Can the Depth Map computation potential of Stereo Images be transferred to Mono Acoustic Recordings?

-Mapping Space with Time-

Xabier Oyanguren Asua

Abstract— We confirm, following [1], that a single channel recording of the echoes generated by a spiked wave, convey enough information as to map the 3D disposition (or at least the depth information) of the elements in a closed scene. For this, neural networks are successfully trained to transfer the potential of stereo images to generate depth maps into single channel sound echo recordings.

1 Introduction

Employing projective geometry it is possible to generate depth maps: 2D images where each pixel has the 3D distance of the projected point to the projection camera plane. Libraries like OpenCV include specialized modules with functions that allow it. This is the so called stereo reconstruction, which is actually the way in which we humans can perceive visual depth. By knowing the correspondence of points between what we see in each eye, a simple geometrical triangulation known as disparity will give us the depth of those points.

In fact, when we hear, since we have two auditive spatially shifted inputs, we can also perceive depth: this stereo acoustic reconstruction is tightly related to the stereo imaging panorama we just mentioned. Some animals like bats actually relay on this spatial reconstruction based on acoustic waves for 3D navigation (known as echolocation).

Yet, it was proven relatively recently, in reference [1], that in reality, a single channel of any wave that conveys reverberant information of a room should be enough to generate depth maps. Meaning it is not absurd to think that a bat with a single ear could still use echolocation. That is, as amazing as it may sound, it is possible to convert a single dimensional (time) information into almost three dimensional spatial information (a depth image). Could we force this further and extract full three dimensional information of a room from that single wave detection?

2 State of the Art

The authors that had the idea originally in 2020 [1], claim that the depth images they generate from these single channel temporal data, are actually 3D spatial images. This is not really true, since a depth image only contains information of the objects in front of the projection plane. This can be used to generate a 3D point cloud, but such a cloud will be sparse and anisotropic. A multiview stereo approach is the requirement to generate a 3D scene in all of its glory [3]. Multiple algorithms exist for this truly 3D image generation from stereo imaging, as reviewed in [3]. Some use multiple depth range maps directly, and some others employ the point could obtained from each depth map, merged together with a 3D point cloud registration algorithm [4]. For such a registration task, deep learning seem to provide accurate results [5].

Coming back to the single wave-detection channel that learns depth reconstruction, the state of the art is still in a fundamental research step. It was only very recently (2021) proved in physical terms that such an inverse problem, even if ill-posed, is theoretically consistent [2]. As a practical proof, the wave employed in the original paper was a 550 nm pulsed light, which flood-illuminated the scene, the light echoes of which were then detected using a single photon avalanche diode (SPAD). The time of arrival of each echo is strictly correlated with the distance to the objects the wave encounters. The idea aslo seems to work for sound waves and other electromagnetic waves, like radio waves [2]. Yet, in all cases only single view depth maps were generated. The question of whether this information channel also conveys information about the full 3D disposition of the objects in the room (a whole point cloud) remains unanswered. It is yet expected to be the case, since a spherical wave, before being reflected back to the origin, maps the surfaces of all the scene. In particular, this means that even if a camera is not able to see what is behind an object blocking the light rays, the echoes of a wave convey information about the presence of the object behind this wall.

3 Objectives and Proposal

Since it is something yet unanswered by the scientific literature, it would be interesting to prove whether a single sound wave channel conveys enough information for the generation of a whole 3D point cloud of the room (which requires multiview stereo if using depth maps). However, due to the ambitious nature of such an objective and the limited time till the result presentation, we suggest a staggered project proposal in two parts, being the present paper the one concerning all the first steps, and leaving the rest for a future work. As such, the objective of this first part of the study is to try to replicate the result of [1], but employing stereo reconstruction and sound instead of a SPAD and a pulsed light source. For this, the proposal is the following one:

1. Using two cameras, build a physical set-up to fix them as close and aligned with each other as possible.
2. Using Python3 with PyQt and OpenCV, build a graphical user interface (GUI) and back-end to calibrate the cameras, obtain the intrinsic and extrinsic parameters, the rectification maps and ultimately the disparity maps showing proportional information to depth. Post-process them.
3. Add a feature to the GUI to create datasets pairing depth maps with their corresponding recorded echo sound-waves. Post-process them.
4. Explore different Deep Learning architectures with Pytorch, training each network with the generated dataset, and benchmarking the results.

This would allow us to prove that effectively the conclusions of [1] are right. The second part of the work would then consist on:

1. Train (separately) four networks to produce live depth maps using a fixed speaker and a microphone from four different spots of the room. This would allow us to generate live depth maps from four different angles simultaneously (since four microphones are affordable, unlike eight cameras).
2. Using a 3D point registering state of the art library, like [6], generate real 3D images of the room object/person shapes from the four depth maps at each time.
3. Train a single wave emitter/receiver channel to generate these whole 3D images using a mono measurement channel (not the four we used to generate them).

Which would let us prove whether the still unanswered assertion is right or not.

4 METHODS AND MATERIAL

The first three points of this section were implemented as part of a graphical user interface using Python3 and its PyQt library, which has let us deploy a fully usable GUI to repeat those steps if wished, letting all the tunable parameters as graphically modifiable for the user. The deep learning part on the other hand, was implemented using a Jupyter Notebook. All the generated code, GUI and tools can found in our GitHub repository:
https://github.com/Oiangu9/Reconstructing_Space_with_Time

4.1 Depth Maps using Stereo Imaging

The employed pair of imagers (left and right cameras) had an *OmniVision Technologies OV9282* sensor with 1280 x 800 active pixels and sensitivity arriving to near-infrared below 850 nm, as mounted in a *RealSense D435i* device. The device included an infrared 850 nm laser of 360 mW optical power that projects a static infrared pattern (invisible to human eye) that gives texture to untextured regions of the infrared images. This helps the point registration between rectified left and right images. The cameras and laser were controlled and accessed employing *pyrealsense2* which is a Python wrapper for the device SDK.

The pipeline for the generation of depth maps, is graphically shown in Figure 2, and followed the next steps:

(a) For each of the two cameras, a set of 20 images showing each a 9x6 chessboard all over the field of view was taken, in order to get corresponding points between the image projection plane and real world points (considering the chessboard as the static reference frame, even if it was the displaced element). The corner detection was done using the OpenCV function `findChessboardCorners()` to be then refined with a sub-pixel estimate provided by `cornerSubPix()`.

(b) These world-image corresponding points for each camera, were used to obtain the intrinsic parameters of both cameras, following the OpenCV function `calibrateCamera()`. Based on Refs. [7], this implementation considers the search of the homography taking the $\{z = 0\}$ projective plane of $\mathbf{R}^4$ where the chess board is assumed to be placed (as the reference coordinate system), to each image's projective plane. The homography is estimated first with an analyitically closed formula employing orthogonality properties for the extrinsic rotation vectors of each view (relative to the chess plane). Then this estimate is refined using non-linear maximum likelihood optimizaiton of the distances between the predicted projections of the corners (given the camera intrinsic and extrinsics) and the actual projection locations. Distortion parameters are introduced at this step.

(c) The obtained extrinsics relative to the chessboard in this single camera calibrations, would be enough to compute the relative rotation and translation vectors between the cameras. Yet, a more accurate estimation can be obtained taking into account we know the correspondence of pixel points taken with the different cameras (referenced to the same chess corners). With them, the fundamental matrix can be computed through a non-linear minimization employing `stereoCalibrate()`. This matrix could now let us know given a pixel of one of the image, in which line of the second image its corresponding point should be (reducing the search space for correspondence). Yet, since our objective is just to compute disparity maps, it is simpler to proceed as follows.

(d) Given the relative rotation and translation of the cameras computed decomposing the fundamental matrix, we can easily compute the homography that takes images one of the cameras to a projection where an imaginary camera pole is aligned with the other camera's pole and there is a null relative rotation of their planes. This is called rectifying the images, which causes that the epipole of either projection (left or right), after rectification, be in the infinite for the other plane. This means the epipolar lines for all points will be parallel, and since the camera poles are aligned, they will have the same height in the image pixels. Thus, in order to search for corresponding points between two general rectified images, we only need to compare them row-wise. The rectification is computed using `stereoRectify()` and `initUndistortRectifyMap`.

(e) Moreover, a trivial trigonometric task, represented in Figure 1, would then let us know the orthogonal distance of each pixel's real world position to the plane of projection. This is what we called "depth". We see in the graphical prove that it is just inversely proportional to the disparity of corresponding points (their relative pixel positions). This is why we have enough with the disparity map for our qualitative study. For this, after rectifying any image pair with the calibrated maps, we use the `StereoSGBM_create` class, that employs the Semi-Global Matching (SGM) algorithm [9] applied by blocks (a sophistication of the stated). Finally, if we generate a disparity map for both, left and right images, we can post-process them using a Weighted Least Squares (WLS) filter implemented in OpenCV. Both disparity perspectives were used here for a left-right-consistency-based confidence to refine the results in half-occlusions and uniform areas, and produce a single view depth map.

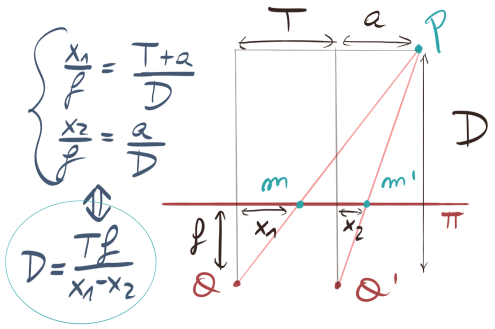


Figure 1: Given O and O' are the aligned projection centers of the rectified images, with a known relative distance T and a common projection plane π with focal length f, given we know m and m' are the projection of a same point P, at a distance x_1 and x_2 from their image's centers (a disparity of $x_1 - x_2$), the depth D (orthogonal distance of the point to the plane parallel to π crossing the poles O, O') is: $D = TF/(x_1 - x_2)$. Since, T, f are fixed for the cameras are relatively fixed, D is always equally inversely proportional to the disparity $x_1 - x_2$.

4.2 Sound Pulse Emission and Echo Recording

Sound pulses were generated using *numpy* and broadcasted plus recorded employing *sounddevice* [8]. We performed the experiment in two different conditions, let us call them A and B.

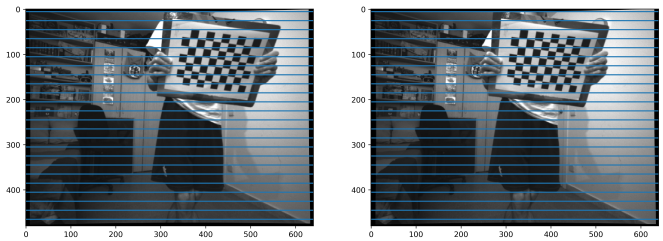
As hardware, for the experimental conditions A, we employed a careful set-up, emitting the sound with an *LG PK3* speaker and recording it with the microphone of some generic earphones

sticked to the speaker. We then performed the experiment B, with the default (poor quality) microphone and speaker of a personal laptop *MSI PS42 8RA*.

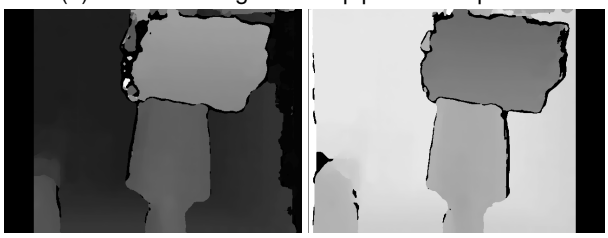
In both cases, the exploratory pulses were designed taking into account the following estimates about the echoes. Sound is well known to travel at about 340 m/s through air. Since we wished to measure depths in a range of $0.5m$ to $4m$ (maximum length of the test room), we needed to record the whole range of sounds between 3 and 24 ms after the emission of the pulse. In order to gather at least three echoes against a same object, we extended this range to 75 ms. It is important to note that the emitted pulse needed to be shorter than 3 ms, in order to avoid any overlap with the echo from an object at 0.5 m. For this, we shaped the emitted pulse as a sinusoid of about 5 kHz with a Gaussian envelope of 0.6 ms standard deviation. This accounted for emitted pulses that took at most 3 ms, with a frequency dispersion of $\pm$ 0.5 kHz.

For the experimenters in the room to be able to talk while data was being collected and to diminish the effect of the noise introduced by the poor quality of the microphones, a band filter was applied in Fourier space, only leaving a $\pm 0.5kHz$ window around $5kHz$.

(a) Detected corners for left and right cam images



(b) Rectified Images with epipolar lines plotted



(c) Left disparity image (d) Right disparity image

(e) Post-processed disparity image

Figure 2: Representative example of different results obtained following the calibration and disparity computation pipeline described in Section 4.1.

4.3 Dataset Generation

We chose the student bedroom that can be seen in Figure 2 as the experimental spot. Since we generated a pipeline that automatically tags a sound recording with a corresponding ground truth depth map, in principle we could generate unlimited samples. Yet, we limited the datasets to 5300 samples. The images were downscaled to a shape 57x68.

Three different datasets were generated. For the experimental conditions A, the audios were recorded with a 48 kHz sampling rate, employing a microphone co-located with the speaker, which was oriented to send the pulses in the direction of the depth stereo cameras. The recordings were treated to erase the first emitted pulse (which was orders of magnitude more intense than the echoes), they were band filtered in Fourier space as explained above and then normalized to the maximum recorded sample of the dataset. Finally they were down-sampled to half their recorded number of points using local averaging and were saved in absolute value (since the recordings appeared to be coarsely symmetric horizontally). The images were normalized in the range of $[0, 1]$, setting a common minimum and maximum for all samples as the limit values found in the dataset. Each recording of this dataset A, resulted in a vector of about 1900 samples, with a defined envelope, but still after the local averaging, with big internal oscillation (as can be seen in Figure 3). Since our hypothesis is that the relevant information is in the arrival times and intensities of the peaks/echoes (and not their internal structure), we created a slightly modified version of dataset A, which we will call dataset A', where the recordings were additionally down-sampled with a local averaging until a length of about 600 samples (in Figure 4).

For test B, on the other hand, audios were recorded at 44.1 kHz sampling rate, in notably poorer hardware and steric conditions (with the speaker of the laptop emitting the pulses from a more occluded position, and the microphone not co-located with the speaker). The audios were only Fourier filtered and otherwise left "raw". The images were not normalized either. Example recordings of this dataset can be found in Figure 5.

4.4 Deep Learning Architectures

4.4.1 Multilayer Perceptron

The first obvious choice was to use a simple multi-layer perceptron (MLP). We used leaky ReLu non-linear activations and dropout. It was designed with a slight bottleneck reminiscent of the autoencoder architecture, in order to force the network to look for the important peaks in the recordings for the required dispositions of the scene. The network was composed of seven fully connected layers, that took the initial sound recording of N samples to a flattened 57x68 vector, reshaped as a depth map. Each layer had a number of neurons following $[N, N, N/2, N/2, N, N, N, 57x68]$.

4.4.2 MLP+(De)convolution layers

A network that started with convolutions over the sound recording appeared to be unsuitable, since the translational invariance introduced by convolutional layers, which is usually an advantage, could spoil the echo time-arrival information, which presumably relays on the position of the peaks within the input vector. Yet, for the image generation task, convolutional layers are known to be a reasonable choice to reduce the number of required

parameters. Thus, we decided to design a more versatile network by adding some "de-convolution" layers after reducing the dimensionality of the recordings to an encoded latent space using fully connected layers. Four fully connected layers with neuron numbers $[N, N/2, N/2, 900]$ and five de-convolution layers (after a reshaping to 30x30), of filter sizes 6 (the last one had shape (8,19)), together with 10 filters per layer were employed. Each layer was followed by a leaky ReLu activation.

4.4.3 Loss and Hyperparameters

As loss, a simple mean square error (MSE) was employed, which is the typical choice for facing a continuous (image) regression task as ours. As a matter of comparison between the resulting network performances, we also employ a visual similarity index like the SSIM [10].

The training on the other hand is done using an Adam optimizer with a learning rate of 0.0001 and beta parameters 0.99 and 0.999. A training time of less than 15 minutes (about 200 epochs) with an *MSI PS42 9RA* laptop, which has an *MX250* graphic card, was enough for approximate convergence of all studied networks, requiring less epochs for datasets A and A' than B, presumably due to the fact that in the first ones, data was normalized.

5 Results and Discussion

Some resulting predictions on the 300 samples left aside as test set (samples that were not shown to the network during training) both for experiments A and B can be found in Figures 3, 4 and 5, together with the employed recordings as input and the ground-truth disparity maps. Visually, the achieved results are quite impressive, since not only the networks are capable to correctly predict when there is a person in the room, or the distance of the person to the camera plane, but even when the person is lying on the ground or is kneeling, among others.

Some goodness of fit metrics computed on the test sets for each network and dataset are shown in Table 1, together with the employed number of parameters for each network. On the one hand, we see in all cases that reducing the number of fully connected layers to be replaced by de-convolution layers, resulted in worse results, both for the mean square error (MSE, the smaller the better) and the visual structural similarity index (SSIM, the higher the better [10]). In fact, visually, in the figures, we readily see that the fully connected network generates less noisy predictions. Yet, it must be noted that the number of parameters for the convolution networks are less than a half in all cases, so the trade-off might be worth-it.

Table 1: Metrics computed for the different trained models for each dataset's train set. MSE stands for mean squeare error, SSIM is the visual similarity index of Ref. [10].

Dataset	Network	Num. params.	MSE	SSIM
A	MLP	23.12e6	0.00088	0.930
A	MLP+Deconv	7.3e6	0.0012	0.901
A'	MLP	4.2e6	0.00093	0.915
A'	MLP+Deconv	2.02e6	0.0011	0.906
B	MLP	14.6e6	0.019	0.720
B	MLP+Deconv	4.3e6	0.021	0.660

It is also interesting to note that even if the quality of the gathered data in experiment B is significantly worse than the other two (which is of course reflected in the poorer metrics), we are still able to predict with a qualitatively good precision the depth maps for all the situations contemplated in A and A' as seen in Figure 5. This is a strong indicator that the recording conveys indeed clear information about the disposition of the scene.

On the other hand, regarding the similar metrics for A' and A, we can see that, even if we down-sampled the recordings and only left the envelope, the network was equally capable to predict the depth maps. This seems to confirm that most of the information is indeed carried by the peaks and their relative intensities and distances, rather than a finner underlying high frequency structure. This observation allows the usage of lighter networks (more than three fold lighter) with practically no quality loss.

Finally, it is important to notice that in reality what the networks learned were the particular acoustic rules of the room, since, we tried to use them to predict depth maps in a different room and the results were very poor. Thus, it is apparent that the network "black-box"-es encode the time arrival/intensity relations for the particular room/furniture disposition shown during the training. It is still interesting to ask whether a general purpose depth detector for the immediate vicinity of the wave emitter/detector could be possibly trained using data recorded in different environments, and perhaps a more complex attention based network.

6 Conclusions and Outlook

To conclude, we have proved that indeed single sound emission and reception channels convey a big part of the depth information, since we achieved the generation of depth maps from them even when recordings were performed with poor quality hardware. It is to be proved yet what happens when such an algorithm is trained in a more general open environment and to see if the acoustic rules learned by the "black-box" could be generalizable. In addition, the second part of the research, described in Section 3, is still to be developed, which is left as future work.

References

[1] A. Turpin et al., *"Spatial images from temporal data,"* Optica 7, 900-905 (2020)

[2] A. Turpin et al.,*"3D imaging from multipath temporal echoes,"* Phys. Rev. Lett. 126, 174301 (2021).

[3] Furukawa, Y.; Hernández, C. (2015). *"Multi-View Stereo: A Tutorial."* Foundations and Trends in Computer Graphics and Vision, 9(1-2), 1–148.

[4] Huang, Xiaoshui, et al. *"A comprehensive survey on point cloud registration."* arXiv preprint arXiv:2103.02690 (2021).

[5] Gojcic, Zan et al. "The Perfect Match: 3D Point Cloud Matching With Smoothed Densities." 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 5540-5549.

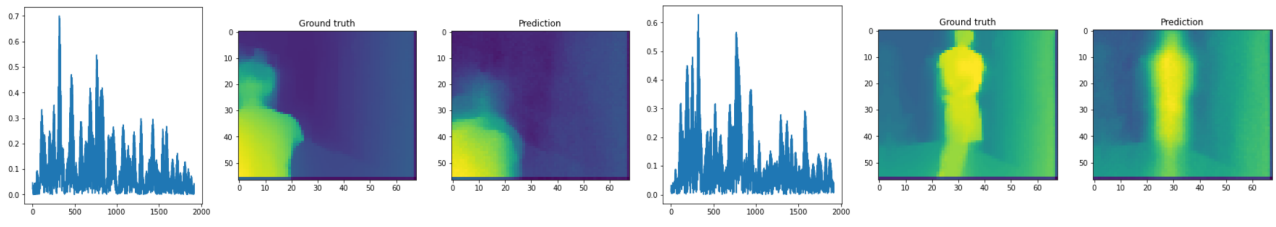
[6] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration", IEEE Transactions on Robotics, 2020.

[7] Z. Zhang, *"A flexible new technique for camera calibration,"* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000

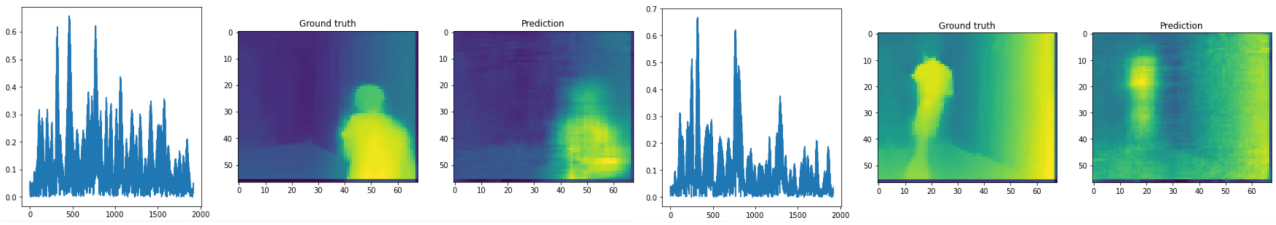
[8] Geier, M., et al. (2020). *Sounddevice*. GitHub. Retrieved from https://python-sounddevice.readthedocs.io/en/0.4.4/

[9] Hirschmuller, H. *"Stereo processing by semiglobal matching and mutual information."* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30(2):328–341, 2008.

[10] Wang, Zhou et al. *"Image quality assessment: from error visibility to structural similarity"*. (2004) IEEE Transactions on Image Processing. 13 (4): 600–612

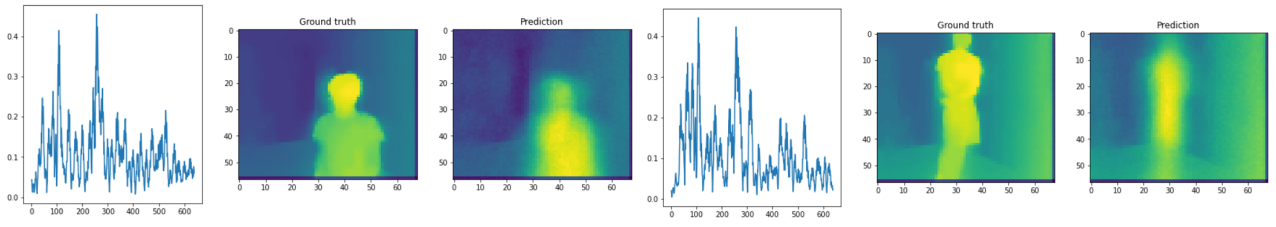


(a) Test set example 1, MLP (b) Test set example 2, MLP

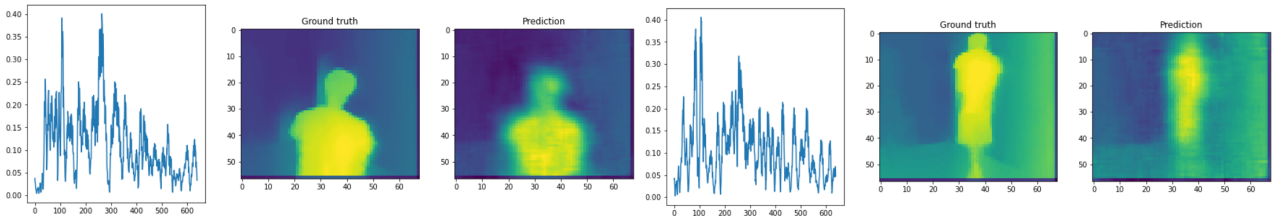


(c) Test set example 1, MLP+Deconv (d) Test set example 2, MLP+Deconv

Figure 3: Representative example predictions for the test set, employing dataset A. In each subfigure, in the left, the recording inputted to the network, in the center, the ground-truth depth map generated with stereo imaging, in the left the predicted map.

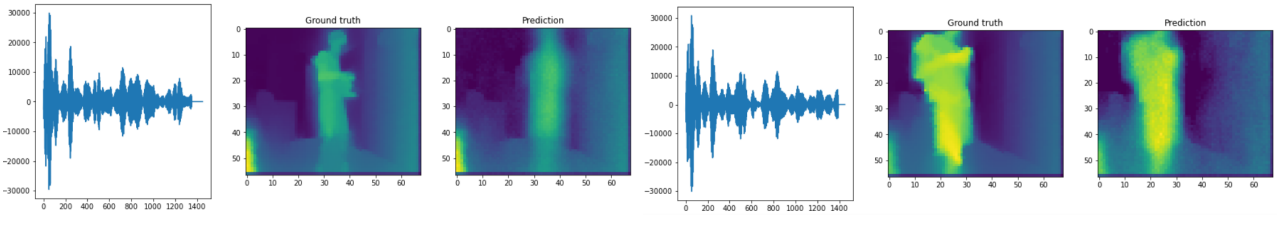


(a) Test set example 1, MLP (b) Test set example 2, MLP

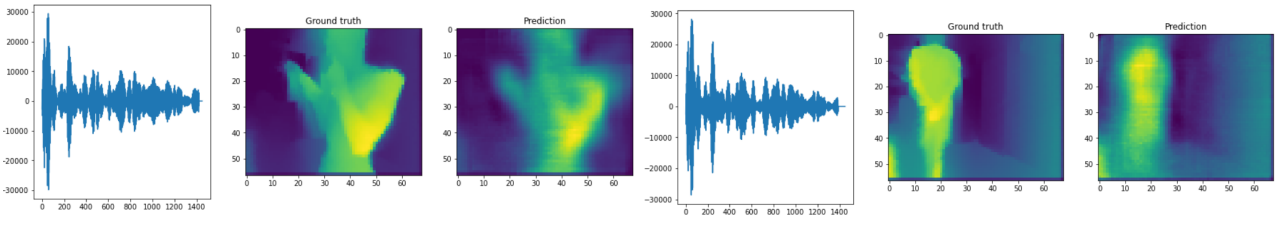


(c) Test set example 1, MLP+Deconv (d) Test set example 2, MLP+Deconv

Figure 4: Representative example predictions for the test set, employing dataset A'. In each subfigure, in the left, the recording inputted to the network, in the center, the ground-truth depth map generated with stereo imaging, in the left the predicted map.



(a) Test set example 1, MLP (b) Test set example 2, MLP



(c) Test set example 1, MLP+Deconv (d) Test set example 2, MLP+Deconv

Figure 5: Representative example predictions for the test set, employing dataset B. In each subfigure, in the left, the recording inputted to the network, in the center, the ground-truth depth map generated with stereo imaging, in the left the predicted map.