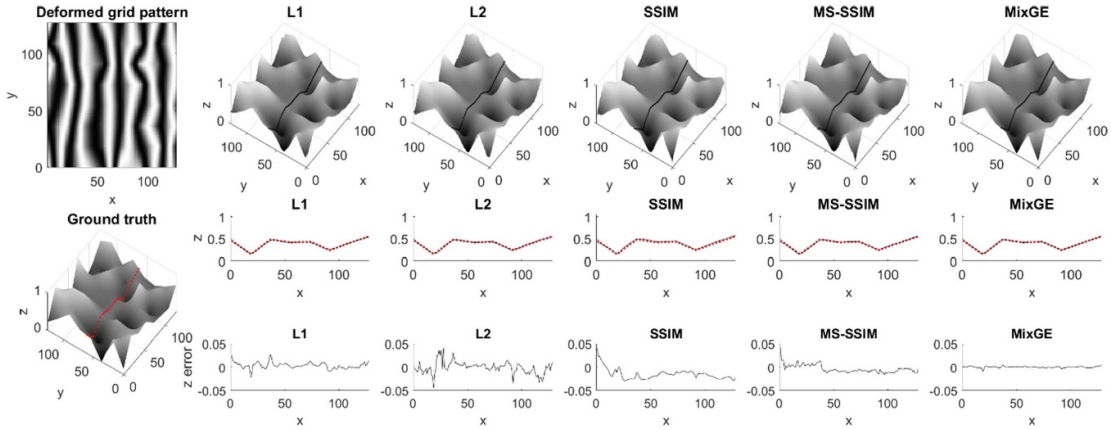

Figure 5. Sample from the validation set composed of linearly interpolated height maps. L_2-loss struggles to resolve the sharp edges present in the height map and produces a higher error. X- and Y-coordinates are displayed as pixel numbers on a 128 × 128 grid; Z-values are normalized to [0, 1].

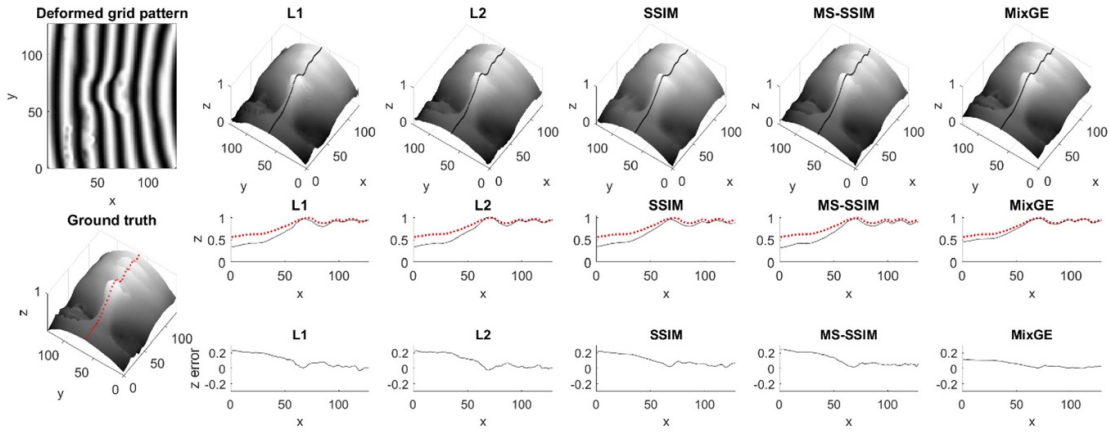

Figure 6. Mannequin doll head sample obtained with standard 4-step phase-shifting profilometry demonstrates superior prediction performance of the model trained with MixGE loss function. X- and Y-coordinates are displayed as pixel numbers on a 128 × 128 grid; Z-values are normalized to [0, 1].

Table 2. Evaluation of the modified U-net trained using different loss functions when Poisson noise is added to the deformed fringe patterns.

Loss function Metric	L_1	L_2	SSIM	MS-SSIM	MixGE
Poisson noise					
L_1 ($\times 10^{-3}$)	10.66	10.92	18.23	19.48	9.13
L_2 ($\times 10^{-4}$)	3.08	1.92	4.72	4.26	1.78
SSIM ($\times 10^{-2}$)	2.14	2.15	1.41	1.55	1.33
MS-SSIM ($\times 10^{-3}$)	10.98	9.01	7.90	7.14	7.07
MixGE ($\times 10^{-3}$)	2.74	2.93	2.93	5.77	2.28

reconstructed with the MixGE loss. This may be due to the strong localized height variations near and above the eye region of the mannequin head model. In the training dataset, no samples with such relatively large jumps in Z-value have been included and none of the loss functions seem to handle these properly.

Finally, to assess how the different loss functions are influenced by the presence of noise in the fringe patterns, an additional validation set of $N = 1000$ data couples was generated in which each input fringe map contained Poisson noise with the pixel value as the mean (before normalization). The corresponding error matrix is included in table 2 and a random sample from the validation set is shown in figure 7. First of all, it can be seen that the models trained with the L_1, L_2 and MixGE loss functions are able to extract the general form of the 3D map sufficiently well. As the neural network was trained to produce smooth, noise-free surface maps only, the predicted height maps are largely noise-free even when the input maps themselves are not. This way, noise filtering is implicitly learned by the network. The models trained with the SSIM-based

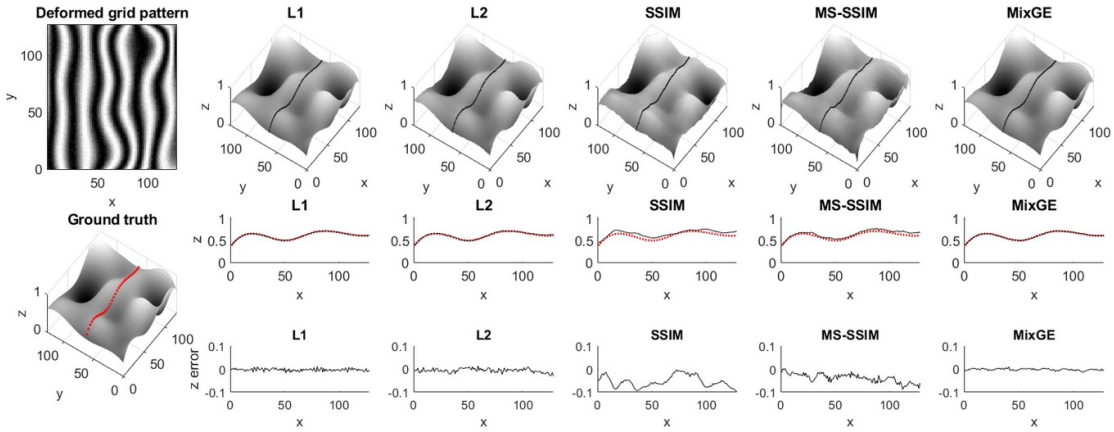

Figure 7. Sample from the validation set composed of input fringe maps that were corrupted with Poisson noise. The models trained with L_1, L_2-and MixGE-loss functions are relatively unaffected by the added noise, but the models trained with SSIM-based loss functions experience larger structural errors. X- and Y-coordinates are displayed as pixel numbers on a 128×128 grid; Z-values are normalized to $[0, 1]$.

losses, however, are more affected by the increased noise level. It should be noted that we assess 'pure' SSIM loss-prediction accuracy here: a mixed loss function containing an additional contribution of L_1- or L_2-loss might alleviate this oversensitivity to the added shot noise of the SSIM-family of loss functions. It can be seen from table 2 that MixGE outperforms all other loss functions by a significant margin.

4. Conclusion

We demonstrated that careful selection of the loss function can considerably improve the prediction accuracy of a neural network that is trained to reconstruct height maps from single deformed fringe patterns. Our results show clearly that the performance of the commonly used L_2-loss is highly dependent on the nature of the surface shape: it performs significantly better when applied to height maps with curved, smooth surface variations than it does on surfaces with sharp edges. We introduced the MixGE loss function, which takes into account the local gradient variations of the height maps, and showed that it outperforms any previously used loss functions in deep learning profilometry.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

S Van der Jeught and P Muyshondt thank the Research Foundation—Flanders (FWO) for their financial support. The Titan X GPU used for this research was donated by the NVIDIA Corporation.

ORCID iD

Sam Van Der Jeught https://orcid.org/0000-0002-4184-6147

References

[1] Salvi J, Fernandez S, Pribanic T and Llado X 2010 A state of the art in structured light patterns for surface profilometry *Pattern Recognit.* **43** 2666–80
[2] van der Jeught S and Dirckx J J J 2016 Real-time structured light profilometry: a review *Opt. Lasers Eng.* **87** 18–31
[3] Nguyen H, Nguyen D, Wang Z, Kieu H and Le M 2015 Real-time, high-accuracy 3D imaging and shape measurement *Appl. Opt.* **54** 9–17
[4] Zhang S, Royer D and Yau S-T 2006 GPU-assisted high-resolution, real-time 3D shape measurement *Opt. Express* **14** 9120–9
[5] Takeda M and Mutoh K 1983 Fourier transform profilometry for the automatic measurement of 3D object shapes *Appl. Opt.* **22** 3977–82
[6] Zhang L Z L, Curless B and Seitz S M 2002 Rapid shape acquisition using color structured light and multi-pass dynamic programming *Proc. First Int. Symp. on 3D Data Processing Visualization and Transmission* pp 24–36
[7] Pan J, Huang P S, Zhang S and Chiang F-P 2004 Color N-Ary gray code for 3D shape measurement *ICEM12-12th Int. Conf. on Experimental Mechanics*

[8] Van Der Jeught S and Dirckx J J J 2019 Deep neural networks for single shot structured light profilometry *Opt. Express* **27** 17091–101
[9] Nishizaki Y, Horisaki R, Kitaguchi K, Saito M and Tanida J 2020 Analysis of non-iterative phase retrieval based on machine learning *Opt. Rev.* **27** 136–41
[10] Yu H, Zheng D, Fu J, Zhang Y, Zuo C and Han J 2020 Deep learning-based fringe modulation-enhancing method for accurate fringe projection profilometry *Opt. Express* **28** 21692–703
[11] Qian J, Feng S, Tao T, Hu Y, Li Y, Chen Q and Zuo C 2020 Deep-learning-enabled geometric constraints and phase unwrapping for single-shot absolute 3D shape measurement *APL Photonics* **5** 046105
[12] Yu H, Chen X, Zhang Z, Zuo C, Zhang Y, Zheng D and Han J 2020 Dynamic 3D measurement based on fringe-to-fringe transformation using deep learning *Opt. Express* **28** 9405–18
[13] Zhang L, Chen Q, Zuo C and Feng S 2020 High-speed high dynamic range 3D shape measurement based on deep learning *Opt. Lasers Eng.* **134** 106245
[14] Feng S, Zuo C, Yin W, Gu G and Chen Q 2019 Micro deep learning profilometry for high-speed 3D surface imaging *Opt. Lasers Eng.* **121** 416–27
[15] Aguénounon E, Smith J T, Al-Taher M, Diana M, Intes X and Gioux S 2020 Real-time, wide-field and high-quality single snapshot imaging of optical properties with profile correction using deep learning *Biomed. Opt. Express* **11** 5701–16
[16] Machineni R C, Spoorthi G E, Vengala K S, Gorthi S and Gorthi R K S S 2020 End-to-end deep learning-based fringe projection framework for 3D profiling of objects *Comput. Vis. Image Underst.* **199** 103023
[17] Qiao G, Huang Y, Song Y, Yue H and Liu Y 2020 A single-shot phase retrieval method for phase measuring deflectometry based on deep learning *Opt. Commun.* **476** 126303
[18] Feng S, Chen Q, Gu G, Tao T, Zhang L, Hu Y, Yin W and Zuo C 2018 Fringe pattern analysis using deep learning *Adv. Photonics* **1** 1–7
[19] Qian J, Feng S, Li Y, Tao T, Han J, Chen Q and Zuo C 2020 Single-shot absolute 3D shape measurement with deep-learning-based color fringe projection profilometry *Opt. Lett.* **45** 1842–5
[20] Wang Z and Bovik A C 2009 Mean squared error: love it or leave it? A new look at signal fidelity measures *IEEE Signal Process. Mag.* **26** 98–117
[21] Ronneberger O, Fischer P and Brox T 2015 U-net: convolutional networks for biomedical image segmentation *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention* pp 234–41
[22] Liu L, Cheng J, Quan Q, Wu F X, Wang Y P and Wang J 2020 A survey on U-shaped networks in medical image segmentations *Neurocomputing* **409** 244–58
[23] Zhao H, Gallo O, Frosio I and Kautz J 2016 Loss functions for image restoration with neural networks *IEEE Trans. Comput. Imaging* **3** 47–57
[24] Lu Z and Chen Y 2019 Single image super resolution based on a modified U-net with mixed gradient loss (arXiv:1911.09428)
[25] Abadi M *et al* 2016 TensorFlow: a system for large-scale machine learning *Proc. 12th USENIX Symp. on Operating Systems Design and Implementation, OSDI 2016* pp 265–83