Single Image Depth Prediction with Wavelet Decomposition

Michaël Ramamonjisoa[1,*] Michael Firman[2] Jamie Watson[2]
Vincent Lepetit[1] Daniyar Turmukhambetov[2]
[1]LIGM, IMAGINE, Ecole des Ponts, Univ Gustave Eiffel, CNRS [2]Niantic
www.github.com/nianticlabs/wavelet-monodepth

Abstract

We present a novel method for predicting accurate depths from monocular images with high efficiency. This optimal efficiency is achieved by exploiting wavelet decomposition, which is integrated in a fully differentiable encoder-decoder architecture. We demonstrate that we can reconstruct high-fidelity depth maps by predicting sparse wavelet coefficients.

In contrast with previous works, we show that wavelet coefficients can be learned without direct supervision on coefficients. Instead we supervise only the final depth image that is reconstructed through the inverse wavelet transform. We additionally show that wavelet coefficients can be learned in fully self-supervised scenarios, without access to ground-truth depth. Finally, we apply our method to different state-of-the-art monocular depth estimation models, in each case giving similar or better results compared to the original model, while requiring less than half the multiply-adds in the decoder network.

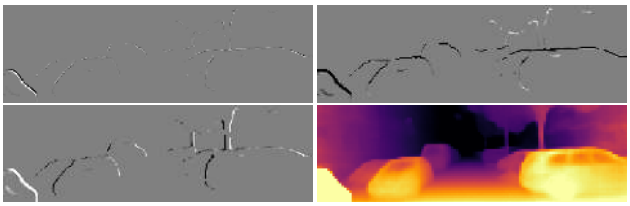
1. Introduction

Single-image depth estimation methods are useful in many real-time applications, for example robotics, autonomous driving and augmented reality. These areas are typically resource-constrained, so efficiency at prediction time is important.

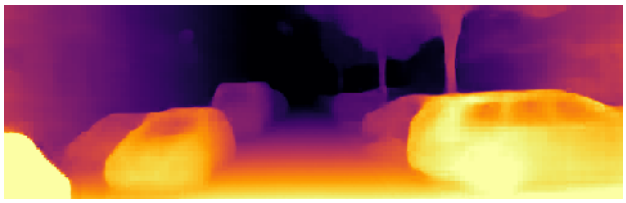
Neural networks which estimate depth from a single image overwhelmingly use U-Net architectures, with skip connections between encoder and decoder layers [55]. Most work on single-image depth prediction has focused on improved depth accuracy, without focusing on efficiency. Those that have cared about efficiency have typically borrowed tricks from the "efficient network" world [30, 56] to make faster depth estimation, with the network using standard convolutions all the way through [65, 51]. All these approaches still use standard neural network components: convolutions, additions, summations and multiplications.

Inspired by sparse representations that can be achieved with wavelet decomposition, we propose an alternative network representation for more efficient depth estimation, us-

(a) Input color image – (320×1024)



(b) Sparse estimation using wavelets. Our network up-samples and refines a 1/16-resolution depth map (bottom-right), by estimating wavelet coefficients only in sparse regions.



(c) Reconstruction of the output depth map using the inverse wavelet transform.

Figure 1: We can represent depth maps more efficiently with wavelets. Here the network takes image (a) as input and outputs a low resolution depth map, together with sparse wavelet coefficients (b). We can reconstruct a high-resolution depth map (c) using the inverse wavelet transform. In our model we *predict* multi-scale wavelet coefficients with an image-to-image network, and we exploit sparseness of the output to save computation.

*Work done during an internship at Niantic.

ing *wavelet decomposition*. We call this system **Wavelet-Monodepth**. We make the observation that depth images of the man-made world are typically made up of many piecewise flat regions, with a few 'jumps' in depth between the flat regions. This structure lends itself well to wavelets. A low-frequency component can represent the overall scene structure, while the 'jumps' can be well captured in high-frequency components. Crucially, the high-frequency components are sparse, which means computation can be focused only in certain areas. This has the effect of saving run-time computation, while still enabling high-quality depths to be estimated.

To the best of our knowledge, we are the first to train a single-image depth estimation network that reconstructs depth by predicting wavelet coefficients. Furthermore, we show that our models can be trained with self-supervised loss on the final depth signal, in contrast to other methods that directly supervise predicted wavelet coefficients.

We evaluate on NYU and KITTI datasets, where we train supervised and self-supervised, respectively. We show that our approach allows us to effectively trade off depth accuracy against runtime computation.

2. Related work

We first give an overview of monocular depth estimation, before looking at works which have made depth estimation more efficient. We then discuss related works which have used wavelets for computer vision tasks, before finally looking at other forms of efficient neural networks.

2.1. Monocular Depth Estimation

Beyond early shape-from-shading methods, most works that estimate depth from a single image have been learning-based. Early works used a Markov random field [57], but more recent works have used deep neural networks. Supervised approaches use image-to-image networks to regress depth maps *e.g.* [12, 11, 38, 13]; however these require ground truth depth data at training time. Self-supervision reduces the requirement of supervised data by using stereo frames [14, 16] or nearby video frames [73] as supervision, exploiting 3D geometry with image reconstruction losses to learn a depth estimator. Focus in this area is typically around improving the depth accuracy scores, *e.g.* by modelling moving objects at training time [3, 5, 70, 19, 54] or by modelling occlusion [19, 18]. While these improvements achieve higher scores with equivalently trained architectures, some works aim for improved depth accuracy *at the expense of efficiency*. For example, by using higher resolution images [47], larger networks [20] or classification instead of regression at the output layer [13].

Efficient depth estimation. A relatively small number of works focus on efficiency specifically for depth. Poggi *et al.* [51] introduce PyDNet, which uses an image pyramid to enable a high receptive field with a small number of parameters. Wofk *et al.* [65] introduce FastDepth, which uses depthwise separable layers and network pruning to achieve efficient depth estimation. An alternative angle on efficient depth estimation is to focus on the training procedure. Several works use knowledge distillation to enable a small depth estimation network to learn some of the knowledge from a larger network *e.g.* [59, 44].

In contrast to these works, our contribution is to change the internal representation of depth within the network itself. We note that our contributions could be used in conjunction with the above efficient architectures or distillation schemes.

2.2. Wavelets in Computer Vision

Wavelet decomposition is an extensively used technique in signal processing, image processing and computer vision. The discrete wavelet transform (DWT) allows a representation of a discrete signal which is more redundant and hence compressible. A notable example is compression of images with JPEG2000 format [62, 60]. Furthermore, wavelet decomposition is also a frequency transform, and can be used for denoising [9, 10, 34]. Wavelet transforms have also recently been combined with Deep Learning to restore images affected by Moiré color artifacts, which occur when RGB sensors are unable to resolve high-frequency details [48, 43]. Li *et al.* [42] show that by substituting pooling operations in neural networks with discrete wavelet transforms it is possible to filter out high-frequency components of the input image during prediction and thus improve noise-robustness in image classification tasks. Super-resolution methods [21, 33, 8] learn to estimate the high-frequency wavelet coefficients of an input low-resolution image to generate high-frequency image through inverse wavelet transform.

Closer to our work, Yang *et al.* [67] use wavelets in a stereo matching network but require supervision of wavelet coefficients while we do not. Similarly, Luo *et al.* [46] replaced the down-sampling and up-sampling operations of UNet-like architectures with DWTs and inverse DWT respectively, and replaced standard skip-connection with high-frequency coefficient skip-connections. However, they do not directly predict wavelet coefficients of depth and as such are unable to exploit the sparse representation of wavelets for efficiency. In contrast with both these works, we focus on efficient depth prediction *from a single image*.

2.3. Efficient Neural Networks

Convolutional Neural Networks (CNNs) [40] have revolutionized the field of computer vision as CNN based methods tend to outperform every other competing methods on regression or classification tasks, if they are provided

enough training data. However, the best performing neural networks contain a large number of parameters and require a large number of floating point operations (FLOPs) at runtime, making deployment to lightweight platforms problematic. Many architectures have been developed to improve the accuracy/speed tradeoff in deep nets. For example, depth-wise separable convolutions [30], inverted residual layers [56], and pointwise group convolutions [72]. An alternative approach though is to train a network before cutting down some of its unnecessary computations.

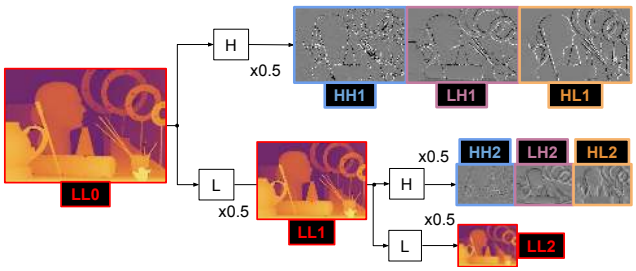
Channel pruning. One line of research is *network pruning* [45, 24, 71], which consists of removing some of the redundant filters in a trained neural network. While this helps reducing the network memory footprint as well as the number of FLOPs necessary for inference, sparsity is typically enforced through regularization terms [64, 25] to compress the network without losing performance. Using such regularisation, however, often requires careful tuning to achieve the desired result [69]. In contrast, our wavelet-based method intrinsically provides sparsity in outputs and intermediate activations, and the wavelet predictions coincide with *edges* in the depth map, knowledge of which has direct applications *e.g.* in augmented reality [53, 29].

While most works focus on classification, channel pruning has also been successfully applied to depth estimation in the aforementioned FastDepth [65], which uses NetAdapt [68] to perform channel pruning.

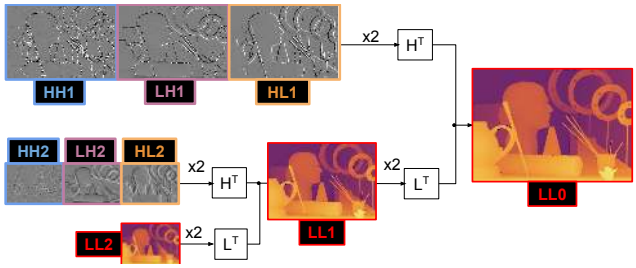
Sparse inference. Another recent work considers spatially sparse inference in image-to-image translation tasks. PointRend [35] treats semantic segmentation as a rendering process, where a high-resolution estimate is obtained from a low-resolution one through a cascade of upsampling and sparse refinement operations. The location of these sparse *rendering* operations is chosen based on an uncertainty measure of the classification method. However, while they demonstrate the efficiency and applicability of their method to classification tasks, their method cannot directly be applied to regression tasks because of the requirement to evaluate an uncertainty heuristic for all pixel locations. In contrast, our method can directly be applied to regression tasks, as *rendering* locations are directly predicted by our model as non-zero-valued high-frequency wavelet coefficients.

3. Method

In this section, we first introduce the basics of 2D wavelet transforms. We chose Haar wavelets [22] due to their simplicity and provided efficiency. Next, we describe how to use the cascade nature of wavelet representations to build our efficient depth estimation architecture, which we call **WaveletMonodepth**. Finally, we discuss the computational benefits of sparse representations.



(a) Forward transform (DWT). This decomposes the high resolution input (left) into multi-scale low resolution outputs.



(b) Inverse transform (IDWT). This reconstructs the original input from estimated low resolution inputs.

Figure 2: Illustration of a two-level wavelet representation of a depth image. The input image LL_0 is passed through a two-level wavelet decomposition (a), to produce a low frequency depth map together with associated wavelets for high frequency detail. The inverse wavelet transform (b) can reconstruct the original image from the wavelet decomposition.

3.1. Haar Wavelet Transform

The Haar wavelet basis is the simplest basis of functions for wavelet decomposition. A discrete wavelet transform (DWT) with Haar wavelets decomposes a 2D image into four coefficient maps: a low-frequency (L) component LL and three high-frequency (H) components LH, HL, HH at half the resolution of the input image. For the remainder of the paper, we refer to the coefficient maps as the output of the **DWT**. The **DWT** is an invertible operation, where its inverse, **IDWT**, converts four coefficient maps into a 2D signal at twice the resolution of the coefficient maps.

The multi-scale and multi-frequency wavelet representation is build by recursively applying **DWT** to the low-frequency coefficient map LL, starting from the input image—see Figure 2(a). Similarly, the multi-scale representation can be recursively inverted to reconstruct a full resolution image (Figure 2(b)). This synthesis operation is the building block of our depth reconstruction method.

3.2. WaveletMonoDepth

Our method, which we call WaveletMonoDepth, is summarized in Figure 3. It builds on a recursive use of **IDWT** operation applied to predicted coefficient maps. Thus, we reconstruct a depth map at the input scale by first predict-

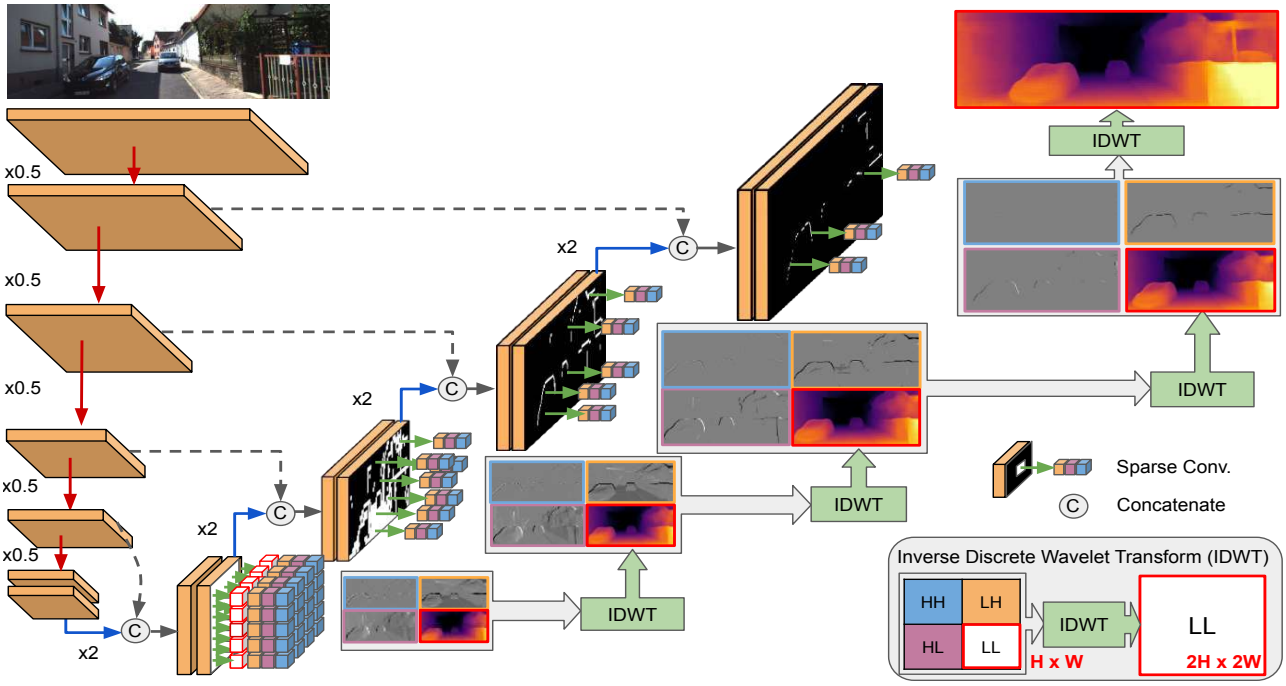


Figure 3: Our method WaveletMonoDepth predicts depth from a single image using wavelets. At each stage in our decoder, we predict sparse wavelet coefficients $\{\mathsf{LH}, \mathsf{HL}, \mathsf{HH}\}$. These capture the high-frequency details of the depth map, *e.g.* occlusion boundaries. These are combined with the low-frequency depth map LL, taken from the previous level in the decoder, and passed through an inverse discrete wavelet transform (IDWT). This generates a new depth map at twice the resolution of LL. This process is continued through the decoder until the original input image resolution is reached. Because the wavelet coefficients are *sparse*, we can save computations; we need only to evaluate each decoder layer at the *non-zero* wavelet locations in the previous level. See Algorithm 1 for more details.

ing a coarse estimate at the bottleneck scale of a UNet-like architecture [55], and iteratively upscale and refine this estimate by predicting high-frequency coefficient maps.

In our network architecture, the coarse depth estimate LL_3 is estimated at 1/16 of the input scale. This depth map is then progressively upscaled and refined using Algorithm 1. A forward pass of our model generates a collection of 5 depth maps LL_s for scales [1/16, 1/8, 1/4, 1/2, 1]. We choose to supervise only the four last scales as in [18]. It is worth noting that the coefficient maps are predicted at scales [1/16, 1/8, 1/4, 1/2], thus removing the need for full-resolution computation.

3.3. Sparse Computations in Decoder

For piecewise flat depth maps, high-frequency coefficient maps have a small number of non-zero values; these are located around depth edges. Hence, for full-resolution depth reconstruction, only some pixel locations need to predict non-zero coefficient map values at each scale. At any scale, we assume that these pixel locations with non-zero values can be determined from high-frequency coefficient maps estimated at the previous scale defined by a mask M described in `GetSparseMask` of Algorithm 1.

The sparsity level achieved by using mask M is

$$\psi = \frac{\sum_{r,c=1,1}^{H,W} M_{r,c}}{HW}, \tag{1}$$

which allows us to remove redundant computation in the decoder layer. Indeed, for a typical $K \times K$ convolution (with a bias term) on a feature tensor of size $H \times W$ that has C_{in} input channels and C_{out} output channels, the number of multiply-add operations is

$$\text{MAC}_{\text{dense}} = HW(C_{\text{in}}K^2 + 1)C_{\text{out}}. \tag{2}$$

With the sparsity level ψ, it would be

$$\text{MAC}_{\text{sparse}} = \psi HW(C_{\text{in}}K^2 + 1)C_{\text{out}}. \tag{3}$$

Note that our sparsification strategy aims to reduce FLOPs by decreasing the number of pixel locations at which we need to compute an output. This approach is orthogonal and complements other approaches such as channel pruning, which instead reduces C_{in} and C_{out}, or separable convolutions. We refer to supplementary material for further details on these.

Considering a quite conservative threshold $\eta = 0.05$ used on high-frequency coefficient maps, the sparse decoder

Algorithm 1: Computing depth with wavelets

Result: Depth map at input scale
Input : Pyramid of feature maps [F_4, F_3, F_2, F_1];
Current scale $s = 3$;
$LL_3 \leftarrow$ `DensePredict` (F_4);
Threshold η ;
Sparse computation mask M = Initialize with 1;
for ($s = 3;\ s >= 0;\ s = s - 1$) {
  $\leftarrow LH_s, HL_s, HH_s \leftarrow$ `SparsePredict` (F_{s+1}; M);
  $LL_{s-1} \leftarrow$ **IDWT**(LL_s, [LH_s, HL_s, HH_s]);
  $\eta_s \leftarrow \eta \cdot (\max(LL_{s-1}) - \min(LL_{s-1}))$;
  $M \leftarrow$ `GetSparseMask` (LH_s, HL_s, HH_s, η_s);
}
procedure `SparsePredict` (*F, M*)
  Input : Feature map F, Sparse mask M
  for (all p s.t. $M[p] == 1$) {
    H[p] = SparseConv3x3(F[p]);
  }
  return H;
procedure `DensePredict` (*F*)
  Input : Feature map F
  Initialize M with ones;
  return `SparsePredict` (*F, M*) ;
procedure `GetSparseMask` (*H, η*)
  Input : High frequency coefficient maps H
  $M = \max(|LH|, |HL|, |HH|) > \eta$;
  $M = \texttt{upsample}_{\times 2}(M)$;
  return M;

computation is about 3× lower in FLOPs compared to standard convolutions at all pixel locations for an image of size 320×1024.

3.4. Self-supervised Training

Our self-supervised losses are as described in [18], which we briefly describe here for completeness. See supplemental material for further details. Given a stereo pair of images (I_L, I_R), we train our network to predict a depth map D_L, pixel-aligned with the left image. We also assume access to the camera intrinsics K, and the relative camera transformation between the images in the stereo pair $T_{R\to L}$. We use the network's current estimate of depth to synthesise an image $I_{R\to L}$, computed as

$$I_{R\to L} = I_R\Big\langle \text{proj}(D_L, T_{R\to L}, K)\Big\rangle, \tag{4}$$

where proj() are the 2D pixel coordinates obtained by projecting the depths D_L into image I_R, and $\langle\rangle$ is the sampling operator. We follow standard practice in training with a photometric reconstruction error pe, so our loss becomes $L_p = pe(I_L, I_{R\to L})$. Following [18, 5] etc., we set pe to a weighted sum of SSIM and L_1 losses.

We also include the depth smoothness loss from [18].

For our experiments which train on monocular and stereo sequences ('MS'), we combine reprojection errors from the three different source images: one frame forward in time, one frame back in time, and the corresponding stereo pair. In this case, we create synthesized images from the monocular sequence using relative poses estimated from a pose network, as described in [18]. In this setting, we use a per-pixel minimum reprojection loss, again following [18].

4. Experiments

Our validation experiments explore the task of training a CNN to predict depth from a single color image, using wavelets as an intermediate representation. Depending on the experiment, we compare against known leading baselines that supplement, and pre- and post-process the stereo pairs used for supervision, and the output depth maps.

4.1. Implementation Details

Datasets We conduct experiments on the KITTI and NYUv2 depth datasets. KITTI [15] consists of 22,600 calibrated stereo video pairs captured by a car driving around a city in Germany. Models are evaluated using the Eigen split [11] using corresponding LiDAR point clouds; see *e.g.* [16] for details. NYUv2 [50] consists of RGBD frames captured with a Kinect sensor. There are 120K raw frames collected by scanning various indoor scenes. As in DenseDepth [2], we use a 50K samples subset of the full dataset where depth is inpainted using Levin *et al.* inpainting method [41]. The NYUv2 evaluation is run on the 654 test frames introduced by Eigen *et al.* [12].

Metrics On the KITTI dataset we compute depth estimation scores based on the standard metrics introduced by Eigen *et al.* [12]: Abs Rel, Sq Rel, RMSE, RMSE$_{log}$, $\delta_1 = \delta < 1.25$, $\delta_2 = \delta < 1.25^2$, and $\delta_3 = \delta < 1.25^3$. We use the same metrics for NYUv2, but we follow standard practice (*e.g.* [13]) in reporting log_{10} instead of RMSE$_{log}$. To evaluate the sharpness of depth maps on NYUv2, we use the metrics introduced by Koch *et al.* [36, 37] and the NYU-OC++ dataset manually annotated by Ramamonjisoa *et al.* [53, 52].

Models To demonstrate the efficiency of our method, we choose two models for experiments on the NYUv2 and KITTI datasets. Both models are implemented using Pytorch and use a compatible implementation of **IDWT** [7].

For KITTI, we choose the weakly-supervised Depth Hints [63] method, which adds Semi Global Matching [27, 28] supervision to the self-supervised Monodepth2 [18], without requiring Lidar depth supervision. At each scale s of the Monodepth2 decoder there is a layer which outputs a one-channel disparity. We replace this layer at each scale

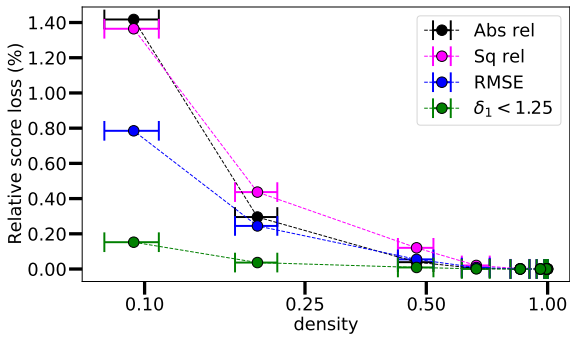


Figure 4: Analysis of performance loss vs density on KITTI. Using our wavelet representation, we can drop up to 90% of the wavelet coefficients while suffering a maximum relative performance loss of less than 1.4%.

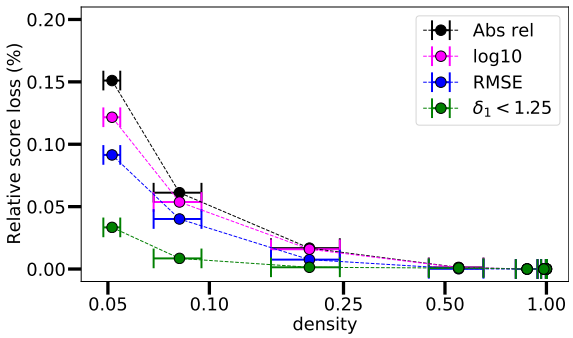


Figure 5: Analysis of performance loss vs density on NYUv2. Using our wavelet representation, we can drop up to 95% of the wavelet coefficients while suffering a maximum relative performance loss of less than 0.2%.

with a 3-channel output layer to predict $\{\mathrm{LH}_s, \mathrm{HL}_s, \mathrm{HH}_s\}$. While our baseline consumes decoder feature maps at scales [1/16, 1/8, 1/4, 1/2, 1], we only need to keep the four scales [1/16, 1/8, 1/4, 1/2], as the **IDWT** outputs disparity at $2\times$ resolution. Both our model and baseline are trained with an Adam optimizer using a learning rate of 10^{-4}, with batch size 12 for 20 epochs. Unless otherwise specified, our experiments are done with Resnet50-based model trained with depth hints loss at 320×1024 resolution.

For NYUv2, we implement a UNet-like baseline similar to DenseDepth [2], and detail its architecture in supplementary material. Similar to our KITTI experiments, we discard the last layer of the decoder as it is not needed, and add one extra layer at each scale to predict the wavelet coefficients. Both our model and baseline are trained using an Adam optimizer with standard parameters, for 20 epochs with batch size 8 and with learning rate 10^{-4}. It is worth noting that DenseDepth predicts outputs at half the input resolution, but evaluates at full resolution after bilinearly upsampling.

4.2. Efficiency vs. Accuracy Trade-off Analysis

In this section, we study the relation between accuracy, sparsity, and efficiency of WaveletMonoDepth. For each set of experiments, we compare our method to an equivalently trained model without wavelets. We first study how wavelets contribute to high-frequency details, then show that they are sparse. Finally, we discuss how we trade off accuracy against efficiency by varying the threshold η used in Algorithm 1 to filter out close-to-zero coefficients.

Wavelets enhance high-frequency details. As mentioned in Section 3.2, the wavelet representation of depth maps allows us to output depth at different resolutions, depending on how many levels of coefficients have been computed. Tables 1 and 2 demonstrate evaluation scores for depth maps produced at different levels of wavelet decomposition on the KITTI and NYUv2 datasets respectively. As can be seen, most of the signal is captured in low-frequency

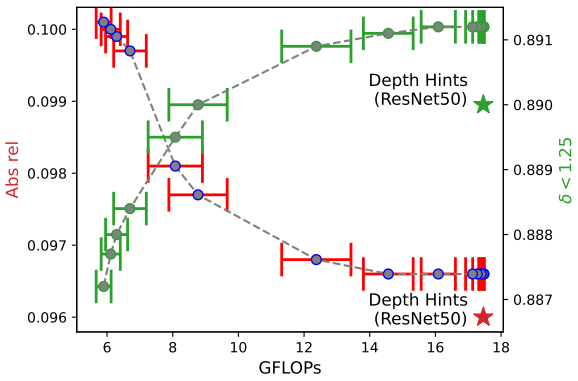


Figure 6: Analysis of performance vs decoder GFLOPs on KITTI. By adjusting the parameter η, we trade off computation in GFLOPs (x-axis) against accuracy (y-axis; Abs Rel in red, and δ_1 in green). We show here that we can reduce the computation by more than half while still remaining on par with our baseline.

estimates of the depth map at the lowest resolution. This confirms previous works observations [11, 4] that a coarse estimate of depth is sufficient to capture the global geometry of the scene. Using more wavelet levels adds more high-frequency details to the depth map, yielding sharper results. Figure 7 shows the sharpening effect of wavelets qualitatively on KITTI and NYUv2 images.

Activated HF	Abs_{Rel}	Sq_{Rel}	R	Rlog	δ_1	δ_2	δ_3
LL only	0.104	0.668	4.415	0.179	0.878	0.962	0.985
[3]	0.097	0.659	4.321	0.177	0.887	0.964	0.984
[3, 2]	0.096	0.679	4.333	0.179	0.890	0.963	0.983
[3, 2, 1]	0.096	0.702	4.366	0.180	0.891	0.963	0.983
[3, 2, 1, 0]	0.097	0.714	4.386	0.181	0.891	0.963	0.983

Table 1: Ablation study on high frequency coefficients on KITTI. While most of the relevant depth information is captured by the low-frequency estimate, predicting higher frequency coefficients increases accuracy. Results are evaluated without post-processing.

Wavelets are sparse. Next, we show that high-frequency coefficients are sparse. As an example, Figure 1(b) shows one low-frequency and three high-frequency coefficient

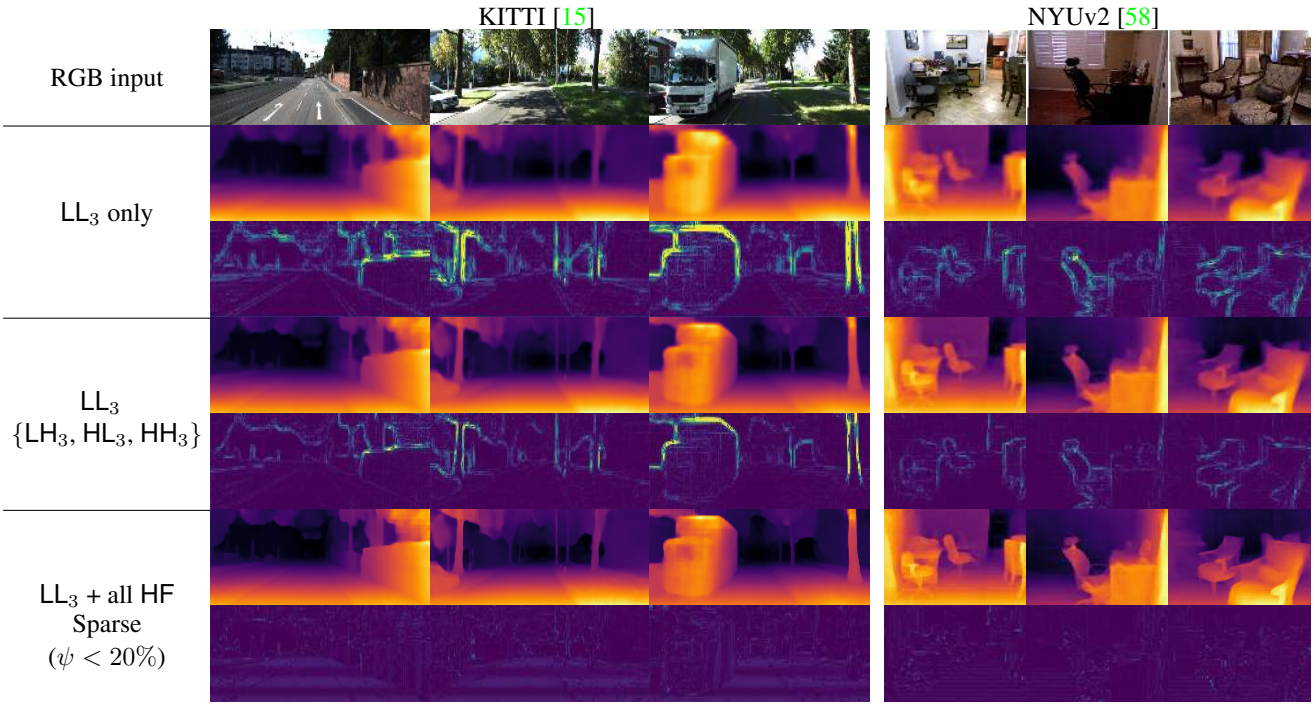


Figure 7: Qualitative results on wavelet representation of depth maps. When using only a subset of wavelet scales, we run the inverse wavelet transform up to the highest scale with those coefficients, then perform a bilinear upsampling up to the full resolution. For each experiment, the bottom line shows the ℓ_1 error map between the considered depth maps and the depth map reconstructed with the complete (dense) set of predicted wavelet coefficients, which shows that wavelets contribute to refining details.

Activated HF	Depth Accuracy						Occ. Boundaries	
	Abs_{Rel}	RMSE	$\log_{10}$	δ_1	δ_2	δ_3	ϵ_{acc}	ϵ_{comp}
LL only	0.1281	0.5549	0.0548	0.8419	0.9674	0.9915	8.3672	9.8552
[3]	0.1264	0.5517	0.0543	0.8446	0.9680	0.9917	3.3945	8.7933
[3, 2]	0.1259	0.5512	0.0542	0.8451	0.9682	0.9917	2.1259	7.6702
[3, 2, 1]	0.1258	0.5515	0.0542	0.8451	0.9681	0.9917	1.8070	7.1073

Table 2: Ablation study on high frequency coefficients on NYU. While most of the relevant depth information is captured by the low-frequency estimate, predicting higher frequency coefficients increases depth and occlusion boundaries accuracy. We evaluate occlusion boundary quality using metrics from Koch *et al.* [36, 37] and the NYU-OC++ dataset manually annotated by Ramamonjisoa *et al.* [53, 52].

maps for a given depth map. We observe that the high-frequency maps have non-zero values near depth edges. More wavelet predictions can be found in supplementary. As depth edges are sparse, high-frequency coefficients at only a few pixel locations are necessary to produce high-accuracy depth maps.

Trading off accuracy against efficiency using sparsity. After training our network with standard convolutions, these are replaced with sparse ones as in Figure 3 and Algorithm 1. Varying the threshold value η allows us to vary the sparsity level ψ in Equation (3), and consequently to trade off accuracy against complexity. Because wavelets are sparse, we can compute them only at a very small number of pixel locations and suffer a minimal loss in depth accuracy. Figures 4 and 5 show relative score changes with varying sparsity threshold on KITTI and NYU datasets respectively. Note that a fixed value of η produces different sparsity levels depending on the content of an image, so we also plot standard deviation of sparsity levels for each η value. Figure 4 indicates that computing the wavelet coefficients at only 10 percent of pixel locations results in a relative loss in scores of less than 1.4% for KITTI images. Similarly, Figure 5 shows that we can compute wavelet coefficients at only 5 percent of pixel locations while suffering a loss in scores of less than 0.20% for NYU images.

Finally, we demonstrate how sparsity of high-frequency coefficient maps can be exploited for efficiency gains in the decoder. Figure 6 shows Abs Rel and δ_1 scores for varying η used during prediction. As can be seen, the score change is minimal when using half multiply-add operations in the decoder and the performance is comparable to SOTA methods using only a third of multiply-add operations. Note that biggest efficiency gains are obtained at higher resolution, as sparsity increases with resolution.

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
[18]	Monodepth2 Resnet18	✓	S	192 × 640	0.108	0.842	4.891	0.207	0.866	0.949	0.976
	WaveletMonodepth Resnet18	✓	S	192 × 640	0.110	0.876	4.916	0.206	0.864	0.950	0.976
	Monodepth2 Resnet50	✓	S	192 × 640	0.108	0.802	4.577	0.185	0.886	0.963	0.983
	WaveletMonodepth Resnet50	✓	S	192 × 640	0.106	0.824	4.824	0.205	0.870	0.949	0.975
[63]	Depth Hints	✓	S_{SGM}	192 × 640	0.106	0.780	4.695	0.193	0.875	0.958	0.980
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.106	0.813	4.693	0.193	0.876	0.957	0.980
	Depth Hints Resnet50	✓	S_{SGM}	192 × 640	0.102	0.762	4.602	0.189	0.880	0.960	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.105	0.813	4.625	0.191	0.879	0.959	0.981
[18]	Monodepth2 Resnet18	✓	MS	192 × 640	0.104	0.786	4.687	0.194	0.876	0.958	0.980
	WaveletMonodepth Resnet18	✓	MS	192 × 640	0.109	0.814	4.808	0.198	0.868	0.955	0.980
[63]	Depth Hints	✓	MS + S_{SGM}	192 × 640	0.105	0.769	4.627	0.189	0.875	0.959	0.982
	WaveletMonodepth Resnet18	✓	MS + S_{SGM}	192 × 640	0.110	0.840	4.741	0.195	0.868	0.956	0.981
[18]	Monodepth2 Resnet18	✓	S	320 × 1024	0.105	0.822	4.692	0.199	0.876	0.954	0.977
	WaveletMonodepth Resnet18	✓	S	320 × 1024	0.105	0.797	4.732	0.203	0.869	0.952	0.977
[63]	Depth Hints	✓	S_{SGM}	320 × 1024	0.099	0.723	4.445	0.187	0.886	0.961	0.982
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.102	0.739	4.452	0.188	0.883	0.960	0.981
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.096**	**0.710**	4.393	0.185	0.890	**0.962**	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	0.097	0.718	**4.387**	**0.184**	**0.891**	**0.962**	**0.982**
	WaveletMonodepth Resnet50 ($\eta = 0.05$)	✓	S_{SGM}	320 × 1024	0.100	0.726	4.444	0.186	0.888	**0.962**	**0.982**

Table 3: Quantitative results on KITTI. We compare our method to our baselines on KITTI [15], using the Eigen split. The *Data* column indicates the training data modality: S is for self-supervised training on stereo images, MS is for models trained with both monocular (forward and backward frames) and stereo data and S_{SGM} refers to the extra stereo ground truth which was used in [63].

Method	H × W	Abs Rel	RMSE	log_{10}	δ<1.25	δ<1.25^2	δ<1.25^3	ϵ_{acc}	ϵ_{comp}
DenseNet baseline	480 × 640	0.1277	**0.5479**	**0.0539**	0.8430	**0.9681**	**0.9917**	**1.7170**	**7.0638**
WaveletMonodepth (last scale sup.)	480 × 640	0.1280	0.5589	0.0546	0.8436	0.9658	0.9908	1.7678	7.1433
WaveletMonodepth	480 × 640	**0.1258**	0.5515	0.0542	**0.8451**	**0.9681**	**0.9917**	1.8070	7.1073
WaveletMonodepth ($\eta = 0.04$)	480 × 640	0.1259	0.5517	0.0543	0.8450	**0.9681**	**0.9917**	1.8790	7.0746

Table 4: Quantitative results on NYUv2 [58] We compare our DenseDepth [2]-inspired baseline to our implementation with wavelets and with sparsity. All results are evaluated in the Eigen center crop, without post-processing. As in DenseDepth, our network outputs a 240 × 320 depth map which is then upsampled for evaluation.

4.3. KITTI results

We summarize our results on the KITTI dataset in Table 3. Here we show that our method, which simply replaces depth or disparity predictions with wavelet predictions, can be applied to a wide range of single image depth estimation models and losses. In each section of the table, the off-the-shelf model numbers are reported, together with numbers from a model trained with our wavelet formulation. For example, we demonstrate that wavelets can be used in self-supervised depth estimation frameworks such as Monodepth2 [18], as well as its weakly-supervised extension Depth Hints [63]. We note that we achieve our best results when using Depth Hints and high-resolution input images. This is not surprising, as supervision from SGM should give better scores, but more importantly using high resolution inputs and outputs allows for more sparsification, as edge pixels become sparser as resolution grows. Importantly, we show overall that replacing fully convolutional layers with wavelets gives models with comparable performance to the off-the-shelf, non-wavelet baselines. We show qualitative results from KITTI in Figure 7 (left).

4.4. NYUv2 results

Scores on NYUv2 are shown in Table 4. Our method performs on par with our baseline, which demonstrates that it is possible to estimate accurate depth and sparse wavelets without directly supervising the wavelet coefficients, in contrast with [67]. In Table 4, we show that supervising depth only at the last scale performs on par with our network supervised at all scales, which shows that a full multi-scale wavelet reconstruction network can be trained end-to-end. Qualitative results from NYUv2 are shown in Figure 7 (right).

5. Conclusion

In this work we combine wavelet representation with deep learning for a single-image depth prediction task. We demonstrate that a neural network can learn to predict wavelet coefficient maps through supervision of the reconstructed depth map with existing losses. Our experiments using KITTI and NYUv2 datasets show that we can achieve scores comparable to SOTA models using similar encoder-decoder neural network architectures to the baseline models, but with wavelet representations.

We also analyze sparsity of wavelet coefficients and show that sparsified wavelet coefficient maps can generate high-quality depth maps. Finally, we exploit this sparsity to reduce multiply-add operations in the decoder network by at least a factor of 2.

Acknowledgements

We would like to thank Aron Monszpart for helping us set up cloud experiments, and our reviewers for their useful suggestions.

References

[1] Filippo Aleotti, Giulio Zaccaroni, Luca Bartolomei, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 2021.

[2] Ibraheem Alhashim and Peter Wonka. High Quality Monocular Depth Estimation via Transfer Learning. *arXiv:1812.11941*, 2018.

[3] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *NeurIPS*, 2019.

[4] Xiaotian Chen, Xuejin Chen, and Zheng-Jun Zha. Structure-aware residual pyramid network for monocular depth estimation. 2019.

[5] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[7] Fergal Cotter. Uses of complex wavelets in deep convolutional neural networks (doctoral thesis). *https://doi.org/10.17863/CAM.53748*, Chapter 3, 2019.

[8] Xin Deng, Ren Yang, Mai Xu, and Pier Luigi Dragotti. Wavelet domain style transfer for an effective perception-distortion tradeoff in single image super-resolution. In *ICCV*, 2019.

[9] David L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[10] David L Donoho and Iain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 09 1994.

[11] David Eigen and Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *ICCV*, 2015.

[12] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In *NeurIPS*, pages 2366–2374, 2014.

[13] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[14] Ravi Garg, Vijay Kumar BG, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[16] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[17] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017.

[18] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[19] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *ICCV*, 2019.

[20] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *CVPR*, 2020.

[21] Tiantong Guo, Hojjat Seyed Mousavi, Tiep Huu Vu, and Vishal Monga. Deep wavelet prediction for image super-resolution. In *CVPR Workshops*, 2017.

[22] Alfréd Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69:331–371, 1910.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[24] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, Oct 2017.

[25] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *CVPR*, 2017.

[26] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[27] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, 2005.

[28] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *PAMI*, 2007.

[29] Aleksander Holynski and Johannes Kopf. Fast depth densification for occlusion-aware augmented reality. *ACM Transactions on Graphics (TOG)*, 2018.

[30] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[31] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[32] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *ICCV*, 2017.

[34] Eunhee Kang, Won Chang, Jaejun Yoo, and Jong Chul Ye. Deep convolutional framelet denosing for low-dose CT via wavelet residual network. *IEEE Transactions on Medical Imaging*, 37(6):1358–1369, 2018.

[35] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020.

[36] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Körner. Evaluation of CNN-Based Single-Image Depth Estimation Methods. In *ECCV*, 2018.

[37] Tobias Koch, Lukas Liebel, Marco Körner, and Friedrich Fraundorfer. Comparison of monocular depth estimation methods using geometrically relevant metrics on the ibims-1 dataset. *CVIU*, 2020.

[38] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. DepthNet: A recurrent neural network architecture for monocular depth prediction. In *CVPR Workshops*, 2018.

[39] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *3DV*, 2016.

[40] Yann LeCun and Yoshua Bengio. Convolutional Networks for Images, Speech, and Time Series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

[41] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization Using Optimization. In *ACM Transactions on Graphics*, 2004.

[42] Qiufu Li, Linlin Shen, Sheng Guo, and Zhihui Lai. Wavelet integrated cnns for noise-robust image classification. In *CVPR*, June 2020.

[43] Lin Liu, Jianzhuang Liu, Shanxin Yuan, Gregory Slabaugh, Ales Leonardis, Wengang Zhou, and Qi Tian. Wavelet-based dual-branch network for image demoireing. In *ECCV*, 2020.

[44] Yifan Liu, Changyong Shun, Jingdong Wang, and Chunhua Shen. Structured knowledge distillation for dense prediction. *PAMI*, 2020.

[45] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[46] Chenchi Luo, Yingmao Li, Kaimo Lin, George Chen, Seok-Jun Lee, Jihwan Choi, Youngjun Francis Yoo, and Michael O. Polley. Wavelet synthesis net for disparity estimation to synthesize dslr calibre bokeh effect on smartphones. In *CVPR*, 2020.

[47] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding. *PAMI*, 2019.

[48] Xiaotong Luo, Jiangtao Zhang, Ming Hong, Yanyun Qu, Yuan Xie, and Cuihua Li. Deep wavelet network with domain adaptation for single image demoireing. In *CVPR Workshops*, 2020.

[49] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[50] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[51] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *IROS*, 2018.

[52] Michael Ramamonjisoa, Yuming Du, and Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *CVPR*, 2020.

[53] Michael Ramamonjisoa and Vincent Lepetit. SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. In *ICCV Workshop*, 2019.

[54] Anurag Ranjan, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, 2019.

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[56] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[57] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3D: Learning 3D Scene Structure from a Single Still Image. *IEEE TPAMI*, 2009.

[58] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012.

[59] Andrew Spek, Thanuja Dharmasiri, and Tom Drummond. CReaM: Condensed real-time models for depth prediction using convolutional neural networks. In *IROS*, 2018.

[60] David Taubman and Michael Marcellin. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated, 2013.

[61] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *3DV*, 2017.

[62] Michael Unser and Thierry Blu. Mathematical properties of the jpeg2000 wavelet filters. *IEEE Transactions on Image Processing*, 12(9):1080–1090, 2003.

[63] Jamie Watson, Michael Firman, Gabriel J. Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[64] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NeurIPS*, 2016.

[65] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *ICRA*, 2019.

[66] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[67] Menglong Yang, Fangrui Wu, and Wei Li. Waveletstereo: Learning wavelet coefficients of disparity map in stereo matching. In *CVPR*, June 2020.

[68] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *ECCV*, 2018.

[69] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv:1802.00124*, 2018.

[70] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018.

[71] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *ICLR*, 2019.

[72] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[73] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

Supplementary Material

6. Network Architectures and Losses

6.1. On direct supervision of wavelet coefficients

The previous work WaveletStereo [67] supervises its wavelet based stereo matching method with ground truth wavelet coefficients at the different levels of the decomposition. However, wavelets can only reliably be supervised when ground truth depth -or disparity- is provided and when it does not contain missing values or high-frequency noise, as they show on the synthetic SceneFlow [49] dataset. The sparsity of ground truth data in the KITTI dataset especially around edges makes it impossible to estimate reliably ground truth wavelet coefficients. On NYUv2, the noise in depth maps is also an issue for direct supervision of wavelets, e.g. with creases in the layout or inaccurate depth edges. This noise also prohibits the use of Semi Global Matching ground truth for wavelet coefficient supervision.

As we show in our work, supervising the network on wavelet reconstructions allows us to ignore missing values and be robust to noisy labels.

6.2. Experiments on KITTI

Architecture The architecture we use for our experiments is a modification of the architecture used in [18], as described in the main paper. In Table 5, we set out our decoder architecture in detail.

Self-supervised losses Our self-supervised losses are as described in [18], which we repeat here for completeness. Given a stereo pair of images (I_L, I_R), we train our network to predict a depth map D_L, pixel-aligned with the left image. We also assume access to the camera intrinsics K, and the relative camera transformation between the images in the stereo pair $T_{R\to L}$. We use the network's current estimate of depth to synthesise an image $I_{R\to L}$, computed as

$$I_{R\to L} = I_R\Big\langle \text{proj}(D_L, T_{R\to L}, K)\Big\rangle, \tag{5}$$

where proj() are the 2D pixel coordinates obtained by projecting the depths D_L into image I_R, and $\langle\rangle$ is the sampling operator. We follow standard practice in training the model under a photometric reconstruction error pe, so our loss becomes

$$L_p = pe(I_L, I_{R\to L}). \tag{6}$$

Following [18, 5] etc. we use a weighted sum of SSIM and L1 losses

$$pe(I_a, I_b) = \alpha\frac{1-\text{SSIM}(I_a, I_b)}{2} + (1-\alpha)\|I_a - I_b\|_1,$$

where $\alpha = 0.85$. We additionally follow [18] in using the smoothness loss:

$$L_s = |\partial_x d_L^*|\, e^{-|\partial_x I_L|} + |\partial_y d_L^*|\, e^{-|\partial_y I_L|}, \tag{7}$$

where $d_L^* = d_L/\overline{d_L}$ is the mean-normalized inverse depth for image I_L.

When we train on monocular and stereo sequences ('MS'), we again follow [18] — see our main paper for an overview, and [18] for full details.

Depth Hints loss When we train with depth hints, we use the proxy loss from [63], which we recap here. For stereo training pairs, we compute a proxy depth map $\tilde{D}_L$ using semi-global matching [27], an off-the-shelf stereo matching algorithm. We use this to create a second synthesized image

$$\tilde{I}_{R\to L} = I_R\Big\langle \text{proj}(\tilde{D}_L, T_{R\to L}, K)\Big\rangle, \tag{8}$$

We decide whether or not to apply a supervised loss using $\tilde{D}_L$ as ground truth on a *per-pixel* basis. We only add this supervised loss for pixels where $pe(I_L, \tilde{I}_{R\to L})$ is lower $pe(I_L, I_{R\to L})$. The supervised loss term we use is $\log L_1$, following [63]. For experiments where Depth Hints are used for training, we disable the smoothness loss term.

Additional experiments We additionally tried training using edge-aware sparsity constraints that penalize non-zero coefficients at non-edge regions, by replacing depth gradients with wavelets coefficients in Monodepth's [17] *disparity smoothness loss*, which unfortunately made training unstable. We also tried to supervise wavelet coefficients using distillation [26, 1] from a teacher depth network, which resulted in lower performances.

6.3. Experiments on NYUv2

Architecture We adapted our architecture from the PyTorch implementation of DenseDepth [2]. Our implementation uses a DenseNet161 encoder instead of a DenseNet169, and a standard decoder with up-convolutions. We first design a baseline that does not use wavelets, using the architecture detailed in Table 7. Our wavelet adaptation of that baseline is then detailed in Table 8. For experiments reported in the main paper, we follow the DenseDepth strategy and predict outputs at half the input resolution. Hence, the last level of the depth decoder in Table 8 is discarded. For experiments using a light-weight decoder discussed later in Section 9.4, which predicts 224×224 depth maps given a 224×224 input image, we keep all four levels of wavelet decomposition.

Supervised losses For our NYU results in the main paper, we supervise depth using an L1 loss and SSIM:

$$L_D(y, y^*) = \lambda_1 \ell_1(y, y^*) \tag{9}$$

*Work done during an internship at Niantic

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	256	32	econv5	ELU [6]
Level 3 coefficients predictions						
iconv4	3	1	256	16	↑upconv5, econv4	ELU
disp4	3	1	1	16	iconv4	Sigmoid
wave4	3	1	3	16	iconv4	Sigmoid
upconv4	3	1	128	16	iconv4	ELU
IDWT$_3$	-	-	1	8	**disp4**, **wave4**	-
Level 2 coefficients predictions						
iconv3	3	1	128	8	↑upconv4, econv3	ELU
wave3	3	1	3	8	iconv3	Sigmoid
upconv3	3	1	64	8	iconv3	ELU
IDWT$_2$	-	-	1	8	**IDWT**$_3$, **wave3**	-
Level 1 coefficients predictions						
iconv2	3	1	64	4	↑upconv3, econv2	ELU
wave2	3	1	3	4	iconv2	Sigmoid
upconv2	3	1	32	4	iconv2	ELU
IDWT$_1$	-	-	1	8	**IDWT**$_2$, **wave2**	-
Level 0 coefficients predictions						
iconv1	3	1	32	2	↑upconv2, econv1	ELU
wave1	3	1	3	2	iconv1	Sigmoid
IDWT$_0$	-	-	-	1	**IDWT**$_1$, **wave1**	-

Table 5: Our decoder network architecture for experiments on the KITTI [15] dataset using ResNet backbone Here **k** is the kernel size, **s** the stride, **chns** the number of output channels for each layer, **res** is the downscaling factor for each layer relative to the input image, and **input** corresponds to the input of each layer where ↑ is a $2\times$ nearest-neighbor upsampling of the layer. **disp4** is used produce the low-resolution estimate LL_3, while **waveJ** is used to decode $\{\mathsf{LH}_\mathbf{J}, \mathsf{HL}_\mathbf{J}, \mathsf{HH}_\mathbf{J}\}$ at level **J**. **disp4** and **waveJ** are convolution blocks detailed in Table 6.

disp4 Layer						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
disp4(1)	1	1	**chns**(iconv5) / 4	16	iconv5	LeakyReLU(0.1) [66]
disp4(2)	3	1	1	16	disp4-1	Sigmoid

Wavelet Decoding Layer - waveJ						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
wave**J**(1+)	1	1	**chns**(iconv[**J**+1])	2^J	iconv[**J**+1]	LeakyReLU(0.1)
wave**J**(2+)	3	1	3	2^J	wave**J**(1+)	Sigmoid
wave**J**(1-)	1	1	**chns**(iconv[**J**+1])	2^J	iconv[**J**+1]	LeakyReLU(0.1)
wave**J**(2-)	3	1	3	2^J	wave**J**(1-)	Sigmoid
substract	1	1	3	2^J	wave**J**(2+), wave**J**(2-)	Linear

Table 6: Architecture of our wavelet decoding layer used for KITTI experiments J denotes the level of the decoder. **disp4** is used produce the low-resolution estimate LL_3, while **waveJ** is used to decode $\{\mathsf{LH}_\mathbf{J}, \mathsf{HL}_\mathbf{J}, \mathsf{HH}_\mathbf{J}\}$.

where y and y^* are respectively predicted and ground truth depth and $\lambda_1 = 0.1$. Similar to [53, 52], we clamp depth between 0.4 and 10 meters.

7. Scores on Improved KITTI Ground Truth

We report results on the improved KITTI ground truth [61] in Table 9. As we saw in the main paper, our method is competitive on scores with non-wavelets baselines, but as we have shown our wavelet decomposition enables more efficient predictions.

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	1104	32	econv5	Linear
iconv4	3	1	552	16	↑upconv5, econv4	LeakyReLU(0.2)
iconv3	3	1	276	8	↑iconv4, econv3	LeakyReLU(0.2)
iconv2	3	1	138	4	↑iconv3, econv2	LeakyReLU(0.2)
iconv1	3	1	69	2	↑iconv2, econv1	LeakyReLU(0.2)
outconv0	1	1	1	2	iconv1	Linear

Table 7: Architecture of our DenseNet baseline decoder for experiments on the NYUv2 [58] dataset Note that as in DenseDepth [2] we produce a depth map at half-resolution. Table adapted from [18].

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	1104	32	econv5	Linear
Level 3 coefficients predictions						
iconv4	3	1	552	16	↑upconv5, econv4	LeakyReLU(0.2)
disp4	1	1	1	16	upconv5	Linear
wave4	3	1	3	16	upconv5	Linear
IDWT$_3$	-	-	1	8	disp4, wave4	-
Level 2 coefficients predictions						
iconv3	3	1	276	8	↑iconv4, econv3	LeakyReLU(0.2)
wave3	3	1	3	8	iconv3	Linear
IDWT$_2$	-	-	1	4	**IDWT**$_3$, wave3	-
Level 1 coefficients predictions						
iconv2	3	1	138	4	↑iconv2, econv2	LeakyReLU(0.2)
wave2	3	1	3	4	iconv2	Linear
IDWT$_1$	-	-	1	2	**IDWT**$_2$, wave2	-

Table 8: Our decoder network architecture for experiments on the NYUv2 [58] dataset Note that since like in DenseDepth [2] we produce a depth map at half-resolution, we only need to predict wavelet coefficients until quarter-resolution. Table adapted from [18].

8. Qualitative Results

In this section, we show qualitative results of our method.

In Figures 10-11-12 and Figures 13-14-15 we first show our sparse prediction process with corresponding sparse wavelets and masks, on the NYUv2 and KITTI datasets respectively. While we only need to compute wavelet coefficients in less than 10% of pixel locations in the decoding process, we show that our wavelets efficiently retain relevant information. Furthermore, we show that wavelets efficiently detect depth edges and their orientation. Therefore, future work could make efficient use of our wavelet based depth estimation method for occlusion boundary detection.

In Figure 8, we show comparative results between our baseline Depth Hints [63] and our wavelet based method.

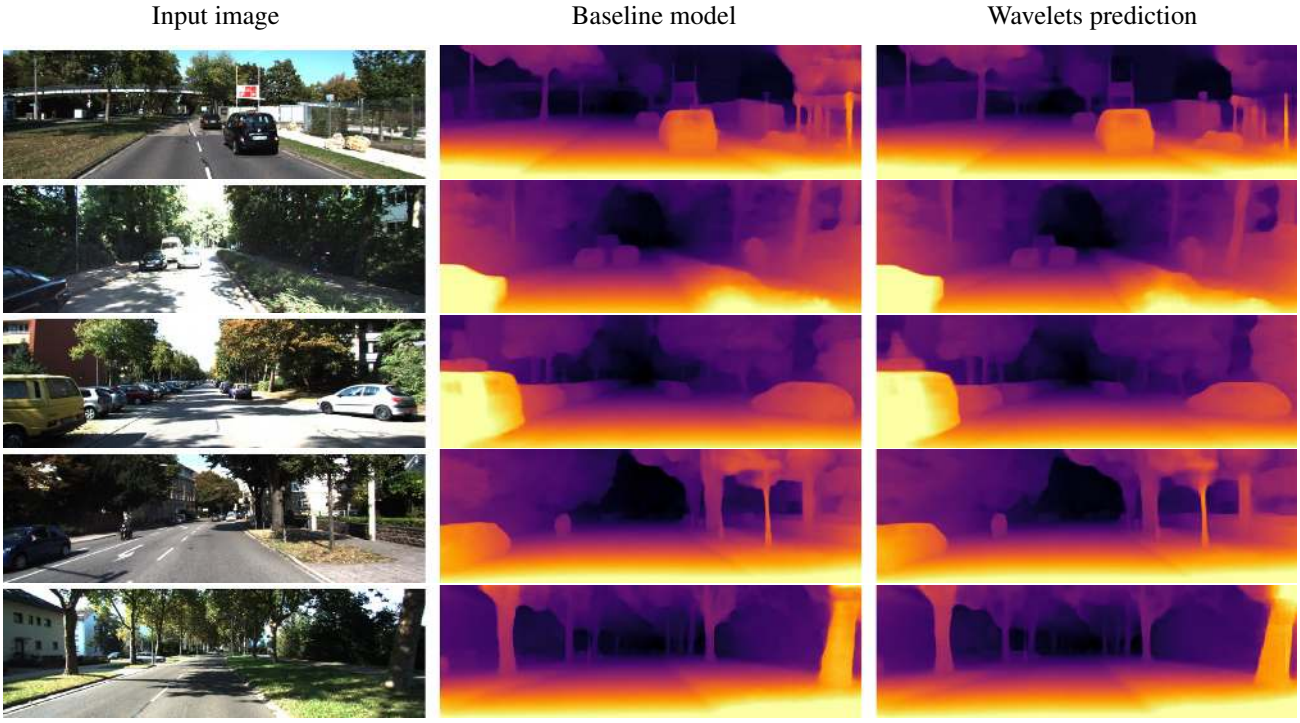


Figure 8: Comparing wavelet predictions to a baseline model on the KITTI dataset. On the left we show the input image, and in the middle column we show the prediction from an off-the-shelf Depth Hints ResNet 50 model [63]. On the right we show an equivalently trained ResNet 50 model, but with our wavelets in the decoder. We see that our predictions retain the high quality of the baseline predictions, but are more efficient to predict.

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
[18]	Monodepth2 Resnet18	✓	S	192 × 640	0.079	0.512	3.721	0.131	0.924	0.982	0.994
	WaveletMonodepth Resnet18	✓	S	192 × 640	0.084	0.523	3.807	0.137	0.914	0.980	0.994
	WaveletMonodepth Resnet50	✓	S	192 × 640	0.081	0.477	3.658	0.133	0.920	0.981	0.994
[63]	Depth Hints	✓	S_{SGM}	192 × 640	0.085	0.487	3.670	0.131	0.917	0.983	0.996
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.083	0.476	3.635	0.129	0.920	0.983	0.995
	Depth Hints Resnet50	✓	S_{SGM}	192 × 640	0.081	0.432	3.510	0.124	0.924	0.985	0.996
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.081	0.449	3.509	0.125	0.923	0.986	0.996
[18]	Monodepth2 Resnet18	✓	MS	192 × 640	0.084	0.494	3.739	0.132	0.918	0.983	0.995
	WaveletMonodepth Resnet18	✓	MS	192 × 640	0.085	0.497	3.804	0.134	0.912	0.982	0.995
[63]	Depth Hints	✓	MS + S_{SGM}	192 × 640	0.087	0.526	3.776	0.133	0.915	0.982	0.995
	WaveletMonodepth Resnet18	✓	MS + S_{SGM}	192 × 640	0.086	0.497	3.699	0.131	0.914	0.983	0.996
[18]	Monodepth2 Resnet18	✓	S	320 × 1024	0.082	0.497	3.637	0.132	0.924	0.982	0.994
	WaveletMonodepth Resnet18	✓	S	320 × 1024	0.080	0.443	3.544	0.130	0.919	0.983	0.995
	WaveletMonodepth Resnet50	✓	S	320 × 1024	0.076	0.413	3.434	0.126	0.926	0.984	0.995
[63]	Depth Hints	✓	S_{SGM}	320 × 1024	0.077	0.404	3.345	0.119	0.930	0.988	0.997
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.078	0.397	3.316	0.121	0.928	0.987	0.997
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.074**	0.363	3.198	**0.114**	**0.936**	**0.989**	**0.997**
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	**0.074**	**0.357**	**3.170**	**0.114**	**0.936**	**0.989**	**0.997**

Table 9: Quantitative results on the improved KITTI benchmark. We compare our method to our baselines on the KITTI [15] improved dataset introduced by [61], using the Eigen split. *Data column* (data source used for training): S is for self-supervised training on stereo images, MS is for models trained with both M (forward and backward frames) and S data and S_{SGM} refers to the extra stereo ground truth which was used in [63].

9. Exploring Other Efficiency Tracks

Our paper mainly explores computation reduction in the decoder of a UNet-like architecture. However, this direction is orthogonal and complementary with all other complexity reduction lines of research.

Our approach is for example complementary with the FastDepth approach, which consists in reducing the overall complexity of a depth estimation network by compressing it in many dimensions such as (1) the encoder, (2) the decoder (3) the input resolution. They argue that the deep network introduced by Laina *et al.* [39] suffers from high complexity, while it could largely be reduced. Here we present a set of experiments we conducted to explore these different aspects of complexity reduction.

9.1. Experiment with a light-weight MobileNetv2 encoder

First, we replace the costly ResNet [23] or DenseNet [32, 31] backbone encoders with the efficient MobileNetv2 [56]. Indeed, in contrast with FastDepth, in the main paper, we report results using large encoder models (Resnet18/50 or Densenet161). Although this helps achieving better scores, we show in Table 10 and Table 11 that we can reach close to state-of-the-art results even with a small encoder such as MobileNetv2.

9.2. Separable convolutions

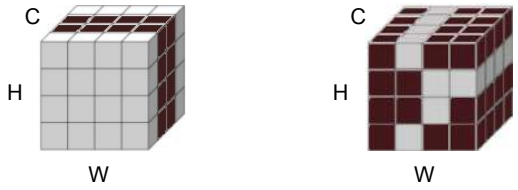
Secondly, FastDepth also shows that separable convolutions in their "NNConv" decoder provides the best score-efficiency trade-off. Since this approach is orthogonal to our sparsification method, it therefore complements our method and can be used to improve efficiency. Interestingly, we show in Table 11 that replacing sparse convolutions with sparse-depthwise separable convolutions works on par with standard convolutions. This can be explained by the fact that **IDWT** is also a separable operation, and therefore efficiently combines with depthwise separable convolutions.

9.3. Channel pruning

A popular approach to complexity and memory footprint reduction is channel pruning, which aims at removing some of the unnecessary channel in convolutional layers. Note that our wavelet enabled sparse convolutions are complementary with channel pruning, as can be seen in Figure 9. While channel pruning can, in practice, greatly reduce both complexity and memory footprint, it requires heavy hyper-parameter search that we therefore choose to leave for future work.

9.4. Input resolution

Finally, one important factor that makes FastDepth efficient is that it is trained with 224×224 inputs, against our 640×480 input. While our method is best designed for higher-resolution regime where sparsity of wavelets is stronger, we still show that our method achieves decent results even at low-resolution, and report our scores in Table 12.



(a) **(b)**

Figure 9: Channel pruning (a) vs our sparse computation (b). Our sparse computation enabled by wavelets is complimentary with the channel pruning strategy to reduce the amount of computation, as both computation reduction methods operate in orthogonal dimensions.

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
[63]	Depth Hints	✓	S_{SGM}	192 × 640	0.106	0.780	4.695	0.193	0.875	0.958	0.980
	WaveletMonodepth MobileNetv2	✓	S_{SGM}	192 × 640	0.109	0.851	4.754	0.194	0.870	0.957	0.980
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.107	0.829	4.693	0.193	0.873	0.957	0.980
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.105	0.813	4.625	0.191	0.879	0.959	0.981
[63]	Depth Hints	✓	S_{SGM}	320 × 1024	0.099	0.723	4.445	0.187	0.886	0.961	0.982
	WaveletMonodepth MobileNetv2	✓	S_{SGM}	320 × 1024	0.104	0.772	4.545	0.188	0.880	0.960	0.982
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.102	0.739	4.452	0.188	0.883	0.960	0.981
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.096**	**0.710**	4.393	0.185	0.890	**0.962**	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	0.097	0.718	**4.387**	**0.184**	**0.891**	**0.962**	**0.982**

Table 10: Quantitative results on the KITTI dataset using MobileNetv2 encoder. We evaluate results of our method using a lighter encoder on KITTI [15], using the Eigen split. *Data column* (data source used for training): S is for self-supervised training on stereo images, MS is for models trained with both M (forward and backward frames) and S data and S_{SGM} refers to the extra stereo ground truth which was used in [63].

Method	Encoder	Depthwise	H × W	Abs Rel	RMSE	log_{10}	δ_1	δ_2	δ_3	ϵ_{acc}	ϵ_{comp}
Dense baseline	DenseNet161	-	480 × 640	0.1277	**0.5479**	**0.0539**	0.8430	**0.9681**	**0.9917**	**1.7170**	**7.0638**
Ours	DenseNet161	-	480 × 640	**0.1258**	0.5515	0.0542	**0.8451**	**0.9681**	**0.9917**	1.8070	7.1073
Ours	DenseNet161	✓	480 × 640	0.1275	0.5771	0.0557	0.8364	0.9635	0.9897	2.0133	7.1903
Dense baseline	MobileNetv2	-	480 × 640	0.1772	0.6638	0.0731	0.7419	0.9341	0.9835	1.8911	7.7960
Ours	MobileNetv2	-	480 × 640	0.1727	0.6776	0.0732	0.7380	0.9362	0.9844	1.9732	7.9004
Ours	MobileNetv2	✓	480 × 640	0.1734	0.6700	0.0731	0.7391	0.9347	0.9844	2.3036	8.0538

Table 11: Quantitative results on NYUv2 [58] using depth-wise convolutions and light-weight encoder We show that our method is compatible with other efficiency seeking approaches such as depth-wise separable convolutions and lower complexity encoders.

Method	Encoder	Depthwise	H × W	Abs Rel	RMSE	log_{10}	δ_1	δ_2	δ_3
Dense baseline	DenseNet161	-	224 × 224	0.1278	0.5715	0.0557	0.8368	0.9620	0.9901
Ours	DenseNet161	-	224 × 224	0.1279	0.5651	0.0549	0.8399	0.9652	0.9899
Ours	DenseNet161	✓	224 × 224	0.1304	0.5775	0.0564	0.8329	0.9613	0.9892
Dense baseline	MobileNetv2	-	224 × 224	0.1505	0.6221	0.0632	0.7984	0.9526	0.9878
Ours	MobileNetv2	-	224 × 224	0.1530	0.6409	0.0655	0.7844	0.9500	0.9864
Ours	MobileNetv2	✓	224 × 224	0.1491	0.6463	0.0646	0.7880	0.9506	0.9871

Table 12: Quantitative results on NYUv2 [58] using depth-wise convolutions and light-weight encoder We show that our method is compatible with other efficiency seeking approaches such as depth-wise separable convolutions and lower complexity encoders.

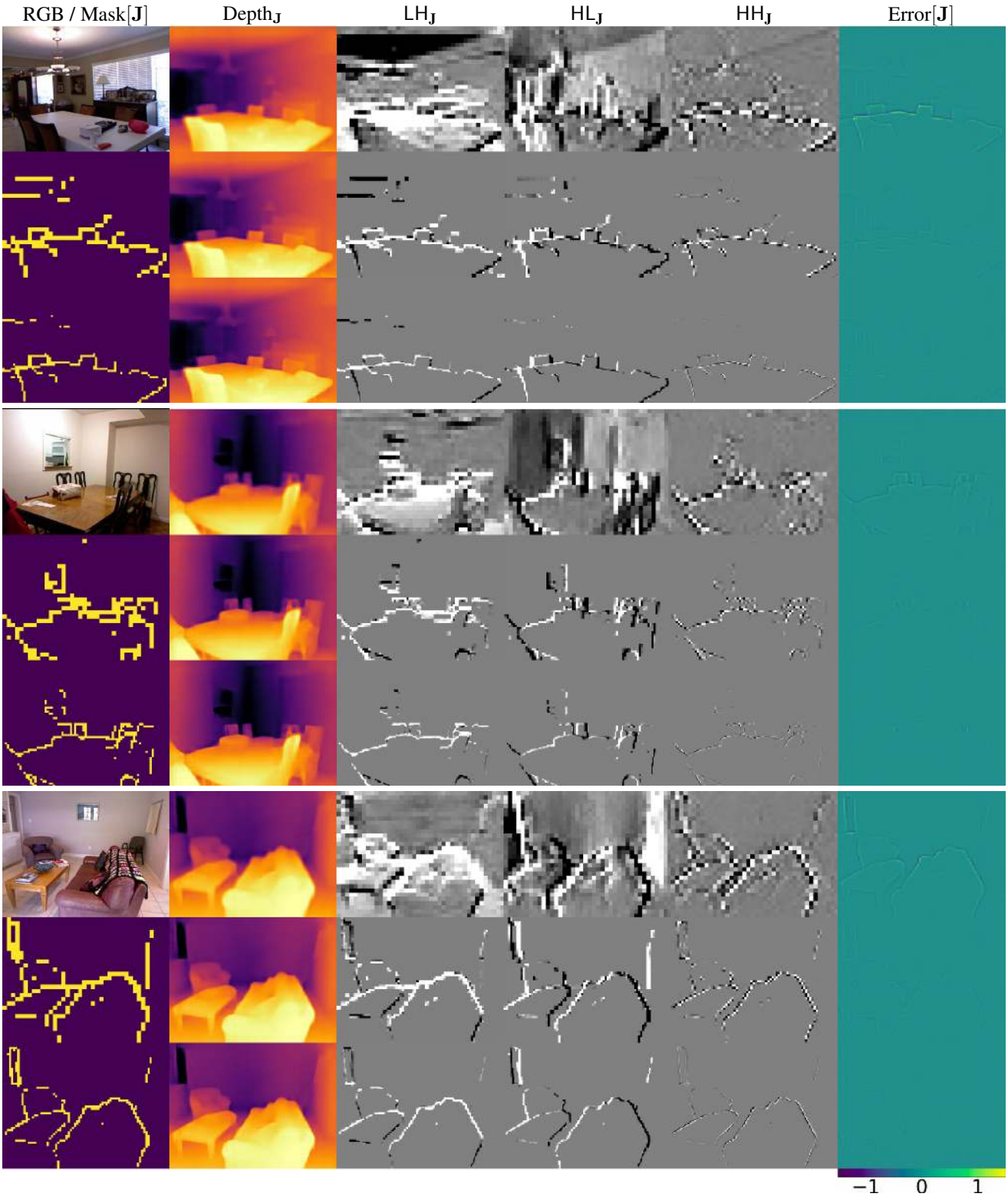


Figure 10: Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (1/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

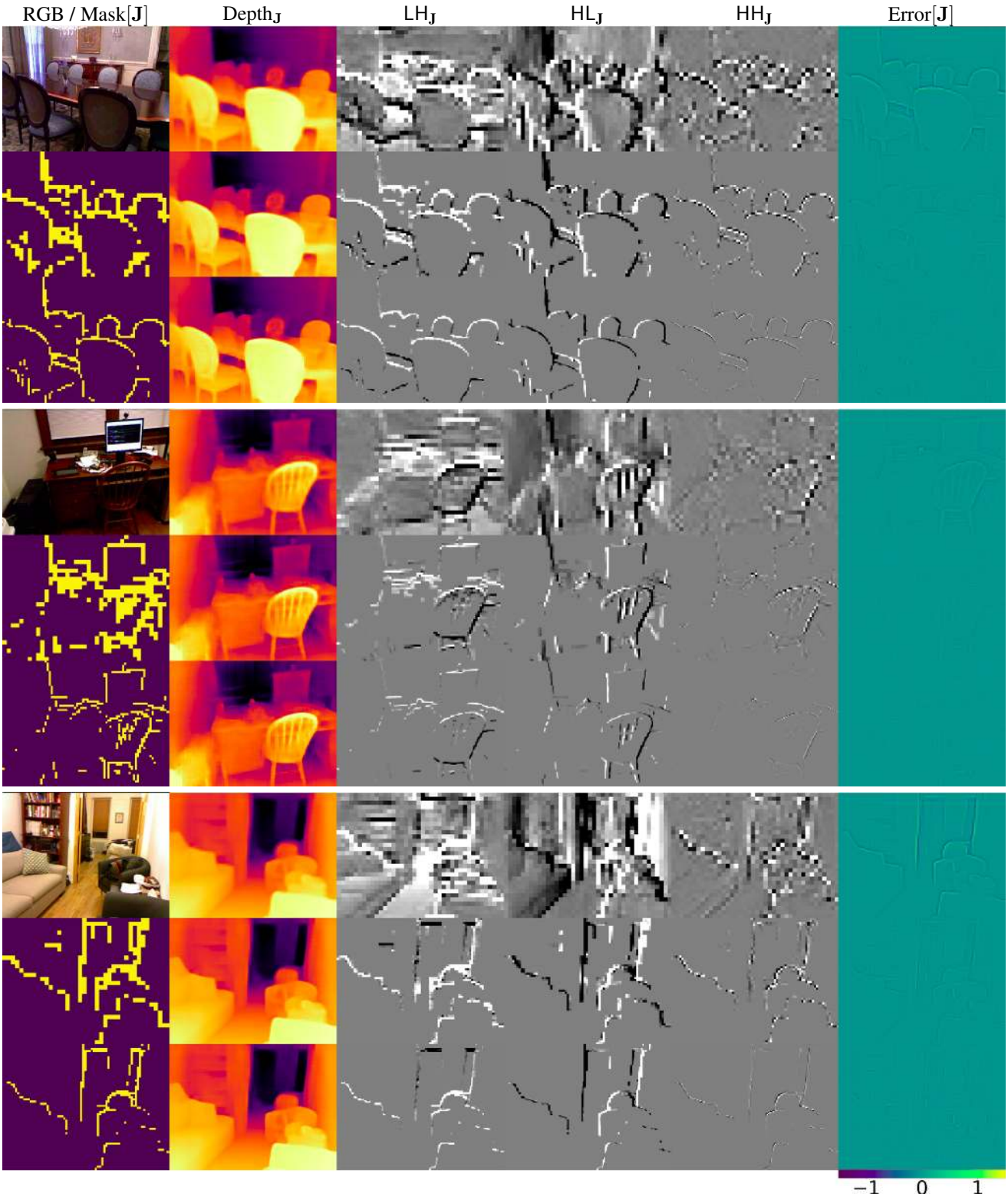


Figure 11: Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (2/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

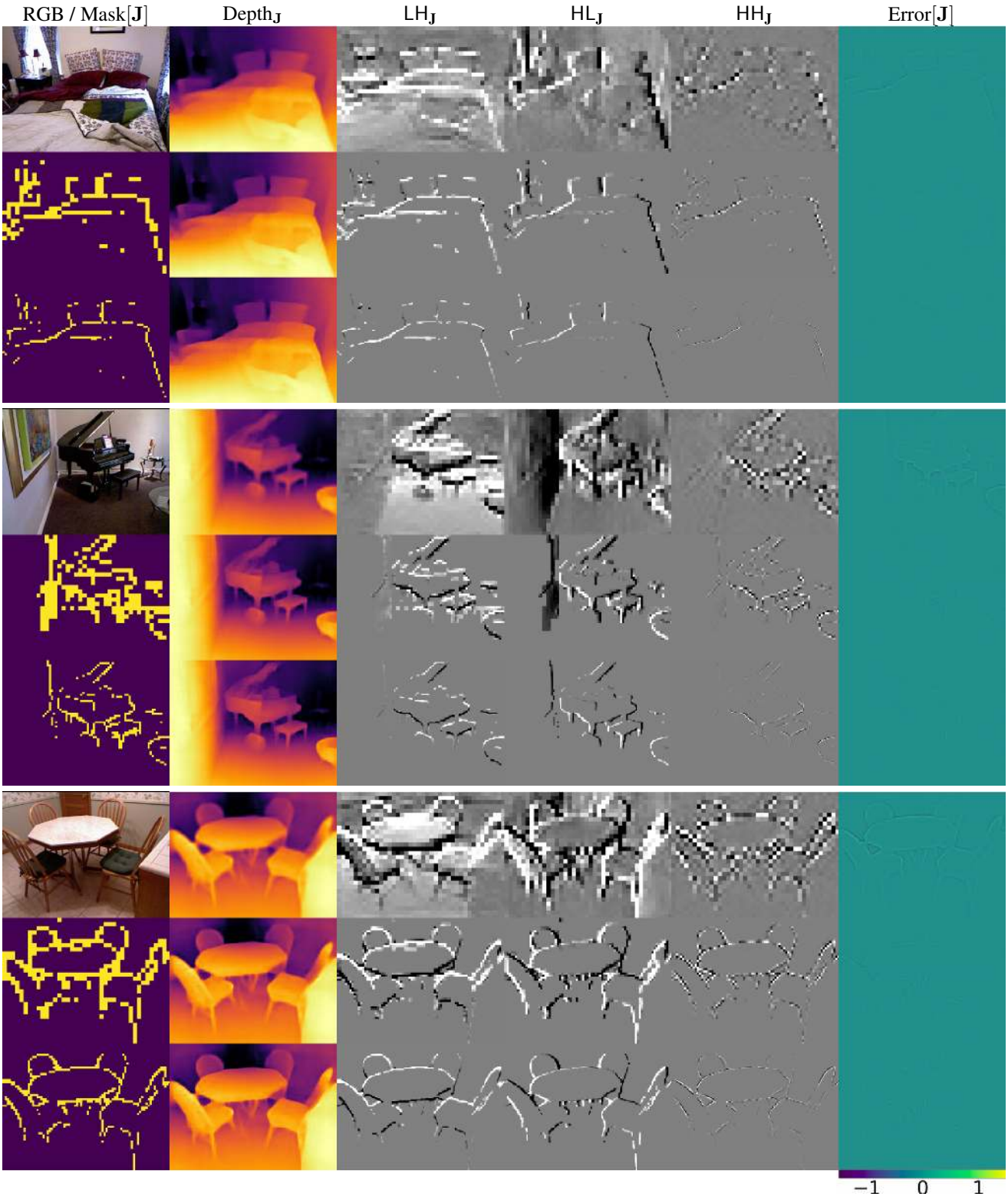


Figure 12: Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (3/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

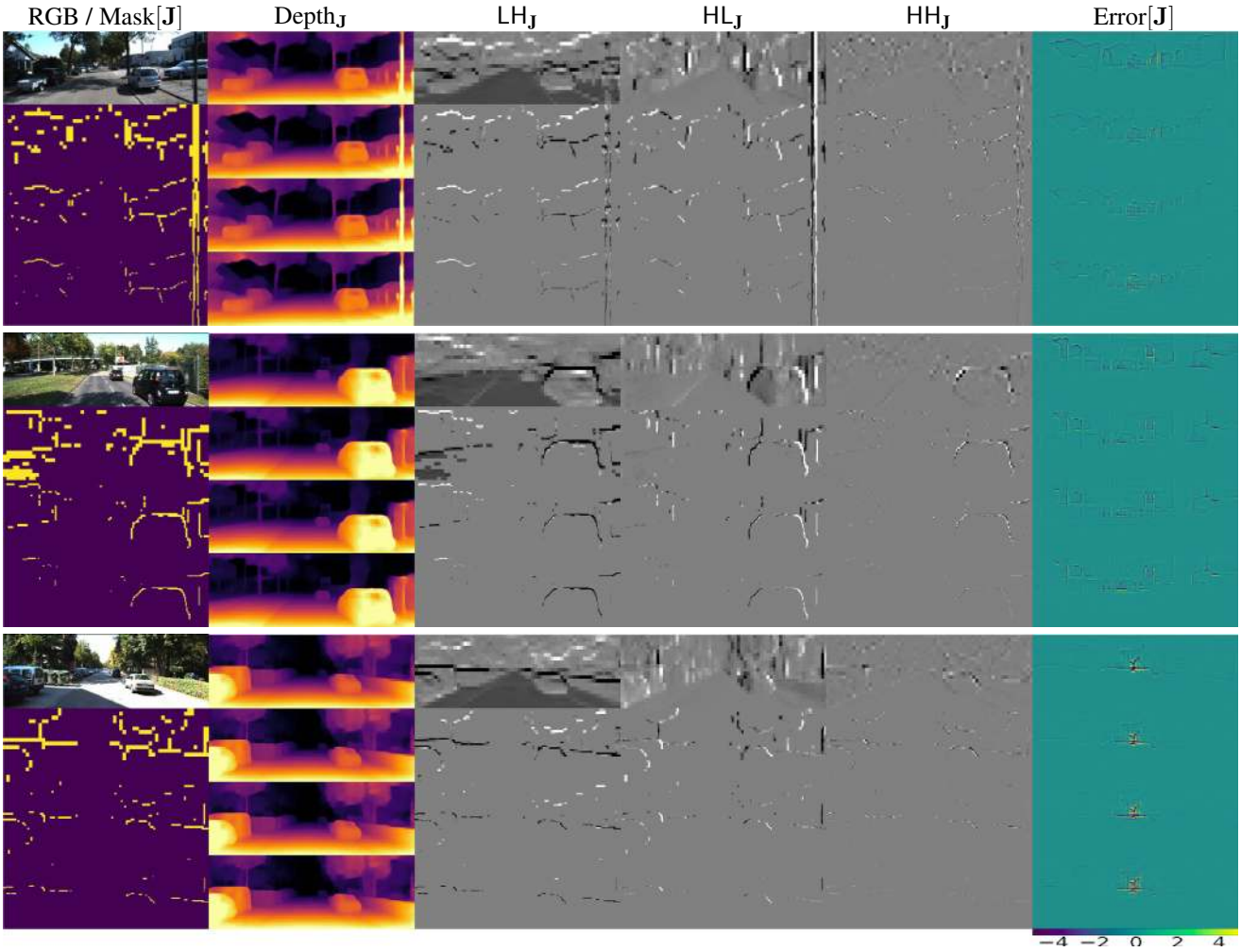


Figure 13: Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (1/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.

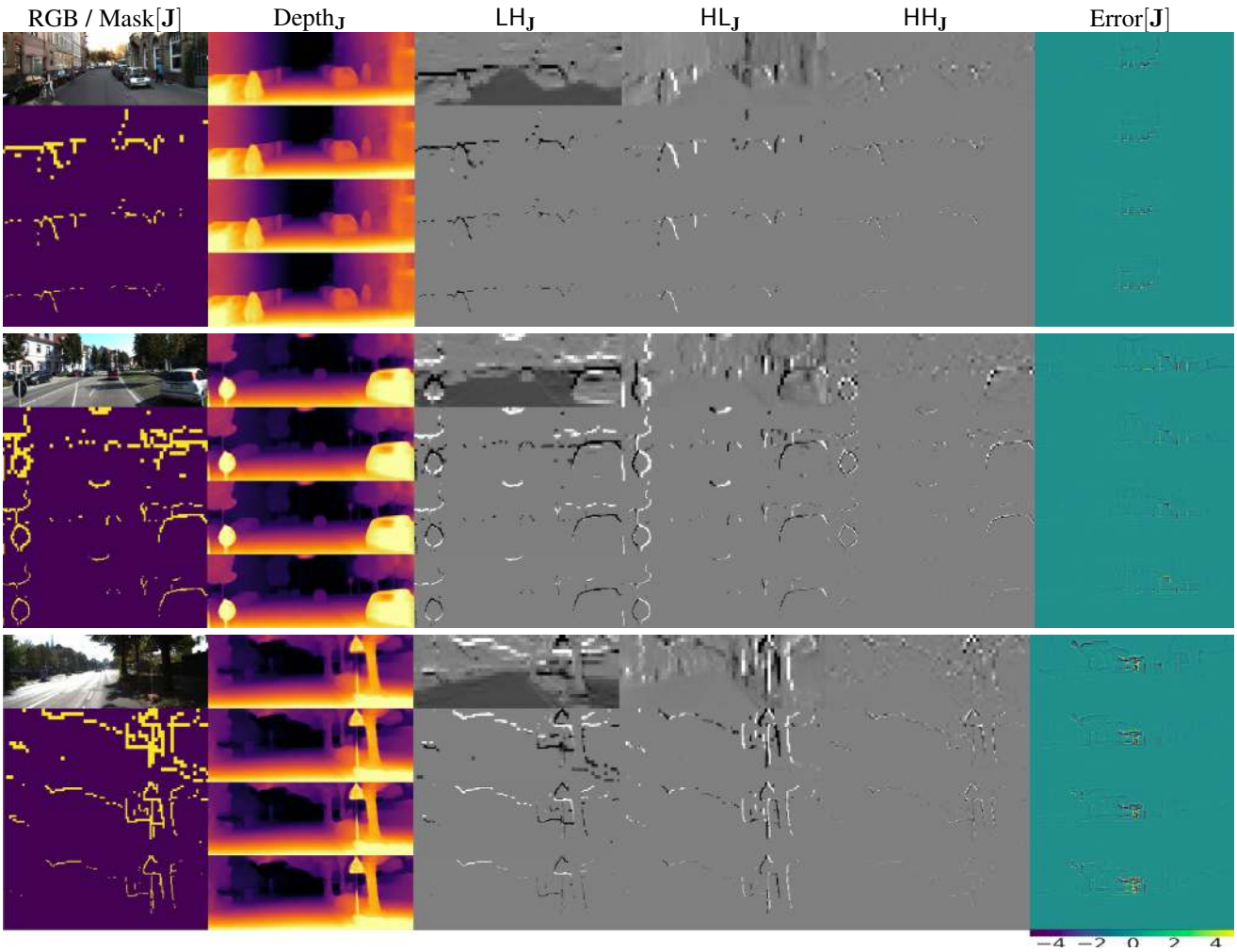


Figure 14: Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (2/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.

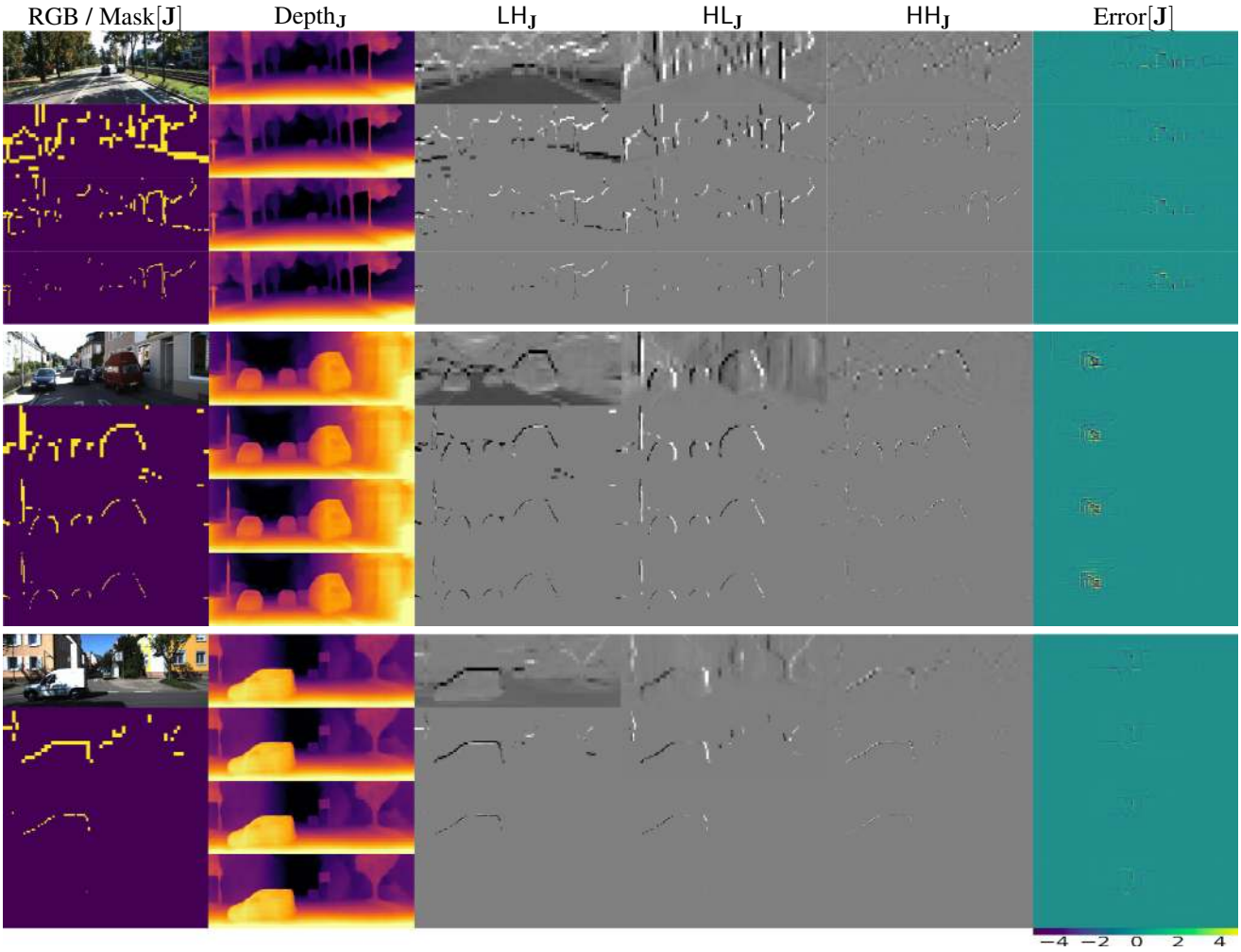


Figure 15: Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (3/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.