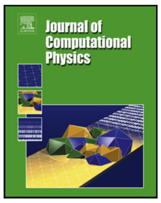
Journal of Computational Physics (2019)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



Obstacle Identification using Learning based on the Wave Equation

Adar Kahana[a,*], Eli Turkel[a], Shai Dekel[a], Dan Givoli[b]

[a] *Department of Applied Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel*
[b] *Department of Aerospace Engineering, Technion, Haifa 32000, Israel*

ARTICLE INFO

Article history:
Received 1 May 2013
Received in final form 10 May 2013
Accepted 13 May 2013
Available online 15 May 2013

Communicated by S. Sarkar

ABSTRACT

We model the inverse physical problem of identifying an underwater obstacle by using the acoustic wave equation. We store the solution in a set of sensors placed in the medium and use this partial information to retrieve the shape, size, location and other properties of the scatterer. The problem is very ill-posed and so we propose a learning algorithm to get accurate approximations.

1. Introduction

1.1. Physical model

We wish to locate obstacles in an underwater medium using acoustic wave propagation (sonar). Given measurements we formulate the problem as an inverse problem [1]. Inverse problems are important mathematical problems since their solution is frequently information we cannot directly observe. They appear in a variety of fields including acoustics, geophysics, material testing (including non-destructive testing), radar and signal processing, computer imaging and vision, tomography, meteorology and oceanography, remote sensing and machine learning.

A common difficulty with inverse problems is that they are ill-posed. Since, we have only partial information the solution need not exist or be unique. Additionally, when we perturb the parameters of the problem the solution might not depend continuously as a function of the parameters. Solving inverse problems is a very vast field and there are many examples, such as Refs. [2, 3, 4, 5]. As a result, in our case, (solving the wave equation in a medium), one cannot retrieve the solution, based on partial data, in the entire medium but only in a small subset of the it (sensors). Even though the problem is ill-posed we hope to retrieve some properties of the source or obstacle that. This scenario includes at least two interesting inverse problems -

- **Source refocusing -** Given properties of the medium and a set of measurements of the wave field at some points in space and time (sensors), find the original source/s that generated the wave field. Usually we are interested in finding a support of the original source and sometimes additional properties [6, 7, 11, 12, 13].

*Corresponding author: Tel.: +972-546-895-475;
e-mail: adarkahana@mail.tau.ac.il (Adar Kahana)

- **Obstacle location and identification -** Given properties of the medium and a set of measurements of the wave field at some points in space and time and assuming we know the location of the original source/s that generated the wave field, find a scatterer in the medium. The scatterer properties (shape, size, density etc.) are also of interest.

We remark that both problems are ill-posed and sensitive to partial and noisy information (we use only the data recorded in the sensors). Therefore, we will not be able to get perfect reconstruction in any of the cases. However, if we limit our demands to a set of properties we can retrieve a lot of information. For example, in the first inverse problem we are interested only in locating the compact support of the original source. In this work we focus on the obstacle location and identification problem. We are interested in finding as much information about the unknown scatterer in the medium.

1.2. Innovation

The inverse problem of scatterer location and identification has been studied by many authors. Most direct methods for approximating the location and shape of the obstacle involve prior knowledge. Assuming, for instance, that the obstacle is a rectangle we can model it with 4 degrees of freedom (Height, Width and two coordinates lengths). After modeling the obstacle using a set of parameters one frequently uses an optimizer to reconstruct the parameters subjected to a predefined cost function. This process is usually computationally heavy and requires intelligent construction of the cost function. Most of the recent developments in this field involve improving the obstacle parameterization, cost function or optimizer.

From the perspective of machine learning, the field is expanding rapidly. For problems similar to the one we present here, general algorithms that handle input data and fit a certain classifier are commonly used. The general models are then parameterized and modified for the specific task until optimal accuracy is achieved, but there is no relation to the data - it is just a set of numbers we are trying to organize.

We propose a new method which does not require parameterization of the obstacle (can handle arbitrary shapes). Also, using a machine learning process, although the model takes a long time to train - inference is done rather quickly and can be deployed to real time solutions. We construct a model that makes use of the wave equation - knowing the training data was created using it (physically informed). The innovative architecture shows very promising results and an interesting idea of combining a state of the art learning algorithms with the classical methods of applied mathematics, specifically numerical methods for partial differential equations.

2. Numerical development

2.1. Mathematical formulation

We approximate the solution of the wave equation. The continuous formulation is given by [] -

$$\begin{cases} \ddot{u}(\overrightarrow{x},t) = div(c(\overrightarrow{x})^2 \nabla u(\overrightarrow{x},t)) & \overrightarrow{x} \in \Omega, t \in (0,T] \\ u(\overrightarrow{x},0) = u_0(\overrightarrow{x}) & \overrightarrow{x} \in \Omega \\ \dot{u}(\overrightarrow{x},0) = v_0(\overrightarrow{x}) & \overrightarrow{x} \in \Omega \\ u(\overrightarrow{x},t) = f(\overrightarrow{x},t) & \overrightarrow{x} \in \partial\Omega_1, t \in [0,T] \\ \nabla u(\overrightarrow{x},t) = g(\overrightarrow{x},t) & \overrightarrow{x} \in \partial\Omega_2, t \in [0,T] \qquad \partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega \end{cases} \tag{1}$$

where u is the wave amplitude (dependent on the time t and space $\overrightarrow{x}$). For given initial conditions (u_0, v_0) and boundary conditions (f, g, of types Dirichlet and Neumann respectively) the wave equation (1) is well-posed, and so small changes in the conditions cause small changes in the solution and there exists a unique continuous solution to the problem inside the domain.

In this work we are interested in identifying an obstacle - an object placed inside the domain ω - which is a perfect acoustic reflector. Mathematically, we refer to the obstacle as a small sub-region Ω_s such that $\Omega_s \subset \Omega$ and the boundary condition on Ω_s are homogeneous and of type Dirichlet. An alternative is setting the wave velocity ($c(\overrightarrow{x})^2$ in (1)) to zero in Ω_s. The shape, size, location and other properties of Ω_s are unknown.

To find the obstacle we measure the solution to the wave equation at various times. It is physically impossible to record the signal at every spatial point of the domain. Instead, in a physical experiment we can record the acoustic pressure in only a small amount of sensors spread around the domain. The output of a physical scenario is a set of measurements $u(\overrightarrow{x_n}, t)$ such that $\{\overrightarrow{x_n}\}_{n=1}^{\#sensors}$ is a sequence of sensor locations ($\forall n = 1, ..., \#sensors : \overrightarrow{x_n} \in \Omega$). We synthetically simulate the physical experiment of wave propagation to get measurements in artificial sensors in our medium Ω. Without knowing the entire solution $u(\overrightarrow{x}, t)$, but only the recordings $u(\overrightarrow{x_n}, t)$, we want to recover Ω_s. This partial information makes the problem very ill-posed.

There are several methods of identifying Ω_s in the literature. For example, assuming prior knowledge about the shape and size one can parameterize Ω_s and then try to "guess" Ω_s, run the simulation and define a scoring function to determine if our guess was correct. We then use different parameters (new guess) until we maximize the score. This is an optimization process that usually takes a lot of computational time. There are several methods that improve this process and other entirely different methods. All of them require prior knowledge and a lot of computational time. We propose a method that requires no prior knowledge about the obstacle and after the model finishes training (which might be long), predicting the output will take only a few milliseconds.

2.1.1. Numerical scheme

We approximate the domain using a uniform three dimensional grid sized $l \times m \times n$. We apply a boundary condition on the boundaries and solve inside the domain. Some of the nodes inside the domain are declared as the sensors such that the number of sensors is much smaller than the number of nodes in the domain, i.e. for n_s sensors $n_s \ll l \cdot n \cdot m$. Each sensor is a node in the grid such that the set $\{(x_k, y_k, z_k)\}_{k=1}^{n_s}$ is the set of coordinates of the sensors ($\forall k : 1 \leq x_k \leq l, 1 \leq y_k \leq m, 1 \leq z_k \leq n$). We model the obstacle as a shape inside the domain such that every node inside the obstacle has the value zero along the entire computation procedure. To model it we use a stair-casing process which adds numerical errors that are negligible.

We approximate the solution of the wave equation in space and time using a second order accurate finite differences scheme. We solve forward in time using a finite number of time samples and use the Courant-Friedrichs-Lewy condition:

$$\Delta t \leq \frac{1}{c\sqrt{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)}} \tag{2}$$

such that $\Delta x, \Delta y$ and Δz are the distances between two adjacent points in the directions x, y and z respectively and c is the wave propagation velocity in the domain.

We solve the equation for n_t time samples. After simulating we get a matrix of size $n_t \times n_s$ containing the acoustic pressure received in each sensor in each time step. To create a data-set of measurements we run this simulation multiple times, such that the difference between each and every simulation is the shape, location, size and other properties of the obstacle explained later. Each simulation is labeled with a $l \times m \times n$ sized binary image describing the obstacle.

After running a large amount of simulations denoted in s we get a labeled data-set with samples organized in a table of size $s \times n_t \cdot n_s$ (after column-stacking the data from the sensors) or a tensor of size $s \times n_t \times n_s$ (without column-stacking). The two methods are discussed in sections 2.2.1 and 2.2.2. The labels are a tensor of size $s \times l \times m \times n$. A two dimensional example of the source and sensors placement and an arbitrary scatterer is given in Figure 1.

After training we predict a binary image of the obstacle, denoted as y_p. This element represents the medium Ω which we chose as homogeneous that includes a scatterer Ω_s. The binary image is the equivalent numerical interpretation of Ω, so essentially our scheme recovers the medium in which the PDE is solved. To assess the performance of our model we use a subset of the samples created (called the testing data-set) of size s_{test} and compare each prediction y_{p_k} to a known y_{t_k} such that $k = 1, ..., s_{test}$. To compare y_{p_k} and y_{t_k} we have several measures of the error:

- **Mean Squared Error:** $\frac{1}{l \cdot n \cdot m} \sum_{k=1}^{s_{test}} |y_{p_k} - y_{t_k}|^2$. Sums the amount of mistakes (wrong prediction) and divides by amount of predictions.

- **Intersection over Union:** A popular measure for segmentation: $\sum_{k=1}^{s_{test}} \frac{|u_{p_k} \cap y_{t_k}|}{|u_{p_k} \cup y_{t_k}|}$ counts the amount of pixels in the intersection and divides by the amount of pixels in the union.

2.2. Deep-learning framework

We propose a different approach for the learning process. Instead of using parameterization we address the scatterer as a binary segment in the medium. We enforce homogeneous Dirichlet boundary conditions (reflective

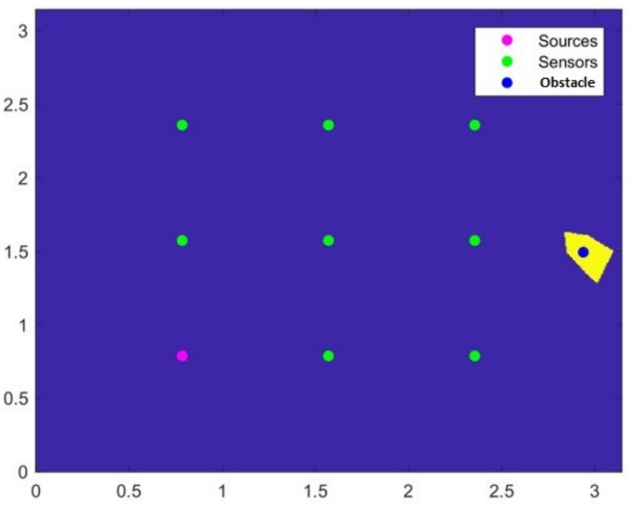


Fig. 1: Example of a 2D medium with 8 sensors (green), a source location (pink) and a binary segment of an obstacle (yellow)

boundaries) on the edge of the scatterer that ensures that the area Ω_s will have zero wave amplitude and zero velocity. Therefore, we consider the compact Ω_s as a segment inside the image that we are trying to locate.

We build a neural network such that the input is the sensor data for recorded for each forward problem with a different scatterer (either a matrix or a tensor as described in the last section). This input is then either convolved with unknown filters (convolution layer) or multiplied by an unknown matrix (dense layer). The unknown coefficients are called weights. We define a target function (specifics are given in the subsections) and iterate using a gradient descent algorithm to minimize the target function with respect to all weights. We design the network architecture (amount and sizes of convolution and dense layers) to get the best convergence of our gradient descent algorithm, resulting in a better intersection over union score on a predefined testing data-set.

2.2.1. Naive approach

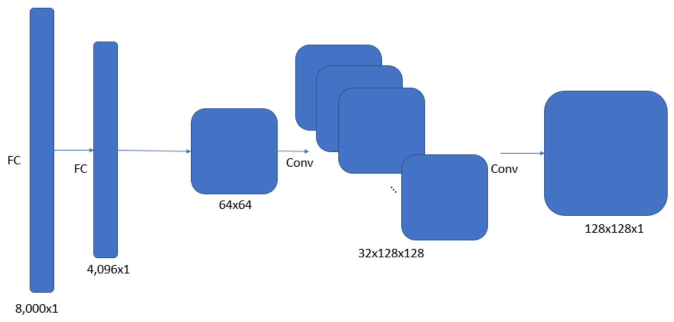
The initial approach we considered was a neural network with a very simple architecture - a fully connected layer followed by a convolution layer. The input is the column-stacked recordings in the sensors (that we synthetically produced during the forward solution of the wave equation) and the output is a probabilistic segmentation map of the obstacles. Training is done on a data-set consisting vectors of recordings in the sensors as data and binary images (segmentation maps) of scatterers used to produce the sensors data as labels. In this case the input data is column-stacked values recorded in the sensors for each sample, yielding a $s \times n_t \cdot n_s$ sized matrix.

We train the network to minimize the Negative Log Likelihood loss function. The output is a matrix of probabilities with size $l \times m \times n$ denoted by $\hat{p}$. Each coordinate of $\hat{p}$ is the probability of that coordinate to be inside Ω_s. Hence, $\hat{p}$ is the probability of pixels to be inside the scatterer. We turn $\hat{p}$ into a binary image using a threshold and check the accuracy using the methods explained in 2.1.1. Detailed results are given in section 3 and a sketch of the network architecture is given in Figure 2(a).

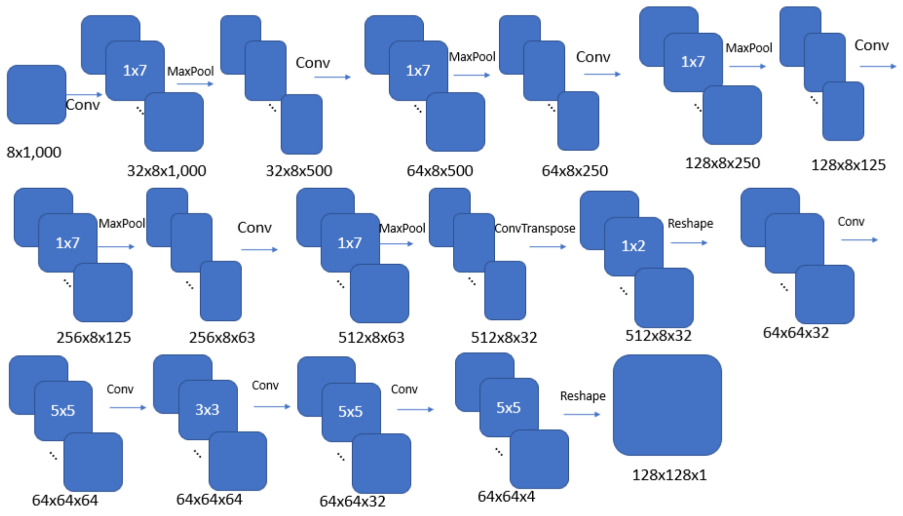
2.2.2. Physical architecture, space-time flow

The motivation for this architecture comes from the physical nature of the problem we are solving. The input data for the network is a time series of recordings. If we do not column-stack we get a tensor of size $s \times n_t \times n_s$. The input data is a time series while the output is a segmentation map of the medium, meaning data in space. We want the network to manipulate the data in a fashion that takes into account a transition from time to space. We propose a different architecture, more complex then the naive one, to take these elements in account. The architecture is demonstrated in figure [].

The first set of layers are 1-dimensional convolutions and pooling - transferring information from the channel



(a) Architecture of the Naive network



(b) Architecture of the space-time flow network

Fig. 2

(recordings in each sensor) to a new dimension which will be the spatial dimension. Detailed results for this architecture are given in section 3 and a sketch of the network architecture is given in Figure 2(b).

2.3. *Physically informed loss*

So far the networks we designed were not influenced by the physical nature of our problem. The only use of the wave equation was creating the data using a forward numerical solution of the wave equation. Although the results using advanced deep learning methods were exceptional, we propose a method that makes the network "aware" of the physical problem it is trying to approximate. We want to exploit the physical aspects of the equation and integrate them into the network. There are several methods of building neural networks that take classical methods from the applied math literature and enforce them on the learning mechanism [14] and achieve remarkable results. We propose a method specific for medium properties reconstruction.

We design a penalty for the network using the properties of the physical wave equation. We recall that the output of the network is a probability map $\hat{p}(\overrightarrow{x})$ where each element is the calculated probability of a pixel to be inside the scatterer. Therefore, $\hat{p}$ is a predicted segmentation map of the scatterer - noted as Ω_{s_p}. The prediction is an approximation of the domain - meaning that if we solve the wave equation using this prediction and record the values in the sensors, we can compare them to the true values recorded in the sensors (during the training process we have access to these values) and thus penalize our domain prediction based on the sensors prediction.

We now discuss the training session. If we assume that $\hat{p}$ is a binary image, we can use it as a medium and solve the wave equation to get recordings in the sensors $\{\overrightarrow{x_{n_p}}\}_{n=1}^{\#sensors}$. The input of the network are all the true recordings in the sensors so in the training part we have $\{\overrightarrow{x_{n_t}}\}_{n=1}^{\#sensors}$, which are the true recordings in the sensors that we train with. We use the Mean Square Error to compare the predicted and true recordings in the sensors. We define the loss function for our network as the weighted sum $PhyLoss = \alpha \cdot l_1 + (1-\alpha) \cdot l_2$ such that l_1 is the Negative Log Likelihood loss while l_2 is the Mean Squared Error between the sensor recordings. The parameter α is chosen via trial and error. Using $\alpha = 1$ is using exactly the architectures described in the previous subsections and using $\alpha = 0$ means approximating based only on the wave equation.

Implementing this method we faced two difficulties:

- We assumed that $\hat{p}$ is binary but the output of the network gives $0 \leq \hat{p}(\overrightarrow{x}) \leq 1$. The theoretical problem in this case is that the boundaries of the scatterer Ω_{s_p}, instead of being zero (reflective), are probabilities between 0 and 1. Therefore we get unpredictable behavior around the scatterer and the solution for the wave equation (calculating l_2) will produce useless results. Also, if we threshold $\hat{p}$ to get a binary image the loss function will have a non-differentiable element and the gradient descent algorithm used in training the algorithm will fail.

  In order to overcome this problem, instead of considering the scatterer as a reflective element in the medium - we claim that the wave propagation velocity is close to zero inside the scatterer. Hence, we solve 1 but with a different operator:
  $$\ddot{u}(\overrightarrow{x}, t) = div((1 - \hat{p}(\overrightarrow{x}))c(\overrightarrow{x})^2 \nabla u(\overrightarrow{x}, t)) \tag{3}$$
  In this way if the network predicts correctly, the velocity inside the scatterer will be around zero and the waves will not propagate inside as desired. Otherwise, the loss will be high, setting a penalty for the next gradient descent step.

- The loss function which determines the next gradient descent step, has to be differentiable. This constrains the implementation to only differentiable actions. In our case, we use a finite difference solver which can be constructed using only linear operations so we can implement a function that solves the wave equation and can fit the loss function. Using Keras [], we observe that the automatic differentiation mechanism recognizes the implementation as differentiable and trains the network using this custom loss function.

3. Numerical tests and results

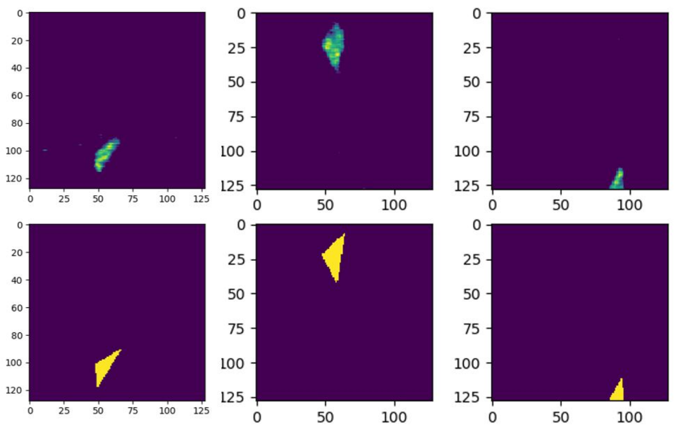
In order to visualize the results we present findings from a 2-dimensional model (A 3-dimensional model was trained as well). We created 25,000 measurements with varying arbitrarily shaped polygons (not necessarily convex) and split them such that 90% of the samples are used for training and the rest for testing. The medium size is $[0, \pi] \times [0, \pi]$ and covered with 128×128 points. We used 8 sensors and 1000 time steps. We emphasize the very small amount of measurements - adding more gives better results but we want to test our architectures on more difficult tasks (in terms of data shortage and limited resources).

Each sample in the data-set has a 8×1000 matrix (the recordings in the sensors) of real values and a label - a 128×128 binary image of the arbitrarily shaped polygon. The models we trained were the naive approach with negative log-likelihood loss, the time-space approach with negative log-likelihood loss and both with the physically informed loss. The intersection over union score is the chosen accuracy metric.

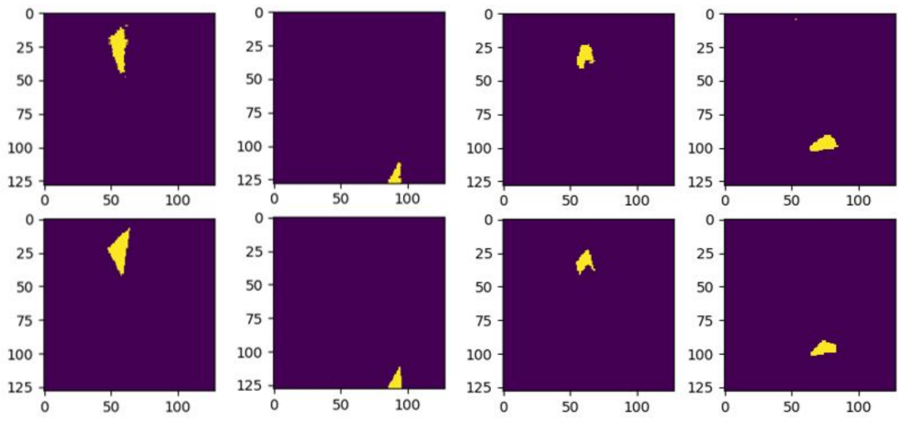
Training the naive network produced remarkable results of 66% intersection over union score for 32 million weights. In this experiment we used the NLL loss for training with around 400 epochs (trained for a couple of minutes on a single nVidia Geforce 1050 on a laptop). The threshold used to compare the predictions and true grounds was 0.5 (if the probability of the pixel to be inside the obstacle is over 50%, the pixel is declared inside the obstacle). The 66% IOU score was the highest score we managed to achieve without over-fitting with all architectures on this specific data-set and will be our benchmark.

Using the architecture in section 2.2.2 we achieved the same 66% benchmark but this time for a much smaller amount of weights - around 4 million. To empirically prove that the architecture in section 2.2.2 is more efficient we conducted a different test - we used a lighter dense layer in the naive version with only 2 million weights and compared to the new architecture with 2 million weights as well. The results were 51% for the naive and 57% for the new architecture, giving us higher accuracy of more then 2 standard deviations (definite improvement). Additionally, we reached convergence for a much smaller amount of epochs without over-fitting.

We tested the benchmark naive architecture with the physically informed loss described in section 2.3. With the same amount of data, the calculation of the physically informed loss involves solving the wave equation 25,000 times in each gradient descent step which was not possible with our current resources and we had to reduce the problem in



(a) Examples images of the probability of the pixels to be in the scatterer - the network output as is. Top images are probabilities and bottom are true binary segmentation maps



(b) Example images of the predicted segmentation maps after using a threshold on the probabilities

Fig. 3

order to create test scenarios. The first interesting result was that for a much smaller case, specifically with 100 time steps instead of 1000 and for both $\alpha = 0$ and $\alpha = 1$ we received an IOU score of 40%. We also managed to show that for $\alpha = 0.5$ the model was able to train and converge but not in a sufficient time in order to reach the saturation - we cannot yet quantify the contribution of the physically informed loss but we do expect a contribution.

Examples of the segmentation maps the network produced (before and after setting a threshold for the probabilities) are given in Figure 3. We observe that even non-convex obstacles were inferred correctly. We remark that images with no obstacles were also generated and inferred correctly.

4. Conclusion

We present a new approach for recovering a scatterer in an underwater medium. We solve the acoustic wave equation numerically to create a data-set. We train several neural networks with different architectures and parameters where the goal is to retrieve the unknown scatterer. The method we propose does not need a parameterization of the scatterer and can recover complex geometries. After training the model, the inference takes milliseconds and the model can be used for real-time applications.

We present a method to use the solution of the wave equation embedded in the neural network and create a physically informed process. We penalize the model using the physical solution of the wave equation to get more accurate and robust results.

References

[1] D. Colton, R. Ewing and W. Rundell, *Inverse Problems in Partial Differential Equations* (SIAM, 1990).

[2] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, 3rd edn. Applied Mathematical Sciences, Vol. 93 (Springer, New York, 2013), pp. xiv+405

[3] V. Isakov, *Inverse Problems for Partial Differential Equations*, Applied Mathematical Sciences, Vol. 127, 2nd edn. (Springer, 2006).

[4] A. Tarantola, *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation* (Elsevier, Amsterdam and New York, 1987), pp. xvi+613.

[5] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers in Applied Mathematics, Vol. 23 (SIAM, Philadelphia, PA, 2002), pp. xvi+183.

[6] C. Bardos and M. Fink, *Mathematical foundations of the time reversal mirror*, Asymptotic Anal. 29 (2002) 157182.

[7] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, 3rd edn. Applied Mathematical Sciences, Vol. 93 (Springer, New York, 2013), pp. xiv+405.

[8] M. Fink, F. Wu, D. Cassereau and R. Mallart, *Imaging through inhomogeneous media using time reversal mirrors*, Ultrason. Imaging 13 (1991) 179199.

[9] C. Bardos and M. Fink, *Mathematical foundations of the time reversal mirror*, Asymptotic Anal. 29 (2002) 157182.

[10] P. Blomgren, G. Papanicolaou and H. Zhao, *Super-resolution in time-reversal acoustics*, J. Acoust. Soc. Am. 111 (2002) 230248.

[11] D. Givoli and E. Turkel, *Time reversal with partial information for wave refocusing and scatterer identification*, Comput. Methods Appl. Mech. Eng. 213 (2012) 223242.

[12] I. Levi, E. Turkel and D. Givoli, *Time reversal for elastic wave refocusing and scatterer location recovery*, J. Comput. Acoust. 23(1450013) (2015) 129.

[13] , J. Comput. Acoust. 26(1850016), No.02, (2018). A. Kahana, E. Turkel and D. Givoli, *Convective Wave Equation and Time Reversal Process for Source Refocusing*

[14] , J. Comput. Physic. 378(686-707), (2019). M. Raissi, P. Perdikaris, G.E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*