Transparent Surface Reconstruction Network

Friso Meeusen, Imre Atmodimedjo, and Mohamed Gamil

Eindhoven University of Technology, Department of Electrical Engineering

January 14, 2025

***Abstract*—Accurate measurements of fluid surface deformations is important for understanding rheoligical and interfacial properites in various scientific research. Traditional methods have downsides, such as being labor- and knowledge-intensive and creating waste. The puff rheometer would be a better approach, as it is performs fluid surface deformation measurments more automatically and is non-evasive. However, this device cannot be used on transparent fluids as its laser-distance sensor will not reflect on these fluid surfaces. Therefore, AI-based solutions could provide a solution. These methods could be trained to reconstruct fluid surface deformations using a reference pattern and images of the distorted pattern that is caused by light refraction at the fluid-vapor interface. This study presents the transparent surface reconstruction network, that is designed for the specific use of the puff rheometer. This network is trained on synthetic data that is obtained using a wave generating function and refraction model. Through the use of existing loss functions and the introduction of a novel B-spline loss function, the network demonstrates to effectively capture the fluid surface deformations on both synthetic and real data and is able to outperform existing models.**

***Index Terms*—Fluid Surface Deformation, Puff Rheometer, Rheological Measurement, Non-invasive Measurement, AI-based Fluid Analysis, Neural Networks**

I. Introduction

Background and Motivation

Precise measurement of rheological and interfacial fluid properties, such as viscosity, moduli, surface tension, and density, is required for research in numerous fields, including epidemiology and food and water technology [1, 2, 3, 4]. Traditional measurement methods are performed manually, which is time-, labor-, and knowledge-intensive. Additionally these methods require contact with the sample, producing substantial amounts of waste, including hygienic preventive materials [5]. Therefore, there is a need for an automated, non-evasive measurement method to determine fluid properties.

The puff rheometer is a measurement device that exerts an air pulse at a given pressure onto a fluid sample and has a laser distance sensor to measure the consequent surface deformation as can be seen in Fig. 1[3]. Based on this measurement, a depth map can be created. This map represents how much the fluid is deformed relative to the bottom of the sample holder for a discrete set of points.

When creating multiple depth maps over time, the temporal evolution of the dynamic fluid surface can be reconstructed. This evolution is compared with computational models to determine the physical fluid properties. The puff rheometer is less labor-intensive and more sustainable in obtaining these physical properties than manual measurements [5].

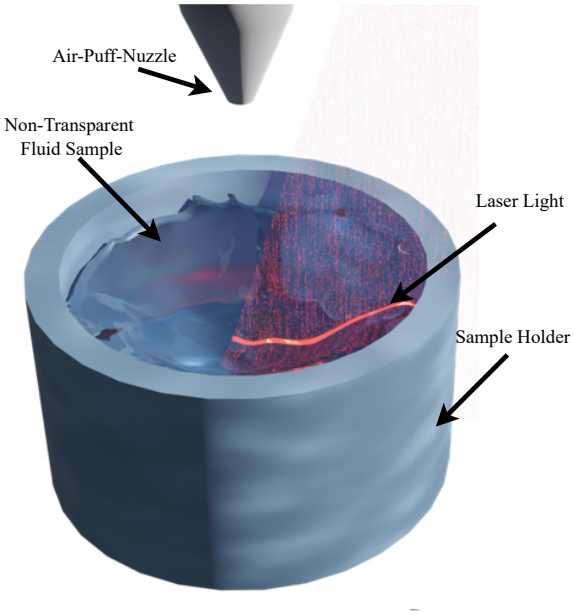

Figure 1: The puff rheometer measuring a transparent fluid sample. The laser light transmits through the water instead of being reflected, causing the measurement to be unreliable. Figure provided with thanks to C.R.C. van der Gracht.

However, this setup only works for non-transparent fluids. When using transparent fluids, most of the laser light will not be reflected, but is transmitted through the fluid surface, causing the laser distance sensor to give unreliable results. Additionally, relatively small measurement errors of the laser distance sensor will propagate to the corresponding depth map. Axisymmetry is assumed around the axis orthogonal to the resting fluid surface, because the laser measures the deformation along a line. The depth map over the entire surface domain is obtained by rotating the laser measurement around the symmetry axis. While this assumption is necessary for the creation of the 3D depth map, it can lead to less reliable results since the measurement error of the laser distance sensor will be spread out over the whole map.

AI-based methods could resolve the issues with the laser distance sensor by using a pre-determined reference pattern and camera that are placed below and above the fluid sample,

respectively, instead. After administering the air puff, the camera captures the temporal evolution of the deformations by taking a sequence of images of the distorted reference pattern, or warped images, as observed from its perspective. A trained machine learning model takes these warped images as input and returns a reconstruction of the distorted fluid surface over time. The goal of this paper is to develop such an AI model that accurately reconstructs transparent fluid deformations given a warped image and its corresponding reference pattern.

Literature Review

Thapa, Li, and Ye propose the fluid surface reconstruction convolutional neural network (FSRN-CNN) to estimate the depth and normal maps of a fluid surface via the refractive distortion image of a reference pattern [6]. The normal map specifies the unit vector that is orthogonal at every point to the depth map. To compute the normal map, the partial derivatives in the planar coordinates of the depth map are required. However, as the depth map is a discrete set of points, its partial derivatives are found using a finite difference scheme which is prone to numerical errors. FSRN-CNN is constructed as a series of convolutional layers, followed by two deconvolutional branches. One branch's output layer consists of a depth and normal map concatenated to each other, while the second branch aims to predict the absolute ranges of these maps. The two output layers are later rejoined and trained with a combined loss function. It is rather challenging to obtain a dataset of warped and reference patterns and their corresponding depths using physical devices. Instead, a combination of Navier-Stokes-derivations is used to reconstruct realistic fluid surfaces. These surfaces are then discretized to create ground truth depth and normal maps. A physics-based raytracing model simulates the refraction of light rays from the reference pattern that cross the fluid surface given these maps. This allows for creating images of the warped reference pattern observed by an orthographic camera above the fluid surface. FSRN-CNN is trained using a objective function which is a linear combination of several individual loss functions. These functions evaluate, among other things, the loss in the per-pixel difference of depth and normal maps and their absolute scales, the depth to normal consistency, and the raytracing model, with their corresponding ground truths. The results show that FSRN-CNN can estimate highly accurate depth and normal maps on both synthetic and real data and outperforms already existing models, such as DenseDepth and RefractNet [6, 7, 8].

Xiong and Heindrich also propose a framework that learns to reconstruct the temporal evolution of water flows using a video sequence of refractive distortions over time [9]. However, in contrast to the work of Thapa, Li and Ye, this framework does not require the input of the reference pattern, as their model also learns to model the underwater 3D scene geometry. Furthermore, their model is calibration free, allowing it to work in laboratory settings, as well as "in the wild". Additionally, instead of reconstructing the fluid surface as a discrete set of points, they introduce the use of uniform cubic B-splines to create a continuous depth function. Cubic B-splines are $\mathcal{C}^2$-continuous functions, which have as advantage that the normal map can be determined analytically instead of using a numerical approach, making it less prone to numerical errors. Moreover, B-splines allow for the use of a curvature penalty. This penalty prevents the curve from over fitting on the data and directs the network to produce smooth curves to model the water surface. The results show that the model performs better than the model by Thapa et al., when comparing the models under different physical conditions.

TREX Network

We introduce the transparent surface reconstruction (TREX) network which is designed specifically for the use of the puff rheometer. The basis of this netwokr is the FSRN-CNN model by Thapa et al., but with only one output branch for the full scale depth and normal map. On top of this, open-uniform cubic B-splines are fitted on both the ground truth and predictive depth map. This allows to compute the B-spline loss, that guides the network to predict depth maps that describe smooth surfaces as inspired by Xiong and Heidrich.

Fig. 2 shows the high-level architecture of the framework. TREX takes the warped image and underlaying reference pattern as input and uses a convolutional operations to return the corresponding depth map. The TREX network is trained on a synthetic dataset of warped images with the corresponding ground truth depth and normal maps and reference patterns. The warped image is generated using a combination of a wave-generating logistic function and a ray tracing model. By using backpropagation of specifically constructed loss functions, the neural network will be trained to reconstruct the depth map of a new combination of a warped image and reference pattern. The model is trained with and without the use of the B-spline loss to evaluate the performance of this function.

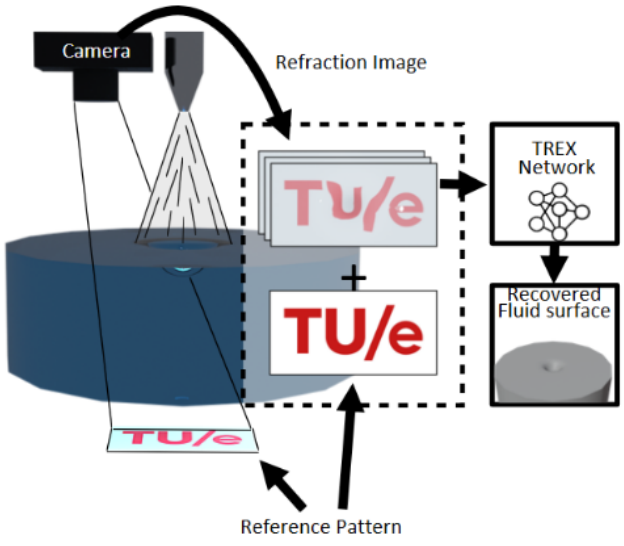

Figure 2: High-level architecture of the whole framework, including the TREX network.
Figure provided with thanks to C.R.C. van der Gracht.

II. Methodology

Data Generation

Data generation is essential for training the neural network to reconstruct the depth map from a warped image and its reference pattern. However, as it is a rather complex process to obtain a dataset of depth maps from an experimental setup, synthetic data is generated by using a physics-based raytracing model.

Synthetic data generation consists of three steps.

1) A wave generation function and its partial derivatives are discretized to obtain a depth map*, $\mathcal{D} \in \mathbb{R}^{M_x \times M_y}$, and corresponding normal map ,$\mathcal{N} \in \mathbb{R}^{M_x \times M_y \times 3}$. The values $M_x = M_y = 128$ denote the number of pixels in the x and y direction respectively. The depth map is mathematically defined as

$$\begin{aligned}\mathcal{D}^{(i,j)} = \{h(x_i, y_j) \text{ for } i \in \{n_x \in \mathbb{Z} | 1 \leq n_x \leq M_x\},\\ j \in \{n_y \in \mathbb{Z} | 1 \leq n \leq M_y\},\\ x_i, y_j \in [-w, w]\},\end{aligned} \tag{1}$$

where $h(x, y) : \mathbb{R}^2 \to \mathbb{R}$ is a continuous function of the height of the fluid relative to the bottom of the sample holder. Furthermore, x_i and y_j are the x and y coordinates of the evaluated pixel with indices (i, j), respectively, and the width of the domain is specified by w.
The normal map is described as

$$\mathcal{N}^{(i,j)} = \bar{\mathbf{n}}^{(i,j)}, \tag{2}$$

where $\bar{\mathbf{n}}$ is a normalized plane normal vector. The normal vector is defined as

$$\mathbf{n}^{(i,j)} = \left[-\frac{\partial h}{\partial x}\Big|_{x_i, y_j}, -\frac{\partial h}{\partial y}\Big|_{x_i, y_j}, 1 \right]^\top \tag{3}$$

and is normalized to get $\bar{\mathbf{n}}^{(i,j)}$.

2) By using the depth and normal maps, a warp map ($\mathcal{W} \in \mathbb{R}^{M_x \times M_y \times 2}$) is obtained. This warp map is created using a raytracing model and defines a set of vectors that specifies by what value each pixel should be translated to get the warped image.

3) The warp map is applied to the reference pattern, which creates the warped image.

Depth Map Generation: The synthetic data is created for the specific use of the puff rheometer, as the depth maps are obtained by discretizing the height function $h(x, y) : \mathbb{R}^2 \to \mathbb{R}$, We use a modified logistic function [10], such that it represent a dimple in the middle of a flat surface and is defined as

$$h(x, y, t) = f_1(t) + \frac{f_2(t) - f_1(t)}{1 + \exp\left[g_1(t) - g_2(t)\sqrt{x^2 + y^2}/r_0\right]}, \tag{4}$$

where $f_1(t)$ and $f_2(t)$ are the exponential functions

$$f_k(t) = \alpha_0^{(k)} + \alpha_1^{(k)} e^{-(t/\alpha_2^{(k)})} \tag{5}$$

and $g_1(t)$ and $g_2(t)$ are second order polynomials

$$g_k(t) = \beta_2^{(k)} t^2 + \beta_1^{(k)} t + \beta_0^{(k)}, \tag{6}$$

where α, β and r_0 are coefficients that describe the temporal and spacial evolution of the wave. Eq. (4) is fitted on experimental data to match the temporal evolution of the surface deformation, such that the coefficients α and β can be found. The experimental data is obtained by measuring the depth maps of silicon oil after the air puff is applied to its surface. A pigment is added to the silicon oil, such that a laser can be used to measure the depth. Using the analytical expression of the depth function, the normal vectors can be calculated with the use of Eq. (3). To obtain more data, the depth map of Eq. (4) is augmented, such that the spatial properties of the deformation change. These changes in- or decrease the depth and width of the dimple. The depth and width can be changed by altering the parameters $\alpha_2^{(1)}$ and r_0, respectively. Using six different values for $\alpha_2^{(1)}$, varying between -0.003 and -0.008, and nine different values for w, varying between 0.5 and 3, results in 54 distinctive initial waves. The initial waves are simulated for 10 seconds with 114 timesteps, giving a total of 6156 depth, normal and corresponding warp maps. Fig. 3 shows a set of three warped images with the corresponding reference patterns and parameters for Eqs. (4) to (6)

Raytracing Model: The raytracing model simulates the behavior of light at the fluid-vapor interface. The model traces the per-pixel incident light rays to the camera, back to the reference pattern. It is assumed that the camera is orthographic, implying that the light rays incident to the camera are all parallel to the camera axis which simplifies the overall model. This assumption is reasonable if the camera has a relatively high focal length and is positioned sufficiently far from the fluid sample [11]. The inputs of the raytracing model are the unit vectors of the light rays that are observed by the camera ($\bar{\mathbf{s}}_\mathbf{1}$), which are defined per-pixel. The vector $\bar{\mathbf{s}}_\mathbf{1} \in \mathbb{R}^3$ only depends on the angle of incidence of the camera, with respect to the resting fluid surface. Using the input, the model computes the incident light rays ($\bar{\mathbf{s}}_\mathbf{2}$) on the fluid surface and traces it back to its origin on the reference pattern. Since the location of each pixel in the reference pattern and in the camera is known, the warp map can be constructed.

This warp map is used to obtain the distorted image from the reference pattern as can be seen in Fig. 4. However, the

*Notation: depth, normal and warp maps are represented in capital calligraphic letters. The predicted maps are assigned with a hat symbol to distinguish them from the ground truths.

Vectors and matrices are denoted in bold letters and vectors of unit length are assigned with an overbar. The term $\|\mathbf{v}\|_n$ denotes the n-th norm of the vector $\mathbf{v}$.

Scalar functions and values and the difference between the predictive and ground truth maps are denoted with denoted in small italic letters.

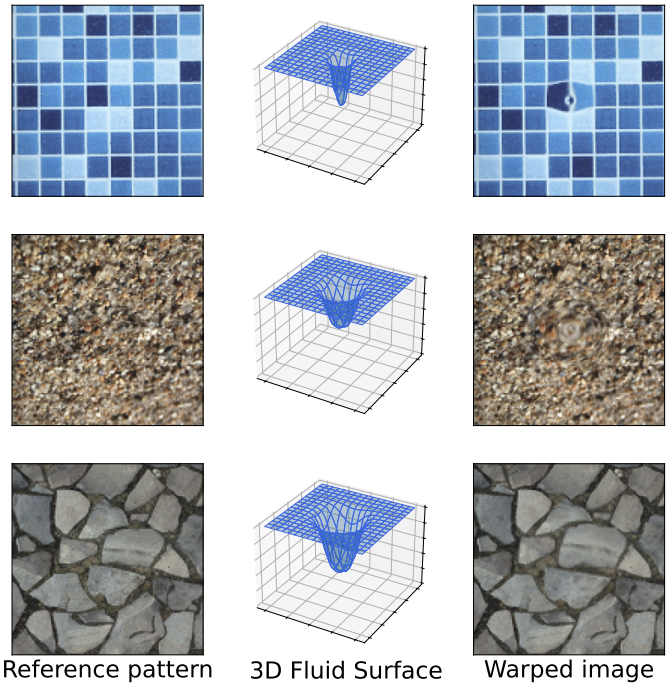

Figure 3: Three sets of examples of a refraction image with the corresponding depth and reference image. The depth maps that are created use, from top to bottom, $r_0 = 1.125$ and $\alpha_2^{(1)} = -0.006$, $r_0 = 2.0625$ and $\alpha_2^{(1)} = -0.005$, and $r_0 = 2.6875$ and $\alpha_2^{(1)} = -0.003$.

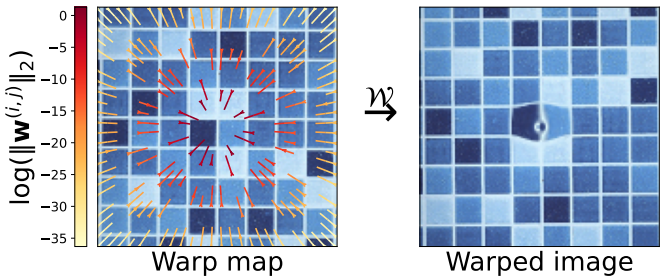

Figure 4: The warp map for some pixels, displayed on top of the reference image. If the warp map is applied to the reference pattern, the warped image is the result.

warp map shifts the original pixels not strictly to another pixel value. Therefore, the values of the warped pixels are linearly interpolated with the `scipy.interpolate.griddata` function, to obtain the values at the pixels in the warped image [12].

The raytracing model simulates only the refractive interactions at the surface of the fluid, which is governed by Snell's law. In Fig. 5, the principle of the raytracing model is visualized. The raytracing model calculates $\bar{\mathbf{s}}_2$ from $\bar{\mathbf{s}}_1$, by applying Snell's law in vector form [13]:

$$\mathbf{s_2} = \frac{n_1}{n_2}\bar{\mathbf{s}}_\mathbf{1} + \left[\frac{n_1}{n_2}\cos(\theta_1) - \cos(\theta_2)\right]\bar{\mathbf{n}}, \tag{7}$$

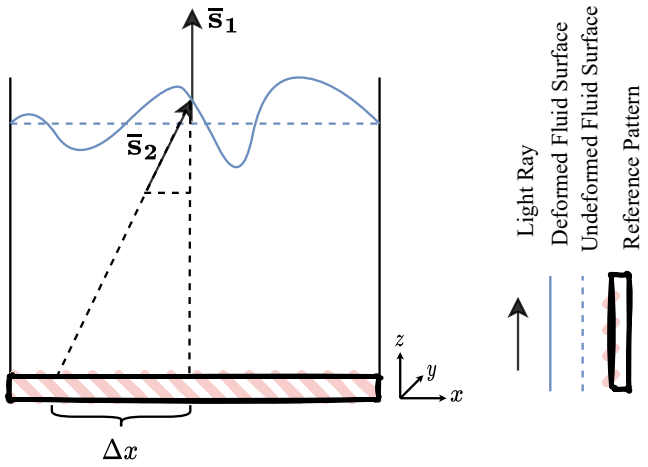

Figure 5: Schematic of the setup and overview of the raytracing model. The camera is positioned vertically above a vessel containing an arbitrary fluid, with a zero angle of incidence.

where

$$\cos(\theta_1) = -\bar{\mathbf{n}} \cdot \bar{\mathbf{s}}_\mathbf{1}, \tag{8}$$

and

$$\cos(\theta_2) = \sqrt{1 - (n_1/n_2)^2(1 - (\cos\theta_1)^2)}. \tag{9}$$

Note that $\mathbf{s_2} \in \mathbb{R}^3$ is normalized to obtain $\bar{\mathbf{s}}_\mathbf{2}$. Here n_1 and n_2 are the refractive indices of the incident and transmitted material, respectively, and in our case $n_1 = 1.00$ and $n_2 = 1.44$ are set to the refractive index of air and water, respectively [14, 15]. The incident and refractive angle are denoted by θ_1 and θ_2 respectively. However, it is not necessary to know θ_1 and θ_2 explicitly due to Eqs. (8) and (9). Using the per-pixel calculated vector $\bar{\mathbf{s}}_\mathbf{2}$ and the height of the fluid surface $h(x_i, y_j)$, the x and y components of the warp map, denoted by $\Delta x^{(i,j)}$ and $\Delta y^{(i,j)}$ respectively, can be found by scaling with respect to $h^{(i,j)}$. For a single pixel, these translations are mathematically formulated as

$$\Delta x = \frac{h}{s_{2,z}} s_{2,x} \tag{10}$$

and

$$\Delta y = \frac{h}{s_{2,z}} s_{2,y}, \tag{11}$$

where $s_{2,x}$, $s_{2,y}$ and $s_{2,z}$ denote the x, y and z component of $\bar{\mathbf{s}}_2$, respectively. Eqs. (10) and (11) are used to create $\mathcal{W}$ by defining its per-pixel elements as

$$\mathbf{w}^{(i,j)} = \begin{pmatrix} \Delta x^{(i,j)} \\ \Delta y^{(i,j)} \end{pmatrix}. \tag{12}$$

Network Architecture

Model Overview: In this study, a neural network specialised for transparent fluid surface reconstruction, TREX, is proposed. Similar to the FSRN-CNN model, TREX uses a deconvolutional-convolutional layers structure and a combined

loss function for training [6]. However, for simplicity, the model has a single output branch which consists of a concatenated depth and normal map. In contrast to FSRN-CNN, the scale-loss function (see Eq. (18)) of TREX dynamically receives the required input. More specifically, the ground truth ranges are obtained within the loss function definition instead of having explicitly fed as an output tensor. This allows for a more flexible and organized software framework without involving relatively large output dimensions and numerous tensor-slicing operations. The high level architecture of the TREX network and its comparison with FSRN-CNN are shown in Figs. 6 and 7 respectively

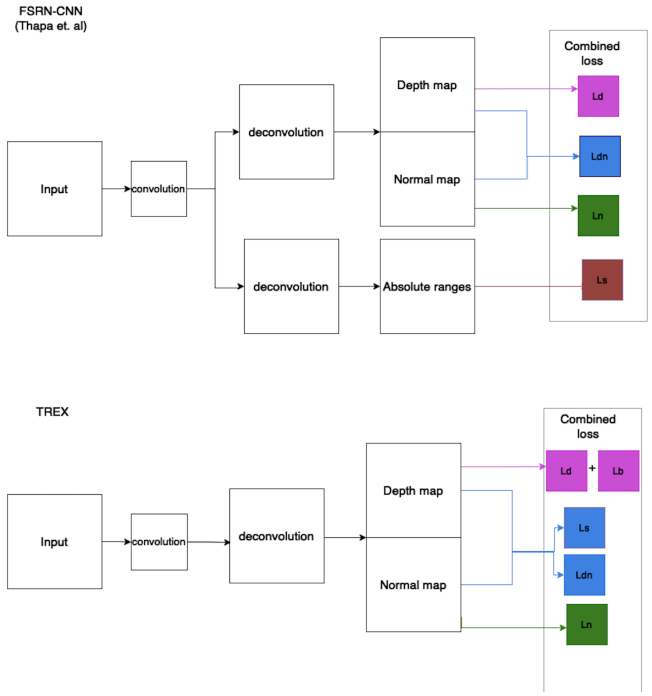

Figure 6: TREX vs FSRN-CNN network architecture. For a more detailed model summary, please refer to Sec. A, Tab. 2.

Loss Functions: The total loss function to train the TREX network is defined as:

$$\mathcal{L}_{\text{tot}} = \alpha_1 \mathcal{L}_{\text{d}} + \alpha_2 \mathcal{L}_{\text{n}} + \alpha_3 \mathcal{L}_{\text{dn}} + \alpha_4 \mathcal{L}_{\text{s}} \left(+\alpha_5 \mathcal{L}_b\right). \quad (13)$$

As can be seen, this loss function is a linear combination of the depth, normal, depth-to-normal, scale loss, which are denoted by $\mathcal{L}_{\text{d}}$, $\mathcal{L}_{\text{n}}$, $\mathcal{L}_{\text{dn}}$, and $\mathcal{L}_{\text{s}}$ respectively. These functions are derived from other studies on depth prediction and surface reconstruction [6, 16]. However, in this study, the B-spline loss function - $\mathcal{L}_{\text{b}}$ - is proposed and its effect is experimented (hence why it is in parentheses ; see Sec. III). The scalars $\alpha_1, \cdots, \alpha_5$ denote the weighting factors of every loss function in the total loss.

Depth loss. The depth loss compares the ground truth and predicted depth map and tries to minimize the error between these two maps. Let the predicted depth map be defined as $\hat{\mathcal{D}}$ and the per-pixel difference between the ground truth and predicted depth map be defined as $d_{i,j} = \mathcal{D}^{(i,j)} - \hat{\mathcal{D}}^{(i,j)}$. Formally, the depth loss is defined as

$$\begin{aligned}\mathcal{L}_{\text{d}} = &\frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} d_{i,j}^2 \\ &- \frac{1}{2(M_x M_y)^2} \left(\sum_{i=1}^{M_x} \sum_{j=1}^{M_y} d_{i,j} \right)^2 \\ &+ \frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \|\nabla d_{i,j}\|_2^2 ,\end{aligned} \quad (14)$$

where $\nabla d_{i,j}$ is the image gradient of the difference at pixel (i, j). The first and second term is called the scale-invariant mean squared error, which penalizes inconsistency [17]. The third term is a first-order matching term that ensures that local structures of the prediction and ground truth will be similar [18].

Normal loss. The normal loss minimizes the error between the ground truth normal map and the predicted normal map. The per-pixel ground truth and predicted normal vectors of a normal map are denoted as $\mathcal{N}^{(i,j)}$ and $\hat{\mathcal{N}}^{(i,j)}$, respectively and the difference vector is formulated as $\mathbf{n}_{i,j} = \mathcal{N}^{(i,j)} - \hat{\mathcal{N}}^{(i,j)}$. The normal loss is defined as

$$\begin{aligned}\mathcal{L}_{\text{n}} = &\frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \|\mathbf{n}_{i,j}\|_2^2 \\ &- \frac{1}{2(M_x M_y)^2} \left\| \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \mathbf{n}_{i,j} \right\|_2^2 .\end{aligned} \quad (15)$$

Depth-normal loss. The depth-normal loss evaluates the loss of the normal map $\mathcal{N}_{\mathcal{D}}$, which is calculated from the depth map. The normal vectors cannot be calculated with Eq. (3), because the depth map is not a continuous function. Therefore, the derivative of the depth map is taking with the use of a first order forward finite difference scheme [19]. However, if there is already a low level of noise present in the depth map, the derivatives can adapt to relatively large values that are not representative. Therefore, a Laplacian filter is applied to smooth the depth map before taking the derivative [20]. The difference between the predicted and ground truth per-pixel calculated normal map $\mathbf{n}'_{i,j}$ is defined as

$$\mathbf{n}'_{i,j} = \frac{\mathcal{N}_{\mathcal{D}}^{(i,j)}}{\|\mathcal{N}_{\mathcal{D}}^{(i,j)}\|_2} - \frac{\hat{\mathcal{N}}_{\mathcal{D}}^{(i,j)}}{\|\hat{\mathcal{N}}_{\mathcal{D}}^{(i,j)}\|_2}. \quad (16)$$

Here both calculated normal vectors are normalized. The depth-normal loss is then defined as

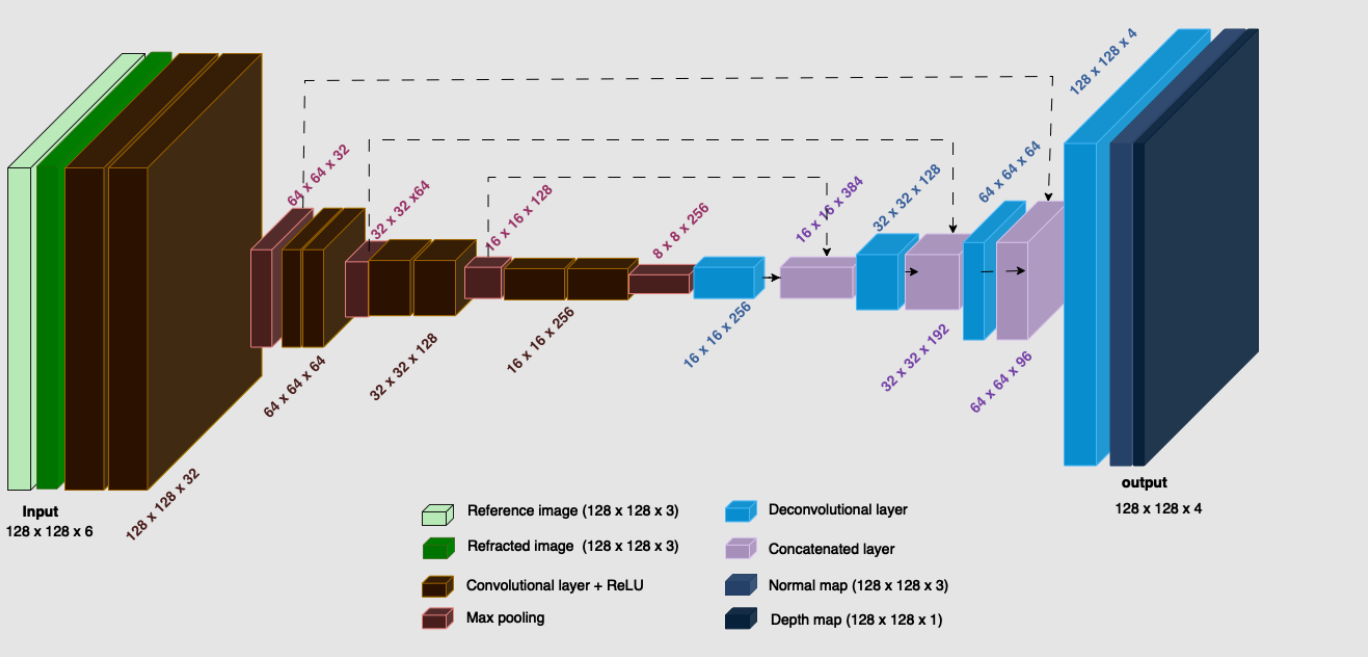

Figure 7: Architecture of the TREX Network.

$$\mathcal{L}_{\text{dn}} = \frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \|\mathbf{n}'_{i,j}\|_2^2 - \frac{1}{2(M_x M_y)^2} \left\| \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \mathbf{n}'_{i,j} \right\|_2^2 . \tag{17}$$

Scale loss. The absolute range of an arbitrary map $\mathcal{M}$ is defined as $\mathbf{R}_{\mathcal{M}} = [\min(\mathcal{M}), \max(\mathcal{M})]^\top$. Accordingly, the absolute ranges of the ground truth and predictive depth and normal maps are $\mathbf{R}_{\mathcal{D}}$, $\mathbf{R}_{\mathcal{N}}$, $\mathbf{R}_{\hat{\mathcal{D}}}$, and $\mathbf{R}_{\hat{\mathcal{N}}}$, respectively. The difference vector for the depth and normal map are then defined as $\mathbf{r}_{\mathcal{D}} = \mathbf{R}_{\mathcal{D}} - \mathbf{R}_{\hat{\mathcal{D}}}$ and $\mathbf{r}_{\mathcal{N}} = \mathbf{R}_{\mathcal{N}} - \mathbf{R}_{\hat{\mathcal{N}}}$, respectively. The scale loss is the sum of the L_2-norm of the difference vectors and is defined as

$$\mathcal{L}_s = \frac{1}{4} \left(\|\mathbf{r}_{\mathcal{D}}\|_2^2 + \|\mathbf{r}_{\mathcal{N}}\|_2^2 \right). \tag{18}$$

B-spline loss. Bivariate B-splines ($B(x, y) : \mathbb{R}^2 \to \mathbb{R}$) are fitted on the depth maps to compute a B-spline loss, guiding the neural network to produce outputs that describe a smooth wave. This type of spline that is fitted on a set of data points is formulated as

$$B(x, y) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} C_{i,j} S_{i,p}(x) S_{j,q}(y). \tag{19}$$

Here $C_{i,j}$ denotes one of the $n_x \times n_y$ B-spline coefficients. $S_{i,p}(x)$ and $S_{j,p}(y)$ are the spline basis functions of degree p and q respectively, which are defined by the Cox-de-Boor recursion equation [21]. The integers i and j denote the indices of the elements from the knot vectors $\mathbf{T}_x = (t_{x_0}, t_{x_1}, \cdots, t_{x_{m_x}})$ and $\mathbf{T}_y = (t_{y_0}, t_{y_1}, \cdots, t_{y_{m_y}})$. The relation between m_x, n_x, and p and m_y, n_y and q are given as

$$m_x = n_x + p + 1 \tag{20}$$

and

$$m_y = n_y + q + 1 \tag{21}$$

respectively. In essence, B-splines are parameterized by the knot vector, the set of coefficients, and their degrees. To reduce the set of possible B-splines that can be fitted, open-uniform cubic B-splines are used, which impose the following assumptions [9, 21, 22]:

1) Cubic: $p = q = k = 3$. This type of B-spline can adapt to a rather wide range of forms. Using higher degrees gives more flexibility of the curve but is also more prone to over-fitting.

2) Open: The first $k - 1$ knots in both knot vectors are set to be equal to the k-th knot, which in turn is equal to the lower bounds of the domain in both directions. Similarly, the last $k - 1$ knots in both knot vectors are equal to the last k-th knot, which is equal to the upper bounds of the domain in both directions. This assumption ensures that the B-spline is defined over the whole domain of the depth map.

3) Uniformity: The knots in both $\mathbf{T_x}$ and $\mathbf{T_y}$ are placed equidistantly from each other, except for the first and last k knots as discussed in assumption 2.

Through the cubic assumptions, p and q are now known. Setting the number of coefficients to a constant ($n_x =$

$n_y = 20$), automatically determines $\mathbf{T_x}$ and $\mathbf{T_y}$ when applying Eqs. (20) and (21) and the open-uniformity assumption. The parameter fitting now boils down to finding the optimal values of the coefficients. To find these values, the `scipy.Interpolate.bsplrep` function is used to find a B-spline that minimizes the total squared error between $B(x, y)$ and the ground truth data [23, 24].

The B-spline loss is defined as

$$\mathcal{L}_{\mathrm{b}} = \frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \left[B_{\mathcal{D}}(x_i, y_j) - B_{\hat{\mathcal{D}}}(x_i, y_j) \right]^2, \quad (22)$$

where $B(x, y)$ and $\hat{B}(x, y)$ are the cubic-open-uniform B-splines that are fitted on the ground truth and predictive depth maps respectively. The analytical difference between these two B-splines across the entire domain could theoretically be computed through integration. However, due to the substantial computational expense of this method, the differences are instead evaluated at a discrete set of coordinates (x_i, y_j).

III. Results and Discussion

Two experiments were run on the synthetic dataset described earlier (see Sec. II). On the first set of experiments, the TREX network with original loss functions inspired by Thapa et al were used, while in the second set, the B-spline loss function (to be referred to as TREX-spline) was included. In both cases, all loss functions contributed equally to the total loss function ($\alpha_1 = \alpha_2 = \cdots = \alpha_5 = 0.2$). Additionally, the FSRN-CNN was also attempted. However, due to the high complexity of the model as well as the lack of supplementary code and preprocessing documentation, it is unknown whether the implemented software fits the intended replication of the model. Thus, the results on the test dataset of FSRN-CNN are not completely reliable. Lastly, the TREX-spline network was also fitted on real data obtained by the puff rheometer to see the deployability of the model.

Experimental Setup

The training and test data was split with respect to the sequences. Splitting the data randomly could lead to data contamination, as two samples from the same sequence could look relatively similar. Therefore, 45 sequences (5130 samples/frames) were allocated to the training dataset, while 9 sequences (1026 samples/frames) went to the test dataset. The model was trained for 10 epochs with the batch size set to 32 and an Adam optimizer was used [25]. To monitor the model performance, a set of error and accuracy metrics were applied in the training and testing process. These metrics include the root mean squared error (RMSE), absolute relative error (ARE), and three threshold accuracies ($\rho < 1.25, \rho < 1.25^2, \rho < 1.25^3$) [6, 26].

Threshold Accuracy Metric: The treshold accuracy metric calculates the proportion of pixels where the ratio of the predicted depth to the true depth (and vice versa) is less than the specified threshold (ρ)

$$\frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \left[\max \left(\frac{\hat{\mathcal{D}}^{(i,j)}}{\mathcal{D}^{(i,j)}}, \frac{\mathcal{D}^{(i,j)}}{\hat{\mathcal{D}}^{(i,j)}} \right) < \rho \right] \quad (23)$$

Root Mean Squared Error: The RMSE is a standard metric for measuring the average magnitude of the error. It is the square root of the average of squared differences between predicted and true values:

$$\mathrm{RMSE} = \sqrt{\frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \left(\mathcal{D}^{(i,j)} - \hat{\mathcal{D}}^{(i,j)} \right)^2} \quad (24)$$

Absolute Relative Error: The ARE measures the mean of absolute differences between the predicted and true values, normalized by the true values:

$$\mathrm{ARE} = \frac{1}{M_x M_y} \sum_{i=1}^{M_x} \sum_{j=1}^{M_y} \frac{|\mathcal{D}^{(i,j)} - \hat{\mathcal{D}}^{(i,j)}|}{\mathcal{D}^{(i,j)} + \epsilon} \quad (25)$$

where ϵ is a small constant added for numerical stability to avoid division by zero.

Synthetic Data Experiments

Tab. 1 presents the test results for each model according to the aforementioned data split. TREX and TREX-spline outperform FSRN-CNN across all metrics. The relatively low scores and accuracy of the FSRN-CNN can be primarily explained by a consistent and substantial shift in the predicted scale, as illustrated in Fig. 8. This shows that our implementation of TREX and TREX-spline surpass the performance of FSRN-CNN. Additionally, the spline implementation further enhances the performance of the network, because both the RMSE and ARE metric are improved, even as the threshold accuracy.

Table 1: Quantitative comparison of the depth estimation between FSRN-CNN and the TREX and TREX-spline networks.

Model	**RMSE**	**ARE**	$\rho < 1.25, \rho < 1.25^2, \rho < 1.25^3$
FSRN-CNN	0.471	0.498	0.009 , 0.023 , 0.139
TREX	0.066	0.048	0.957, 0.988, 0.995
TREX-spline	0.031	0.021	0.984, 0.995, 0.998

However, only evaluating the metrics does not give a complete picture of the performance of the network. To gain a deeper understanding, a prediction on the test set is visualized for the FSRN-CNN, as shown in Fig. 8. While the FSRN-CNN appears to capture the shape of the deformation, the predicted depth ranges do not align with the ground truth, leading to reduces scores and accuracy.

The prediction on the test set for TREX and TREX-spline are shown in Figs. 9 and 10, respectively. Both TREX and TREX-spline successfully capture the correct shape and depth range of the deformation, representing a significant improvement over FSRN-CNN. Accurately predicting the minimum

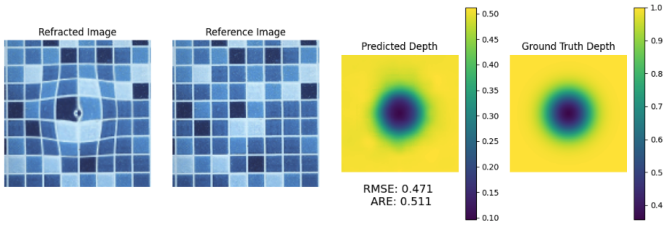

Figure 8: Prediction results on the test set for the corresponding warped image, reference pattern and ground truth depth map by the FSRN-CNN.

and maximum height of the deformation is usefull for the puff rheometer, as these properties define the characteristics of the deformation. However, the predictions of the TREX network contain some artifacts. These artifacts are pixels that do not match their surrounding pixels and make the prediction less smooth. Since fluid deformations are naturally smooth and continuous, these artifacts should be mitigated in the predicted depth maps.

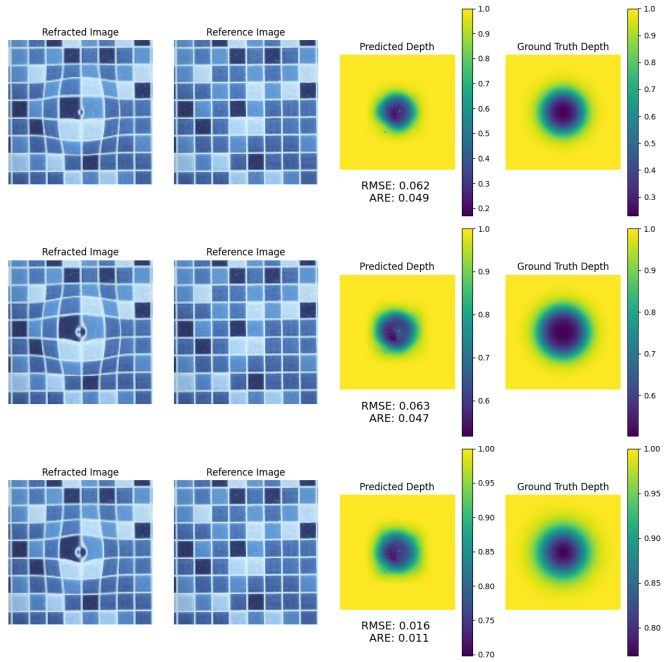

Figure 9: Prediction results on the test set for the corresponding warped images, reference pattern and ground truth depth map by the TREX network.

The TREX-spline network improves upon the TREX network, because the artifacts that can be seen in Fig. 9 are not present anymore in Fig. 10. The implementation of the B-spline loss allows the network to learn that the deformation should be smooth.

Fig. 11 displays the train and validation loss of the TREX-spline network, which illustrates the learning progress on the synthetic data. The training loss is lower than the validation loss, which could suggest that the model is overfitting. However, since the validation loss plateaus rather than increases, similar to the training loss, this is considered acceptable.

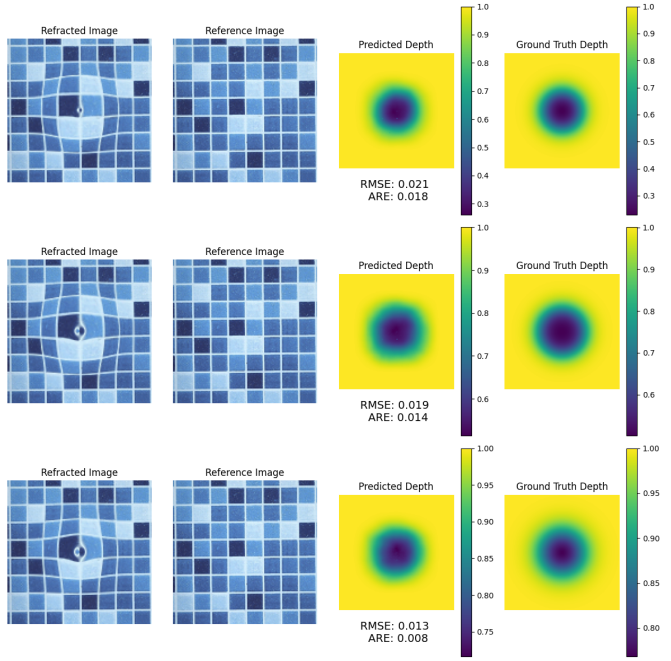

Figure 10: Prediction results on the test set for the corresponding warped images, reference pattern and ground truth depth map by the TREX-spline network.

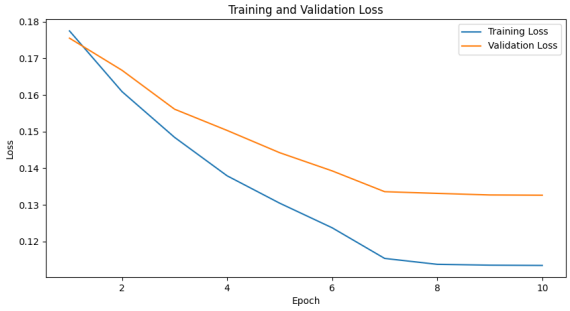

Figure 11: The training and validation loss curve of the TREX-spline network.

Real Data Experiments

To further evaluate the trained TREX-spline model, it was tested on real data.The experimental data is obtained by creating the setup from Fig. 2. The fluid vessel is filled with silicon oil, which is a transparent fluid. The fluid undergoes deformations by applying air pressure on the surface. These pressures range between 60 and 220 mbar. A Raspberry Pi is used to take pictures of the deformed surface, the refraction images, which are used as input of the model. Although the model is trained with a top-down point of view on the synthetic data, the camera in the setup is placed at an angle. In addition, the reference pattern is not the true reference pattern, as it is

also an image taken by the camera, making it slightly warped. Furthermore, the model was not trained on this reference pattern and the effect of testing on unseen reference patterns was not explored in this study. Nevertheless, TREX-spline was able to capture the deformations with respect to the camera angle and magnitude of deformations as can be inferred from Fig. 12).

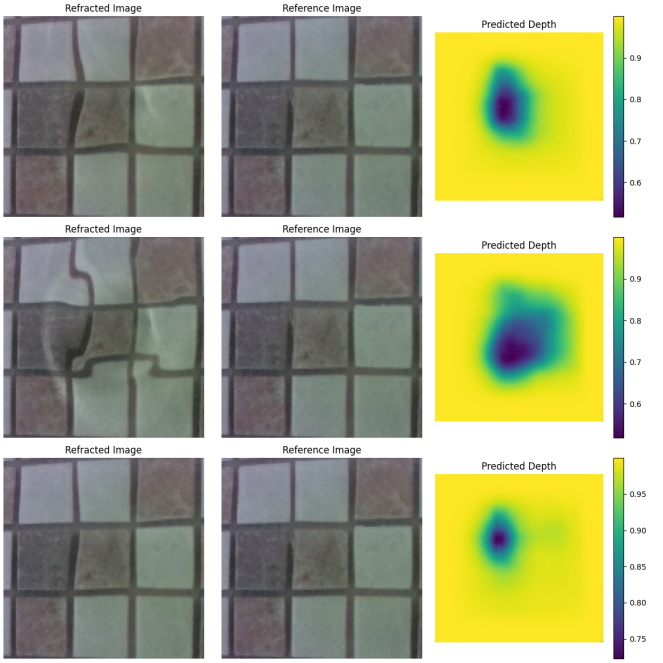

Figure 12: Results on real data using the TREX-spline network.

IV. Conclusion and Future Work

This study researched the concept of transparent fluid surface deformation reconstruction for the specific use of the puff rheometer using refraction images and their corresponding refraction pattern as input. To achieve this, the neural network architecture TREX was proposed. This network was trained on synthetic data that was obtained using a wave generating function and a raytraycing model. Multiple loss functions were studied and applied in the training process as well as the introduction of the novel B-Spline loss function. The results show that TREX performs better to an existing similar model when comparing it on multiple accuracy and error metrics. Especially the B-spline loss function showed a substantial increase in the network performance. Finally, the TREX model demonstrates the capability to reconstruct accurate depth maps from images obtained during real experiments with the puff rheometer.

Further steps could be taken to deepen the research and understanding of the problem, as well as improving the performance of the surface reconstruction model. For instance, the effect of removing the normal map from the output layer as well as the corresponding loss functions should be explored, given that this map can already be obtained from a direct relation with the depth map. In addition, the weights of the loss functions in the total loss can be fine tuned. As for the B-Spline loss function, the first step would be to optimize the number of coefficients and knots. Furthermore, the possibility of a network that reconstructs the deformation using B-splines could be investigated. The goal of this network would be to return the parameter values of the B-splines that describe the fluid deformation most accurately without over fitting on the data. In this case, the network is not limited of the cubic-open-uniform assumption and is free to use other type of B-splines as well. A rather useful property of this approach would be that the modeled deformation is a continuous function from which the normal map could be derived analytically. Lastly, training on real data, or configurations of the physical setup that relate more to the synthetic data could be experimented to understand the deployability of this framework as a whole.

Code Availability

The code repositories for the data generation and TREX network are publicly available at: https://github.com/IDTP-track5-surfaces respectively. Instructions for setting up, using, and contributing to the code are included in the accompanying documentation.

References

[1] P. Katre, S. Banerjee, S. Balusamy, and K. C. Sahu, "Fluid dynamics of respiratory droplets in the context of covid-19: Airborne and surfaceborne transmissions," *Physics of Fluids*, vol. 33, no. 8, 2021.

[2] S. Hanegraaf, "Potential of puff rheometer for complex fluids," Eindhoven, The Netherlands, June 2023, available at https://research.tue.nl/en/studentTheses/potential-of-puff-rheometer-for-complex-fluids.

[3] "An apparatus arranged for measuring a rheological property and a method for measuring a rheological property," Jun 2022. [Online]. Available: https://patents.google.com/patent/NL2032223B1/en

[4] S. Morren, T. Van Dyck, F. Mathijs, S. Luca, R. Cardinaels, P. Moldenaers, B. De Ketelaere, and J. Claes, "Applicability of the foodtexture puff device for rheological characterization of viscous food products," *Journal of Texture Studies*, vol. 46, no. 2, pp. 94–104, 2015.

[5] C. W. Macosko, *Rheology: principles, measurements, and applications*. New York: VCH, 1994.

[6] S. Thapa, N. Li, and J. Ye, "Dynamic fluid surface reconstruction using deep neural network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[7] J. Stets, Z. Li, J. R. Frisvad, and M. Chandraker, "Single-shot analysis of refractive shape using convolutional neural networks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 995–1003.

[8] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.
[9] J. Xiong and W. Heidrich, "In-the-wild single camera 3d reconstruction through moving water surfaces," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 558–12 567.
[10] L. J. Reed and J. Berkson, "The application of the logistic function to experimental data." *The Journal of Physical Chemistry*, vol. 33, no. 5, pp. 760–779, 2002.
[11] G. Avern, "The orthographic approximation-a simple geometrical model for avoiding perspective error in constructing photomosaics," 2004.
[12] "scipy.interpolate.girddata, — scipy v1.13.1 manual," Jun 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html
[13] Z. Zhu, K. Zhu, W. Gao, and H. Zhou, "2012 problem 5: Bright waves," 09 2015.
[14] B. Edlén, "The refractive index of air," *Metrologia*, vol. 2, no. 2, p. 71, 1966.
[15] I. Thormählen, J. Straub, and U. Grigull, "Refractive index of water and its dependence on wavelength, temperature, and density," *Journal of physical and chemical reference data*, vol. 14, no. 4, pp. 933–945, 1985.
[16] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2650–2658.
[17] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," 2014.
[18] D. Basin and T. Walsh, "Difference matching," in *International Conference on Automated Deduction*. Springer, 1992, pp. 295–309.
[19] E. W. Weisstein, "Finite difference," *https://mathworld. wolfram. com/*, 2005.
[20] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand, "Fast local laplacian filters: Theory and applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 5, pp. 1–14, 2014.
[21] S. Biswas and B. C. Lovell, "B-splines and its applications," in *Bézier and Splines in Image Processing and Machine Vision*. Springer, 2008, pp. 109–131.
[22] M. Munguia and D. Bhatta, "Use of cubic b-spline in approximating solutions of boundary value problems," *Applications and Applied Mathematics: An International Journal (AAM)*, vol. 10, no. 2, p. 7, 2015.
[23] "scipy.interpolate.bisplrep — scipy v1.13.1 manual," Jun 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.bisplrep.html#scipy.interpolate.bisplrep
[24] "Smoothing splines — scipy v1.13.1 manual," Jun 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/tutorial/interpolate/smoothing_splines.html#tutorial-interpolate-2d-spline
[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[26] C. Cadena, Y. Latif, and I. D. Reid, "Measuring the performance of single image depth estimation methods," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4150–4157.

APPENDIX A
NETWORK ARCHITECTURE

Table 2: Detailed TREX network architecture.

Type	**Description / Filters**	Output Size
Input	reference + refracted image	128 x 128 x 6
Conv_1	32 filters ,kernel 3 x 3, ReLu	128 x 128 x 32
Conv_2	32 filters ,kernel 3 x 3, ReLu	128 x 128 x 32
MaxPool_f1	size 2 x 2	64 x 64 x 32
Conv_3	64 filters ,kernel 3 x 3, ReLu	64 x 64 x 64
Conv_4	64 filters ,kernel 3 x 3, ReLu	64 x 64 x 64
MaxPool_f2	size 2 x 2	32 x 32 x 64
Conv_5	128 filters ,kernel 3 x 3, ReLu	32 x 32 x 128
Conv_6	128 filters ,kernel 3 x 3, ReLu	32 x 32 x 128
MaxPool_f3	size 2 x 2	16 x 16 x 128
Conv_7	256 filters ,kernel 3 x 3, ReLu	16 x 16 x 256
Conv_8	256 filters ,kernel 3 x 3, ReLu	16x 16 x 256
MaxPool_f4	size 2 x 2	8 x 8 x 256
conv_transpose_1	256 filters ,kernel 4 x 4 ,strides = 2	16 x 16 x 256
concatenated_layer_1	MaxPool_f3 - conv_transpose_1	16 x 16 x 384
conv_transpose_2	128 filters ,kernel 4 x 4, strides =2	32 x 32 x 128
concatenated_layer_2	MaxPool_f2 - conv_transpose_2	32 x 32 x 192
conv_transpose_3	64 filters ,kernel 4 x 4 ,strides = 2	64 x 64 x 64
concatenated_layer_3	MaxPool_f1 - conv_transpose_3	64 x 64 x 96
conv_transpose_4	4 filters ,kernel 4 x 4 ,strides = 2	128 x 128 x 4
output_layer	Sigmoid activation	128 x 128 x 4

APPENDIX B
INDIVIDUAL CONTRIBUTIONS

Friso Meeusen

During the project, I had multiple tasks that I will briefly report on.
I read the papers by Thapa et al. and Xiong and Heindrich, on which the main part of our research is based, in detail. I gave a presentation and wrote an extensive summary of these papers, such that my team members could understand it as well and implement it in their work.
I was responsible for communication with our supervisors and making clear to my team members when and where we had meetings as well as our deadlines. Furthermore, I made the notes during the meetings and made sure that they were available for everyone.
From the start of Q4 onwards, I did all the research necessary to implement B-splines into the TREX network. I wrote down in extensive detail the mathematical background of these functions and how we can use them.
For the report, I took full charge of the abstract, introduction, and the B-spline loss function. Some parts of the data generation and the conclusion and future work sections are also part of my work. Furthermore, I also made a strong contribution to the whole formatting of the paper and the mathematical formulation. Lastly, I helped my other team members by going over their sections and giving feedback to make sure that the paper was a coherent story and met academic standards.

Imre Atmodimedjo

Throughout the duration of the project, my primary focus has been on data generation. I have contributed to the project by creating a raytracing model that was able to create warped images according to a deformed fluid surface.
In the initial phase of the project, I worked towards understanding the current implementation of a raytracing model and recreating the raytracing model in Python. Afterwards, I have explored the raytracing model for refraction, reflection, and tilt angles. The raytracing model was now ready to be used to create data for training our network.
In the second phase of the project, I have finished working on the data generation and switched my focus towards obtaining experimental data and helping my teammates at places where needed. I collected the experimental data, such that we could evaluate the model on real data instead of only on synthetic data.
Overall, my contributions in the project were focused on data and especially synthetic data. The data generation has been an integral project, providing a training data for our model.

Mohamed Gamil

During this project, I played a primary role in the development and implementation phases, focusing on both software management and the practical applications of our research. Below, I outline my contributions in detail:

1) Software Development: Managed the overall software development lifecycle within our project, ensuring that all technical requirements were met efficiently and effectively. This involved overseeing our version control systems using GitHub, which facilitated collaboration and maintained the integrity of our codebase.

2) Implementation of FSRN-CNN: Took charge of implementing the FSRN-CNN architecture as described by Thapa et al. This process included not only coding the model and training it, but also analyzing its relevance to the project's specific requirements.

3) Design and Training of the TREX Neural Network: Designed the TREX neural network, adapting it from theoretical frameworks to a fully functional model specified to our project's objectives. I was responsible for the complete cycle from coding to training the model, optimizing it, and testing its efficacy on both synthetic and real-world data.

4) Documentation and Reporting: Authored the description of the neural network as well as created the respective figures. Also took charge of writing the results, conclusion, and future work sections of our project report as well as reviewing the other sections.

5) Team collaboration.: Introduced the Objective Key Result (OKR) framework to the team, which played a critical role in deciding the individual weekly tasks as well as analyzing the previously completed tasks. In addition, proposed bi-weekly "hackathons" to ensure re-alignment between the three individual tasks and the respective code.