GigaDepth: Learning Depth from Structured Light with Branching Neural Networks

Simon Schreiberhuber[1][0000−0002−1138−7576], Jean-Baptiste Weibel[1][0000−0003−0201−4740], Timothy Patten[2][0000−0003−1139−9451], and Markus Vincze[1][0000−0002−2799−491X]

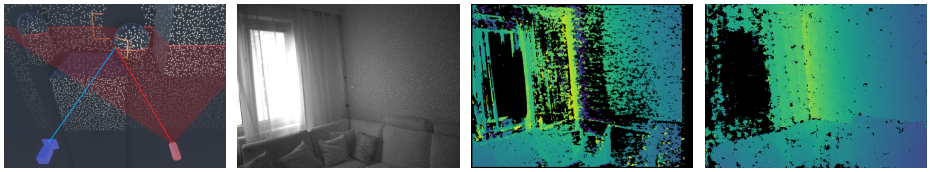
[1] ACIN, TU Wien, Gusshausstrasse 27-29 / E376, 1040 Vienna, Austria
lastname@acin.tuwien.ac.at
[2] UTS Robotics Institute, 81 Broadway, Building 11, NSW 2007 Sydney, Australia
firstname.lastname@uts.edu.au

Abstract. Structured light-based depth sensors provide accurate depth information independently of the scene appearance by extracting pattern positions from the captured pixel intensities. Spatial neighborhood encoding, in particular, is a popular structured light approach for off-the-shelf hardware. However, it suffers from the distortion and fragmentation of the projected pattern by the scene's geometry in the vicinity of a pixel. This forces algorithms to find a delicate balance between depth prediction accuracy and robustness to pattern fragmentation or appearance change. While stereo matching provides more robustness at the expense of accuracy, we show that learning to regress a pixel's position within the projected pattern is not only more accurate when combined with classification but can be made equally robust. We propose to split the regression problem into smaller classification sub-problems in a coarse-to-fine manner with the use of a weight-adaptive layer that efficiently implements branching per-pixel Multilayer Perceptrons applied to features extracted by a Convolutional Neural Network. As our approach requires full supervision, we train our algorithm on a rendered dataset sufficiently close to the real-world domain. On a separately captured real-world dataset, we show that our network outperforms state-of-the-art and is significantly more robust than other regression-based approaches.

Keywords: structured light, depth sensing, CNN, MLP, MLP decision-tree

1 Introduction

Depth sensing is essential for safe interactions in augmented and virtual reality applications as well as mobile robotics. Structured light sensors are a particularly appealing solution for these indoor applications. These sensors predict depth from the alterations of the light patterns they project in the scene rather than the scene appearance, thus overcoming the issue of featureless areas. Spatial neighborhood encoding, which encodes the pattern position of every pixel by creating identifiable structures in the neighborhood of the pixel of interest,



(a) Structured light (b) Left IR image (c) Default disparity (d) Ours

Fig. 1: Triangulation driven by the principle of spatial neighbourhood encoding (a) operates on data captured by a camera (left, blue) to match the pattern projected by a projector (right, red). The pattern needs to encode the projected direction such that a small region (orange rectangle) around each captured pixel can unambiguously trace back to the pattern position along the the epipolar line (transparent red). Given input images such as (b) the internal processing of the used Occipital Structure Core achieves only spotty depth estimates (c) despite using the stereo setup of the sensor. Our algorithm, GigaDepth, (d) on the other hand can derive much denser and more accurate depth due to its knowledge of the utilized pattern using only one of the cameras (b).

provides a good trade-off between accuracy, cost, and power consumption among structured light methods. The projector only needs to project a fixed sparse dot pattern once on the scene rather than encode every captured pixel separately (fig. 1), or require multiple acquisitions like temporal multiplexing.

Such approaches are, however, very sensitive to pattern distortion, fragmentation and attenuation. In practice the correspondence problem between the captured image and the projected pattern is solved by stereo matching to a prestored reference pattern as used in the Kinect v1 or PrimeSense Carmine [16], trading in some robustness and accuracy with the simplicity of matching algorithms. More modern machine learning-based approaches spark the hope to directly, robustly and accurately decode these spatial encodings, but currently only deliver either accurate (e.g. [7]) or dense (e.g. [21, 14]) depth-maps.

With GigaDepth, this work contributes a novel neural network architecture that decodes spatial neighborhood encodings even if the projected pattern is distorted, fragmented or attenuated by the scene's geometry and surface properties. By combining the strengths of Convolutional Neural Networks (CNNs) for feature encoding with those of regression trees, we are able to extract pattern positions from captured images with higher accuracy and produce much denser output. Regression trees are implemented with Multilayer Perceptrons (MLPs) in a novel weight-selective layer for which we provide an efficient CUDA implementation. Novel datasets for training and testing this approach are artificially rendered and captured with the Occipital Structure Core (fig. 1) as well as an industrial-grade 3D Scanner for ground-truth. Our method outperforms state-of-the-art dot-pattern structured light approaches and active stereo in applications where disparity in subpixel accuracy is required.

2 Related work

CNNs have enabled significant improvements for stereo matching [5, 15, 22], particularly in featureless regions, multi-view geometry [2, 8, 20, 24] or even depth completion [11, 23, 25]. Their ability to extract high-level features makes them very useful in passive stereo setups, that is without a pattern projector, and even monocular setups [17] that produce depth without using any conventional principle like triangulation.

To tackle the problems unique to structured light approaches, research has taken on the problem from multiple directions. Structured light approaches depend on the specific pattern that encodes the depth information. Given hardware that is able to adjust the projected pattern on-the-fly, approaches such as [18, 6] demonstrate performance improvements by selecting patterns based on an optimality criterion. Similarly, the work in [9] describes the design of Hamiltonian encodings to either improve quality or drastically reduce the amount of required images. This is a stark improvement from the beginnings of temporal multiplexing by Altschuler et al. [1] who apply binary/gray coding or the direct encoding by Carrihill et al. [3] where the position is encoded in the intensity.

On the other hand, many works improve upon the decoding step while keeping the projected pattern simple. Recent work in profilometry uses CNNs to (directly regress depth values) capture the topography of a surface from only a single exposure with one standard sinusoidal fringe pattern [13, 19]. This is especially noteworthy as the employed high-frequency pattern might be well suited to resolve small details but does not feature an absolute encoding as the utilized fringes are equally spaced.

Although the use of CNNs can make encoding absolute position obsolete for some applications, optimal robustness can only be achieved with adequate encodings. Spatial neighborhood encoding like the dot-pattern often require only a small patch of pixels to regress the pattern position as shown with popular sensors as the Kinect v1, where simple block matching is executed between a captured frame and the reference pattern [16]. Learning-based methods can internalize knowledge about the pattern and the effects of the scene on its appearance to directly regress pattern positions or depth without lookups in a reference pattern. This is demonstrated in the work of Fanello et al. [7] where random forests are employed to directly regress the pattern position without a matching step by comparing intensity values in a small neighborhood around each pixel. Similarly, Riegler et al. [21] as well as Johari et al. [14] use CNNs to directly regress disparity without searching a reference pattern at run-time.

Albeit this work mainly focuses on the decoding of spatial neighborhood patterns, the sensor used is designed for active stereo and thus makes the comparison with methods like [26] possible. Zhang et al. [26] optimize a CNN based stereo algorithm [15] for active stereo and demonstrates improved performance on Intel RealSense hardware, which by default uses semi-global matching (SGM) [10].

3 Method

Precise depth estimates rely on determining pattern positions at subpixel accuracy for the entire pattern. While it is possible to solve this problem by means of a similarity search within a reference pattern, it is more suitable to use a priori knowledge about the pattern's structure to directly derive the pattern position.

Given a rectified image $I(x)$ and pattern $P(x)$ intensities along a given epipolar line, it is possible to extract the pattern position x_P, where the projected pattern $P(x_P)$ resembles the captured intensities $I(x_I)$ around pixel region x_I. The difference between these two positions (along the horizontal axis x) $d = x_P - x_I$ is called disparity and can express depth $z = {}^{fb}/_{d}$ given the baseline b and focal length f.

To estimate depth for a given pixel $I(x_I)$ both estimating the disparity d or the position x_P suffices. However, directly regressing the position x_P poses a challenge for pure CNNs as they exhibit too much noise over such a large output range. This is a motivation for [21] to limit the output range to 128 pixels and estimate disparity d instead. If, however, short-ranged depth estimates are needed, it is necessary to increase the range of disparity, which leads to the same aforementioned challenge. Another approach of splitting the range of pattern positions x_P into classes leads, together with our requirement of subpixel accuracy, to thousands of classes. While it is unmanageable to use one-hot encoded outputs as typically done in CNNs for every pixel, it is a task gracefully managed by the decision trees employed in [7].

Our approach GigaDepth thus employs a similar hierarchical principle of splitting the output range into smaller, easier to regress regions. Instead of directly deriving decisions for tree-traversal from intensity differences as in [7], we employ a CNN to extract features allowing for more robust decision functions based on MLPs. Fig. 2 shows an overview of our architecture. Note that the use of Local Contrast Normalization (LCN) as shown in the figure is intended to accentuate the pattern in the captured images. While used throughout this work and, e.g. [14, 21, 26], we demonstrate in section 4.4 that its contribution to the performance is minimal. For a deeper discussion, we refer to the the mentioned works or the supplemental.

3.1 Backbone CNN

The first step in our pipeline is to condense the local image data such that the features concerning the pattern and the scene can be efficiently processed by the MLP tree. Compared to current CNNs, this backbone is implemented as a shallow CNN (table 1a) with a receptive field of only 21 pixels, which makes the number of considered pixels similar to [7] ($(21 \cdot 2 + 1)^2 = 1849$ vs. $32^2 = 1024$ for [7]). Other networks (e.g. [17, 21]) employed in similar scenarios are multiple times deeper and use a U-shaped structure such that their receptive field spans the whole image and high-level perception of the scene can be learned. For this pattern detection task, however, we found it sufficient to only consider regions

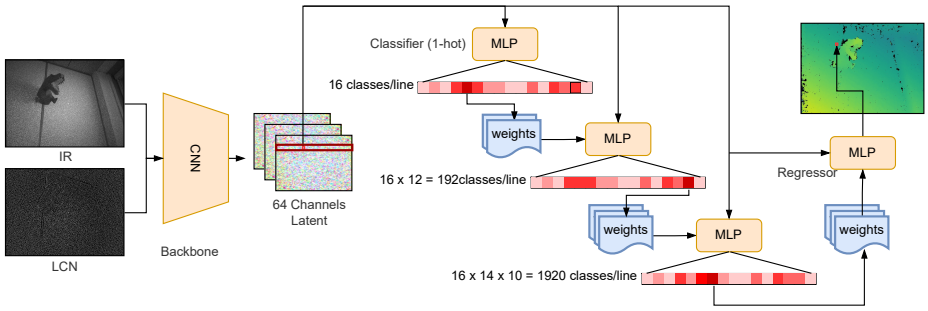


Fig. 2: The proposed network architecture: While the backbone is a relatively shallow CNN, the regressor is a hierarchy of adaptive MLPs operating on individual pixels. The stage 1 MLP features one set of weights for each line and splits it up into 16 classes. Stage 2 has 16 sets of weights for each line to classify into 12 subclasses. Stage 3 further splits these up into 10 categories. Only at the last stage one of 1920 weights / regressors is selected to perform the regression.

large enough to capture uniquely identifiable pattern segments and to detect cues about depth discontinuities around object boundaries.

We furthermore halve the resolution of our feature maps ($1216 \times 896 \rightarrow 608 \times 448$) by convolution with a stride of two early in the network as high-resolution depth-maps are impractical for real-time purposes.

3.2 MLP Tree

For every line of our output image we maintain one specialized tree consisting of one-hot encoding decision MLPs at each node and specialized regression MLPs at each leaf. In contrast to [7], directly comparing two intensity values for a binary decision function, the MLPs constitute much more expressive but also heavier decision functions leading to more robust performance. However, as even the minimally feasible perceptrons are relatively compute intensive, we are forced to use a much shallower tree ($10 - 15$ for [7] vs. 3) with each node splitting into more branches to reach similar width.

Each of these trees is evaluated on a per-pixel basis such that the root node would process the features of one individual pixel to split the output range $x_{P,x}$ into $c_1 = 16$ consecutive, equally spaced regions X_{i_1}. These are further split by consecutive nodes into $c_2 = 12$ regions $\mathcal{X}_{i_1,i_2}$ each. A third stage follows, splitting each of these regions into $c_3 = 10$ regions $\mathcal{X}_{i_1,i_2,i_3}$ leading to a total of $c = c_1 c_2 c_3$ classes/regions. Finally, at the leaves of this tree structure sit specialized regressors that are tuned to estimate $x_{P,x}$ in the small regions $\mathcal{X}_{i_1,i_2,i_3}$.

A few modifications are employed that deviate from this simplified description: We share the trees for two consecutive rows to reduce the overall parameter count. For the same purpose, we have regressor MLPs share weights for four consecutive classes while having per-class weights for the final regression. We also

Table 1: The compositions of our backbone CNN and MLP regression tree. CNN layers with kernel sizes k, stride s and in/output channels C_{in} and C_{out}. Each layer is combined with BatchNorm and ReLU. The classification stages of our MLP tree split the output region into ever smaller subregions until specialized MLP regressors take over. The classifiers have multiple layers with the one-hot encoded output being split in output classes c and overlap o to the neighboring group of classes

(a) CNN backbone layers

	In							Out
k	5	3	3	3	3	5	3	3
s	2	1	1	1	1	1	1	1
C_{in}	2	16	24	32	40	64	64	96
C_{out}	16	24	32	40	64	64	96	160

(b) MLP regression tree stages

stage	C_{in} [*start*, *stop*]	layers [*in*, *hidden*, $^{c}/_{o}$]
class1	[0, 64]	[64, 64, 32, 16]
class2	[16, 80]	[64, 32, $^{12}/_{2}$]
class3	[80, 144]	[64, 32, $^{10}/_{3}$]
reg(out)	[128, 160]	[32, 32, $^{1}/_{1}$]

allow some overlap between the classification results of the classifiers of stage 2 and 3 as the preceding classifications might be inaccurate at the boundaries between neighboring classes/regions e.g. $\mathcal{X}_{i_1}$, $\mathcal{X}_{i_1+1}$. Given e.g. $c_2 = 14$ with an overlap of $o_2 = 1$ means that the MLPs at stage 2 has 16 raw output classes with indices $i'_2 = i_2 + o_2$. In our example, raw results as $i'_2 = 0$ or $i'_2 = 15$ mean that the previous classification result will likely fall in one of the neighboring regions $\mathcal{X}'_{i'_1,0} \equiv \mathcal{X}'_{i'_1-1,14}$, $\mathcal{X}'_{i'_1,15} \equiv \mathcal{X}'_{i'_1+1,1}$. Similarly, the regressors are trained to cover for their neighbors. See table 1b for a detailed account of the involved MLPs.

The described structure requires branching on a per-pixel basis and thus is ill-suited for an efficient implementation on the basis of high-level functionality of popular deep learning frameworks. Therefore we provide a CUDA implementation of a weight-selective layer for pytorch to keep execution time and, more importantly, the memory footprint of our architecture manageable.

3.3 Training

When traversing the MLP tree to reach a leaf, each node takes in the features extracted by the backbone to derive a decision about the path to be taken. If we only apply a loss on the regression predicted by a leaf MLP and apply backpropagation starting there, we would not be able to update the weights of the non-leaf node MLPs. The class indices i_1, i_2, i_3 stemming from these only act to select weights of the according MLPs and do not allow for the gradient to propagate back, making supervision based on principles such as, e.g. photoconsistency, a serious challenge.

We thus employ a training modality that allows full supervision for each of the trees nodes individually, shifting the focus away from elegant means of self-supervision towards the benefits of our branching architecture. We design our system around the availability of complete ground-truth data, which is provided in the form of a novel artificial dataset simulating the sensor. The classification

stages of the MLP trees are supervised by class indices i_1, i_2, i_3 generated by discretizing the horizontal position of the dot-pattern into hierarchical regions $\mathcal{X}_{i_1}$, $\mathcal{X}_{i_1,i_2}$, $\mathcal{X}_{i_1,i_2,i_3}$ that are equally sized at each level. We utilize the cross entropy loss at each level of our MLP tree and apply it to the nodes that would be executed if each MLP performed perfectly. As this will not be the case in practice, we also apply the loss to MLPs that might be traversed if the preceding MLPs are off by one label (e.g. $i_1 \pm 1$). As a result, we train the overlap of class labels described in section 3.2. To train the regressor MLPs at the leaves we use the L1 loss and further employ the same strategy as before to let each regressor cover for its neighbours. The gradients from both losses are propagated all the way back to the backbone CNN and used to update weights via classical Stochastic Gradient Descent. Training the whole system on our artificial dataset takes $\sim 10h$ utilizing one NVIDIA RTX 3090.

We further utilize an edge mask that marks regions around depth discontinuities to emphasise sharpness around edges by increased loss. Another mask is used to remove the loss where the training signal is too ambiguous for spatial neighborhood encodings. This mask covers pixels whose surface do not sufficiently reflect the projector's light due to albedo, distance or occlusion.

Augmentation of the data with noise and a slight vertical jitter of 4 pixels are used to introduce robustness against some of the effects of operating the sensor in real-world conditions. The impact of this measure is discussed in section 4.4.

4 Experiments

Real-world applications require accurate disparity in a subpixel range as well as depth measurements that cover challenging surfaces. Similar to [21] we report the outlier ratio $o(th)$, describing the ratio between the number of pixels that feature a disparity error higher than a given threshold th and the overall number of pixels. We also present the Root Mean Square Error ($RMSE$) of the depth measurements derived by the different algorithms and our specific sensor to give a practical intuition. Pixels with a disparity error greater than one pixel are excluded. The various failure modes and outliers of the different algorithms would otherwise distort this comparison.

A thorough performance comparison to existing methods is conducted using an artificial dataset. The rendering process provides ground-truth, which we use to circumvent some of the sacrifices required by self-supervised training of baseline methods. Finally, we compare on real-world data with ground-truth captured by an industrial grade 3D scanner.

4.1 Baseline methods

The main reference points for our method are HyperDepth [7], Connecting The Dots [21] and DepthInSpace [14], which directly regress a pixel's position within the pattern or the disparity. To contrast with these methods that need a reference image to operate, we also compare to ActiveStereoNet [26], which extends [15]

by modifying the loss to emphasize the dot pattern. It is expected that the usage of the second IR camera gives ActiveStereoNet an advantage wherever the pattern is too weak and classical stereo matching can pick up scene features. Fittingly, all of these methods operate at similar execution times when tested on our RTX 2070 Max-Q with HyperDepth $20 - 70ms$, Connecting The Dots $40ms$, DepthInSpace $20ms$, ActiveStereoNet $45ms$ and ours $60ms$.

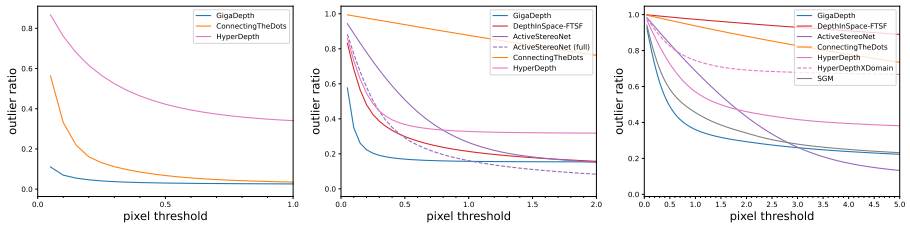
While our rendered dataset enables training of these methods, we deviated from the original methods in a few aspects:

- **Resolution:** For the purpose of comparability, we upscale the results of algorithms operating on a lower resolution than the input resolution. The algorithms themselves operate at or close to their intended resolution.
- **Jitter:** As our sensor hardware exhibits vertical drift (section 4.4) we train all approaches with the appropriately jittered inputs.
- **HyperDepth:** Unlike the original authors [7], Riegler et al. [21] published an implementation of HyperDepth that was used for their baseline comparison. We ported their implementation to CUDA for faster experiments and added k-means clustering to improve the regression accuracy at the leaf nodes. We further utilize deeper trees (16 vs. 14 levels) for improved results.
- **Connecting The Dots:** The original approach of matching with the reference pattern did not converge on our dataset. We therefore use the image captured/rendered by the right camera to have a stereo matching approach during training time. During runtime, the algorithm operates as in the original method, not utilizing the second camera. Despite these efforts, we are unable to bring Connecting The Dots to the same performance levels as on the originally intended dataset. We therefore also include a comparison of our method on the dataset presented in [21].
- **DepthInSpace:** As this method is similar to Connecting The Dots in many aspects and thus facing similar challenges during training, we adopt the same adjustments, which leads to good success. The training requires optical flow [12], such that only 241 of our captured sequences can be utilized as only those have additional captures with a disabled IR-projector. This is more than the 148 training sequences used in the original work [14] but puts it at a disadvantage to the other baseline methods trained on 967 sequences.

4.2 Dataset

The Structure Core allows us to run the algorithms of [7, 14, 21] and [26], thus we render a new dataset based on this sensor. The artificial data is rendered via Unity3D using the High Definition Rendering Pipeline and free assets found on the asset store. Different to [14, 21], which use ShapeNet [4] objects without textures, our objects are textured with partial randomization for selected surfaces[3]. Aside from the ground-truth depth/disparity, we render stereo images as well as

[3] Randomly selected textures on planes used in walls as well as cube, sphere, cylinder and pill shapes.



(a) Rendered ShapeNet (b) Rendered with Unity3D (c) Captured

Fig. 3: Outlier ratios over pixel thresholds on a rendered dataset by [21] (a), our dataset rendered with Unity3D (b) and our real-world captured dataset (c). Aside from HyperDepth [7] and our algorithm, experiments marked with "full" undergo fully supervised training with L1 loss.

pixel-level masks corresponding to areas where the pattern projector does not have enough influence. 15k sequences of four frames each are rendered this way. The test set features a different set of objects and textures and offers 9k frames.

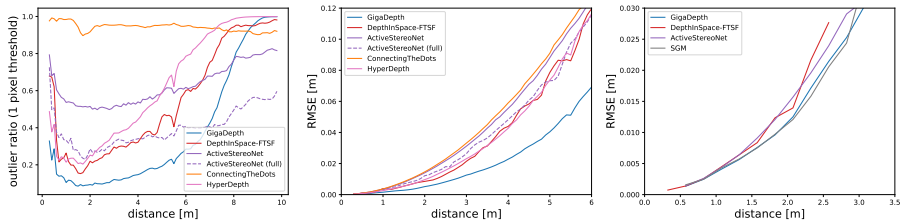
To extract a texture for the pattern, we point the Structure Core as well as a RealSense sensor with disabled projector towards a wall and capture multiple IR frames. While the center of the pattern is covered by the Structure Core itself, the RealSense captures the fringes. In the next steps, center points of speckles for each IR image are extracted, manually matched between RealSense and Structure frames (three points each) and ICP aligned. Finally, a set of textures is created to vary the sharpness of speckles during rendering.

To fine-tune the baseline algorithms for the real-world domain, we capture 967 scenes with four frames each. 241 of these scenes have a second set of images with the dot-projector disabled. This is essential to train the edge detector required by Connecting The Dots [21] and precompute the optical flow required by DepthInSpace [14]. To evaluate the performance, we collect 11 scenes with ground-truth data captured by a Photoneo MotionCam-3D M in scanning mode. With an accuracy stated as $< 0.250mm$ at a distance of $0.65m$ this sensor is adequate for the expected accuracy of our algorithm. Translated to the Structure Core's geometry and provided the alignment between both sensors is similarly accurate, this means we can expect ground-truth disparity with an accuracy of ~ 0.05 pixels.

4.3 Results

The experiments on rendered and real data show that we indeed have a real-time method capable of capturing dense and highly accurate disparity maps. Aside from the increased accuracy and sensitivity to comparable methods we furthermore show that our approach bridges the domain gap much more gracefully.

Rendered Data Evaluating outlier ratios at different thresholds (fig. 3b) and together with $RMSE$ at different distances (figs. 4a and 4b) showcases the ac-



(a) Rendered data (b) RMSE on rendered (c) RMSE on captured data

Fig. 4: Outlier ratio at 1 pixel threshold (a) and RMSE of inliers (b) over distance on our rendered dataset. While the captured dataset is too small to produce a meaningful plot analog to (a) some of the algorithms produce enough inlier estimates to plot the RMSE for depth (c). Aside from HyperDepth [7] and our algorithm, experiments marked with "full" undergo fully supervised training with L1 loss.

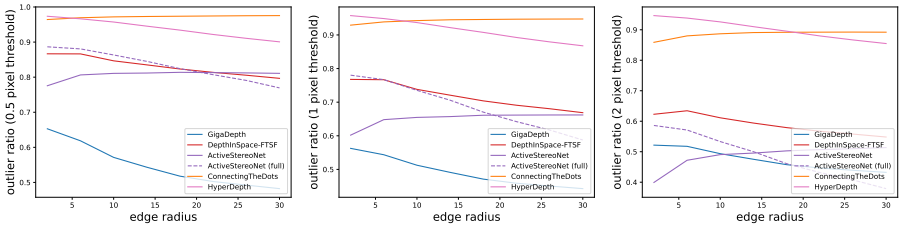
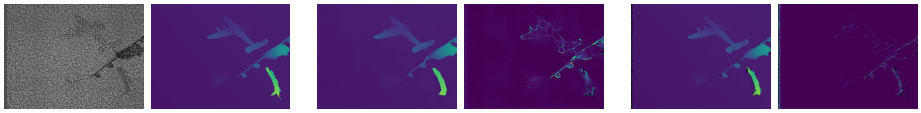


Fig. 5: Outlier ratios on regions around depth discontinuities with our rendered dataset. Evaluated for different outlier thresholds over a growing region around object edges. Note that only estimates by ActiveStereoNet [26] benefit from a proximity to the edges as they often coincide with strong intensity gradients.

curacy as well as sensitivity of each method. To further magnify the focus on the model architectures themselves, we train versions of ActiveStereoNet [26] and Connecting The Dots [21] with full supervision by utilizing the L1 loss.

Presented in figs. 3b, 4a and 4b as well as the qualitative results in fig. 7, it is evident that the branching approaches of HyperDepth [7] and GigaDepth can deliver more precise results but cannot always reach the level of completeness of the CNN-based ActiveStereoNet [26] and DepthInSpace [14]. Plotting the outlier ratios with a one pixel threshold over distances in fig. 4a we can infer that our algorithm has a higher sensitivity towards the dot-pattern than any of the baseline methods. ActiveStereoNet [26], which utilizes the second camera, performs stereo matching and therefore shows better performance at higher distances that are otherwise insufficiently lit by the projector.

To substantiate the claim about our algorithm's robustness in situations of a fragmented pattern, we evaluate its performance in regions around depth discontinuities. After extracting the depth discontinuities from the ground-truth, we increasingly dilate these edges to obtain regions of different proximity. In fig. 5,



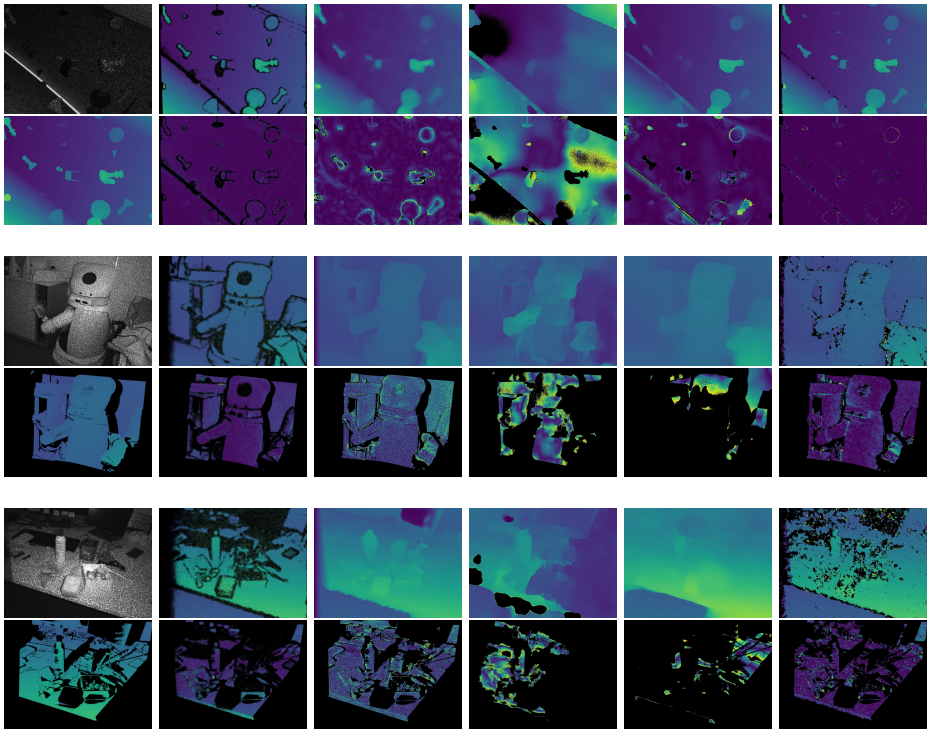
(a) IR + GT (b) Connecting The Dots (c) GigaDepth (ours)

Fig. 6: Disparities (left b, c) of the different algorithms on a dataset based on models from [4]. Color coding is applied for disparity errors (right b, c) of 0 to 5 pixels with outliers (> 5 pixels) being black. (a) shows the input infrared image (IR) and ground-truth (GT).

we plot the outliers in these regions along the a range of radii around edges. It becomes evident that our method exceeds the remaining methods when high pixel accuracy is needed. If accuracy is not of utmost important, ActiveStereoNet [26] can achieve a lower outlier ratio as it actually benefits from the strong intensity gradients that often coincide with object boundaries.

As reflected in figs. 3b, 3c, 4a and 5, it is challenging to achieve acceptable results for the Connecting The Dots [21] algorithm on our dataset. To still include a truthful comparison, we perform a comparison of HyperDepth [7], Connecting The Dots [21] and our algorithm on the dataset provided by Riegler et al. [21] (with a modified test set to include unseen object classes). While the results in fig. 6 and fig. 3a show that Connecting The Dots [21] delivers compelling subpixel accuracy, our approach still vastly outperforms it. Unfortunately, this dataset only offers a small range of distances and does not include important factors such as textures on objects, which limits its expressiveness. DepthInSpace [14], that shares many of the traits and uses a similar dataset as [21], behaves more gracefully when applied to our dataset such that we can offer a fair comparison in our more challenging scenario. We nonetheless refer to our supplementary where we offer a comparison to our method on the dataset native to [14].

Real-world data For our real-world evaluation, we align the data from the Photoneo MotionCam-3D M and the point clouds derived from each algorithm using ICP. Projecting the ground-truth point cloud to the respective camera frames yields the disparity maps we compare against. We plot the outlier ratios for our set of algorithms in fig. 3c and show favorable results compared to all baselines. Only ActiveStereoNet [26] achieves superior outlier ratios above thresholds of $\sim$3 pixels, which we attribute to this method's ability to fall back to its stereo-matching roots when the pattern is absent. Evaluating the $RMSE$ inline with the experiment on artificial data is challenging as not all methods produce enough usable depth samples at the full range of depth. For the remaining methods (HyperDepth [7], DepthInSpace [14] and SGM [10]), we show equivalent to favorable performance. Note that basing a comparison on the depth $RMSE$ leads to a distorted view due to the influence of errors in rectification/calibration and (mis)alignment between the captured frame and ground-truth data. It is advised to focus on the pixel-metrics as they depend less on sensor geometry and

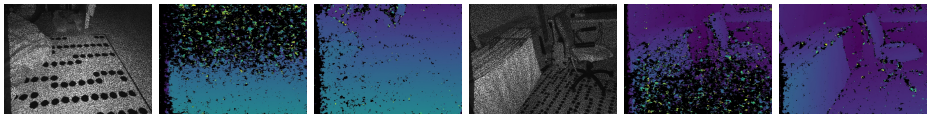


(a) IR + GT (b) HD (c) ASN (d) CTD (e) DIS (f) GD (ours)

Fig. 7: Disparities (top b - f) of different algorithms applied on a scene rendered with Unity3D (rows 1,2) or captured with the Occipital Structure Core (rows 3-6). The ground-truth (GT) is rendered or captured with the Photoneo MotionCam-3D M and compared to other algorithms. Color coding is applied for disparity errors (bottom b - f) of 0-5 pixels with outliers (> 5 pixels) being black. Algorithms are: HyperDepth (b, HD), ActiveStereoNet (c, ASN), Connecting The Dots (d, CTD), DepthInSpace (e, DIS) and GigaDepth (e, GD).

to some extent even allow for cross-sensor comparability. We also include a comparison to HyperDepth [7] when trained on artificial data (marked as XDomain) and a variant trained on SGM [10] to show a fundamental problem: While the perfect ground-truth artificial data enables the algorithm to excel for high precision tasks, this is not necessarily applicable in the real-world, leading to reduced reliability.

A qualitative assessment in fig. 7 shows that GigaDepth delivers notably lower disparity errors compared to the baselines with measurements mostly being omitted at object fringes and pixels that are shadowed from the pattern projector. It nonetheless suffers from the domain transfer as many spots on similar surfaces, that were not a problem on artificial data, now have holes.



(a) IR-input (b) Artifacts (c) Jitter (d) IR-input (e) Artifacts (f) Jitter

Fig. 8: Two frames captured in the same sequence. The captured images (a, d) seem to change their vertical alignment with the projector causing the algorithm to fail in shifting regions (b, e). Applying vertical jitter of four pixels during the training phase makes the network agnostic to this variability (c, f).

4.4 Ablation

Architecture To corroborate our choice of network architecture we benchmark different combinations of feature extractors and regressors on our synthetic dataset and report outlier ratios in table 2. The first set of experiments is based on a relatively powerful and compute intensive UNet that takes an order of magnitude more time to execute than all the other baseline methods ($1s$ compared to $\sim 50ms$). Fully supervising the UNet on our regression task without any additional network does not yield any usable behaviour. The same can be concluded when supplying output layers with per-line weights. Using the network as a backbone for our regression network, however, gives superior performance even to our own backbone, albeit it being at much higher cost. Looking at the outlier ratio for low thresholds ($o(0.1)$) we do not see much improvements but the overall amount of valid pixel ($o(1)$) seems to have increased. We can attribute this to the capacity of the network to incorporate high-level information without sacrificing the ability to encode local features.

The second set of experiments operates with the backbone we tailored for this task and aims at analysing the influence of the MLP tree structure. Varying the amount of output classes between 288 and 2688 we see the return of investment diminishing after 1280 classes. While the runtime is almost unaffected by the class count, the increasing amount of parameters is a cause for concern. Note that the parameter count would rise even more steeply if we would not be increasingly aggressive with our strategy of sharing weights between consecutive output classes in the latent regressor layers. The influence of the depth of the MLPs is shown in three variations (adding/removing one 32 channel latent layer) of the 640 class version showing diminishing returns for MLPs deeper than two layers. Finally we see in a variation (superscript ^{a}) of the 1920 class version, that the lack of the LCN input brings a slight degradation in performance.

Vertical Jitter In fig. 8, we explore the effect of omitting the jittering augmentation during training. It shows shifting regions of failing depth estimation even within short sequences. This is an indication that the sensor geometry and components are not entirely rigid or susceptible to temperature. Randomly shifting the training images by just a few vertical pixel robustifies the algorithm.

Table 2: Different configurations of our architecture tested on synthetic data. We test two backbones, ours as well as a full UNet network, which both dominate execution time with $\sim 70ms$ and $\sim 1s$ respectively (RTX 2070 Max-Q). Regressors are given as $^{c}/_{l}$ with the number of classes c and MLP layers l. All networks take IR + LCN as input (superscript ^{a} omits the LCN). Note that with increasing class count, we more aggressively apply our weight sharing scheme for hidden layers of the regressor stage

B.B.	UNet		Our Backbone									
Reg.	none	lines	$^{1920}/_{2}$	$^{288}/_{2}$	$^{384}/_{2}$	$^{640}/_{1}$	$^{640}/_{2}$	$^{640}/_{3}$	$^{1280}/_{2}$	$^{1920^a}/_{2}$	$^{1920}/_{2}$	$^{2688}/_{2}$
$o(0.1)$	89.55	94.27	**34.01**	51.86	60.49	44.50	44.37	38.49	**34.91**	41.27	38.08	44.59
$o(0.5)$	54.78	72.39	**14.32**	19.88	18.64	19.05	17.53	18.11	**16.97**	17.20	17.20	17.26
$o(1)$	33.41	50.23	**12.74**	17.16	16.29	17.39	16.02	16.62	**15.72**	15.97	15.97	15.93
params	81M	105M	446M	134M	217M	131M	275M	359M	379M	388M	388M	429M

5 Conclusion

This paper introduced an algorithm that outperforms state of the art on extracting depth from structured light-based depth sensors. Benchmarks on artificial as well as real data show superior precision and sensitivity than comparable methods. It is shown that while pure CNN-based methods struggle to deliver high accuracy for these regression tasks, combining a CNN-based backbone with a regressor consisting of weight-adaptive layers can overcome this challenge. These weight-adaptive layers enable us to implement a neural decision tree with small specialized regressors at the leaf nodes. While the comparable HyperDepth [7] follows a branching approach similar to our regression stage, the decision functions on each node are comparably trivial and thus struggle to model the different influences of scene and surface compositions. The focus on dot-patterns and the strategy of keeping the receptive field small allows our large set of small neural networks to specialize on their respective regions within the pattern. Despite the necessity for accurate ground-truth to train the classification part, most easily obtained using artificial data, our approach's resilience to domain shift is demonstrated by the good performance on real-world data.

As most contemporary depth estimation algorithms cope without explicit supervision by using consistency-based loss functions, it would be most desirable to augment our approach with similar mechanisms. Generating meaningful update steps for our tree structure in a semi-supervised setting (by e.g. using principles found in reinforcement learning) would allow for easier domain adaptation.

Acknowledgements The research leading to these results has received funding from EC Horizon 2020 for Research and Innovation under grant agreement No. 101017089, TraceBot and the Austrian Science Foundation (FWF) under grant agreement No. I3969-N30, InDex.

References

1. Altschuler, M.D., Posdamer, J.L., Frieder, G., Altschuler, B.R., Taboada, J.: The numerical stereo camera. In: Three-Dimensional Machine Perception. vol. 0283, pp. 15 – 24. International Society for Optics and Photonics (1981)
2. Bian, J.W., Zhan, H., Wang, N., Li, Z., Zhang, L., Shen, C., Cheng, M.M., Reid, I.: Unsupervised scale-consistent depth learning from video. International Journal of Computer Vision **129**, 2548–2564 (2021)
3. Carrihill, B., Hummel, R.: Experiments with the intensity ratio depth sensor. Computer Vision, Graphics, and Image Processing **32**(3), 337–358 (1985)
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An information-rich 3D model repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
5. Chang, J.R., Chen, Y.S.: Pyramid stereo matching network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5410–5418 (2018)
6. Chen, W., Mirdehghan, P., Fidler, S., Kutulakos, K.N.: Auto-tuning structured light by optical stochastic gradient descent. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5969–5979 (2020)
7. Fanello, S.R., Rhemann, C., Tankovich, V., Kowdle, A., Escolano, S.O., Kim, D., Izadi, S.: HyperDepth: Learning depth from structured light without matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5441–5450 (2016)
8. Godard, C., Aodha, O.M., Firman, M., Brostow, G.: Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3827–3837 (2019)
9. Gupta, M., Nakhate, N.: A geometric perspective on structured light coding. In: Proceedings of the European Conference on Computer Vision. pp. 90–107 (2018)
10. Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 807–814 (2005)
11. Hu, M., Wang, S., Li, B., Ning, S., Fan, L., Gong, X.: PENet: Towards precise and efficient image guided depth completion. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 13656–13662 (2021)
12. Hui, T.W., Tang, X., Loy, C.C.: LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 8981–8989 (2018)
13. Van der Jeught, S., Dirckx, J.J.J.: Deep neural networks for single shot structured light profilometry. Optics Express **27**(12), 17091–17101 (2019)
14. Johari, M., Carta, C., Fleuret, F.: DepthInSpace: Exploitation and fusion of multiple video frames for structured-light depth estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6019–6028 (2021)
15. Khamis, S., Fanello, S.R., Rhemann, C., Kowdle, A., Valentin, J.P.C., Izadi, S.: StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In: Proceedings of the European Conference on Computer Vision. pp. 596–613 (2018)
16. Martinez, M., Stiefelhagen, R.: Kinect unleashed: Getting control over high resolution depth maps. In: Proceedings of the International Conference on Machine Vision Applications. pp. 247–240 (2013)

17. Miangoleh, S.M.H., Dille, S., Mai, L., Paris, S., Aksoy, Y.: Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9680–9689 (2021)
18. Mirdehghan, P., Chen, W., Kutulakos, K.N.: Optimal structured light à la carte. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6248–6257 (2018)
19. Nguyen, H., Wang, Y., Wang, Z.: Single-shot 3D shape reconstruction using structured light and deep convolutional neural networks. Sensors **20**(13) (2020)
20. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12179–12188 (2021)
21. Riegler, G., Liao, Y., Donne, S., Koltun, V., Geiger, A.: Connecting the Dots: Learning representations for active monocular depth estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7616–7625 (2019)
22. Tankovich, V., Hane, C., Zhang, Y., Kowdle, A., Fanello, S., Bouaziz, S.: HITNet: Hierarchical iterative tile refinement network for real-time stereo matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14362–14372 (2021)
23. Van Gansbeke, W., Neven, D., De Brabandere, B., Van Gool, L.: Sparse and noisy LiDAR completion with RGB guidance and uncertainty. In: Proceedings of the International Conference on Machine Vision Applications. pp. 1–6 (2019)
24. Watson, J., Aodha, O.M., Prisacariu, V., Brostow, G., Firman, M.: The temporal opportunist: Self-supervised multi-frame monocular depth. In: Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition. pp. 1164–1174 (2021)
25. Zhang, Y., Funkhouser, T.: Deep depth completion of a single RGB-D image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 175–185 (2018)
26. Zhang, Y., Khamis, S., Rhemann, C., Valentin, J.P.C., Kowdle, A., Tankovich, V., Schoenberg, M., Izadi, S., Funkhouser, T.A., Fanello, S.R.: ActiveStereoNet: End-to-end self-supervised learning for active stereo systems. In: Proceedings of the European Conference on Computer Vision. pp. 802–819 (2018)