Predicting Sharp and Accurate Occlusion Boundaries in Monocular Depth Estimation Using Displacement Fields

Michaël Ramamonjisoa* Yuming Du* Vincent Lepetit
LIGM, IMAGINE, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée France
{first.lastname}@enpc.fr https://michaelramamonjisoa.github.io/projects/DisplacementFields

Abstract

Current methods for depth map prediction from monocular images tend to predict smooth, poorly localized contours for the occlusion boundaries in the input image. This is unfortunate as occlusion boundaries are important cues to recognize objects, and as we show, may lead to a way to discover new objects from scene reconstruction. To improve predicted depth maps, recent methods rely on various forms of filtering or predict an additive residual depth map to refine a first estimate. We instead learn to predict, given a depth map predicted by some reconstruction method, a 2D displacement field able to re-sample pixels around the occlusion boundaries into sharper reconstructions. Our method can be applied to the output of any depth estimation method and is fully differentiable, enabling end-to-end training. For evaluation, we manually annotated the occlusion boundaries in all the images in the test split of popular NYUv2-Depth dataset. We show that our approach improves the localization of occlusion boundaries for all state-of-the-art monocular depth estimation methods that we could evaluate ([35, 10, 6, 31]), without degrading the depth accuracy for the rest of the images.

1. Introduction

Monocular depth estimation (MDE) aims at predicting a depth map from a single input image. It attracts many interests, as it can be useful for many computer vision tasks and applications, such as scene understanding, robotic grasping, augmented reality, etc. Recently, many methods have been proposed to solve this problem using Deep Learning approaches, either relying on supervised learning [7, 6, 35, 10] or on self-learning [17, 58, 47], and these methods already often obtain very impressive results.

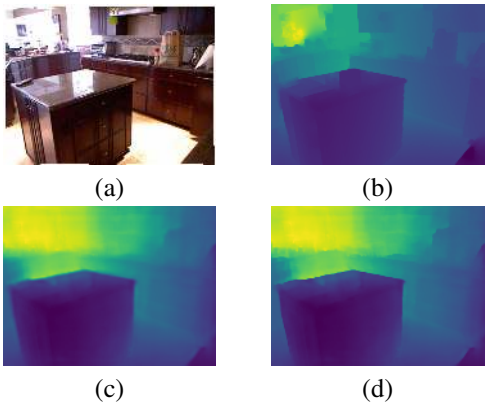


Figure 1. (a) Input image overlaid with our occlusion boundary (OB) manual annotation from NYUv2-OC++ in green, (b) Ground truth depth from NYUv2-Depth, (c) Predicted depth using [48], (d) Refined depth using our pixel displacement method.

However, as demonstrated in Fig. 1, despite the recent advances, the occlusion boundaries in the predicted depth maps remain poorly reconstructed. These occlusion boundaries correspond to depth discontinuities happening along the silhouettes of objects [53, 33]. Accurate reconstruction of these contours is thus highly desirable, for example for handling partial occlusions between real and virtual objects in Augmented Reality applications or more fundamentally, for object understanding, as we show in Fig. 2. We believe that this direction is particularly important: Depth prediction generalizes well to unseen objects and even to unseen object categories, and being able to reconstruct well the occlusion boundaries could be a promising line of research for unsupervised object discovery.

In this work, we introduce a simple method to overcome smooth occlusion boundaries. Our method improves their sharpness as well as their localization in the images. It relies on a differentiable module that takes an initial depth map provided by some depth prediction method, and re-sample it to obtain more accurate occlusion boundaries. Optionally, it can also take the color image as additional input for guidance information and obtain even better contour local-

* Authors with equal contribution.

(a) (b) (c) (d)

Figure 2. Application of our depth map refinement to 3D object extraction. (a-b) and (c-d) are two point cloud views of our extracted object. The left column shows point clouds extracted from the initially predicted depth image while the right one shows the result after using our depth refinement method. Our method suppresses long tails around object boundaries, as we achieve sharper occlusion boundaries.

ization. This is done by training a deep network to predict a 2D displacement field, applied to the initial depth map. This contrasts with previous methods that try to improve depth maps by predicting residual offsets for depth values [30, 64]. We show that predicting displacements instead of such a residual helps reaching sharper occluding boundaries. We believe this is because our module enlarges the family of functions that can be represented by a deep network. As our experiments show, this solution is complementary with all previous solutions as it systematically improves the localization and reconstruction of the occlusion boundaries.

In order to evaluate the occlusion boundary reconstruction performance of existing MDE methods and of our proposed method, we manually annotated the occlusion boundaries in all the images of the NYUv2 test set. Some annotations are shown in Fig. 3 and in the supplementary material. We rely on the metrics introduced in [34] to evaluate the reconstruction and localization accuracy of occlusion boundaries in predicted depth maps. We show that our method quantitatively improves the performance of all state-of-the-art MDE methods in terms of localization accuracy while maintaining or improving the global performance in depth reconstruction on two benchmark datasets.

In the rest of the paper, we first discuss related work. We then present our approach to sharpen occlusion boundaries in depth maps. Finally, we describe our experiments and results to prove how our method improves state-of-the-art MDE methods.

2. Related Work

Occlusion boundaries are a notorious and important challenge in dense reconstruction from images, for multiple reasons. For example, it is well known that in the case of stereo reconstruction, some pixels in the neighborhood of occluding boundaries are hidden in one image but visible in the other image, making the task of pixel matching difficult. In this context, numerous solutions have already been proposed to alleviate this problem [11, 27, 32, 13]. We focus here on recent techniques used in monocular depth estimation to improve the reconstruction of occlusion boundaries.

2.1. Deep Learning for Monocular Depth Estimation (MDE) and Occlusion Boundaries

MDE has already been studied thoroughly in the past due to its fundamental and practical importance. The surge of deep learning methods and large scale datasets has helped it improve continuously, with both supervised and self-supervised approaches. To tackle the scale ambiguity discussed in MDE discussed in [7], multi-scale deep learning methods have been widely used to learn depth priors as they extract global context from an image [6]. This approach has seen continuous improvement as more sophisticated and deeper architectures [51, 21, 35] were being developed.

Ordinal regression [10] was proposed as an alternative approach to direct depth regression in MDE. This method reached state-of-the-art on both popular MDE benchmarks NYUv2-Depth [52] and KITTI [12] and was later extended [36]. While ordinal regression helps reaching sharper occlusion boundaries, those are unfortunately often inaccurately located in the image, as our experiments show.

Another direction to improve the sharpness of object and occlusion boundaries in MDE is loss function design. L1-loss and variants such as the Huber and BerHu estimators [45, 66] have now been widely adopted instead of L2 since they tend to penalize discontinuities less. However, this solution is not perfect and inaccurate or smooth occlusion boundaries remains [31]. To improve their reconstruction quality, previous work considered depth gradient matching constraints [6] and depth-to-image gradient [23, 17] constraints, however for the latter, occlusion boundaries do not always correspond to strong image gradients, and conversely *e.g.* for areas with texture gradients. Constraints explicitly based on occlusion boundaries have also been proposed [62, 61, 48], leading to a performance increase in quality of occlusion boundary reconstruction.

Our work is complementary to all of the above approaches, as we show that it can improve the localization and reconstruction of occlusion boundaries in all state-of-the-art deep learning based methods.

2.2. Depth Refinement Methods

In this section we discuss two main approaches that can be used to refine depth maps predictions: We first discuss the formerly popular Conditional Random Fields (CRFs), then classical filtering methods and their newest versions.

Several previous work used CRFs post-processing potential to refine depth maps predictions. These works typically define pixel-wise and pair-wise loss terms between pixels and their neighbors using an intermediate predicted guidance signal such as geometric features [55] or relia-

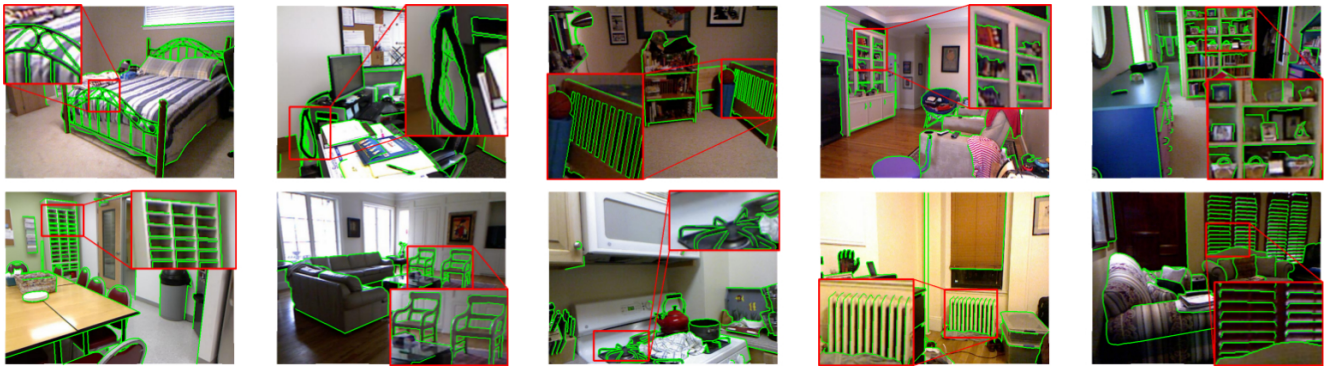


Figure 3. Samples of our NYUv2-OC++ dataset, which extends NYUv2-OC from [48]. The selected highlighted regions in red rectangles emphasize the high-quality and fine-grained annotations.

bility maps [24]. An initially predicted depth map is then refined by performing inference with the CRF, sometimes iteratively or using cascades of CRFs [59, 60]. While most of these methods help improving the initial depth predictions and yield qualitatively more appealing results, those methods still under-perform state-of-the-art non-CRF MDE methods while being more computationally expensive.

Another option for depth refinement is to use image enhancement methods. Even though these methods do not necessarily explicitly target occlusion boundaries, they can be potential alternative solutions to ours and we compare with them in Section 4.5.

Bilateral filtering is a popular and currently state-of-the-art method for image enhancement, in particular as a denoising method preserving image contours. Although historically, it was limited for post-processing due to its computational complexity, recent work have successfully made bilateral filters reasonably efficient and fully differentiable [39, 57]. These recent methods have been successful when applied in downsampling-upsampling schemes, but have not been used yet in the context of MDE. Guided filters [20] have been proposed as an alternative simpler version of the bilateral filter. We show in our experiments that both guided and bilateral filters sharpen occlusion boundaries thanks to their usage of the image for guidance. They however sometime produce false depth gradients artifacts. The bilateral solver [2] formulates the bilateral filtering problem as a regularized least-squares optimization problem, allowing fully differentiable and much faster computation. However, we show in our experiments that our end-to-end trainable method compares favorably against this method, both in speed and accuracy.

2.3. Datasets with Image Contours

Several datasets of image contours or occlusion boundaries already exist. Popular datasets for edge detection training and evaluation were focused on perceptual [44, 43, 1] or object instance [8] boundaries. However, those datasets often lack annotation of the occlusion relationship between two regions separated by the boundaries. Other datasets [50, 26, 25, 56] annotated occlusion relationship between objects, however they do not contain ground truth depth.

The NYUv2-Depth dataset [52] is a popular MDE benchmark which provides such depth ground truth. Several methods for instance boundary detection have benefited from this depth information [18, 5, 49, 19, 4] to improve their performances on object instance boundaries detection.

The above cited datasets all lack object self-occlusion boundaries and are sometimes inaccurately annotated. Our NYUv2-OC++ dataset provides manual annotations for the occlusion boundaries on top of NYUv2-Depth for all of its 654 test images. As discussed in [48], even though it is a tedious task, manual annotation is much more reliable than automated annotation that could be obtained from depth maps. Fig. 3 illustrates the extensive and accurate coverage of the occlusion boundaries of our annotations. This dataset enables simultaneous evaluation of depth estimation methods and occlusion boundary reconstruction as the 100 images iBims dataset [34], but is larger and has been widely used for MDE evaluation.

3. Method

In this section, we introduce our occlusion boundary refinement method as follows. Firstly, we present our hypothesis on the typical structure of predicted depth maps around occlusion boundaries and derive a model to transform this structure into the expected one. We then prove our hypothesis using an hand-crafted method. Based on this model, we propose an end-to-end trainable module which can resample pixels of an input depth image to restore its sharp occlusion boundaries.

3.1. Prior Hypothesis

Occlusion boundaries correspond to regions in the image where depth exhibits large and sharp variations, while the

other regions tend to vary much smoother. Due to the small proportion of such sharp regions, neural networks tend to predict over-smoothed depths in the vicinity of occlusion boundaries.

We show in this work that sharp and accurately located boundaries can be recovered by resampling pixels in the depth map. This resampling can be formalized as:

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p} + \boldsymbol{\delta p}(\boldsymbol{p})) \,, \tag{1}$$

where D is a depth map, $\boldsymbol{p}$ denotes an image location in domain Ω, and $\boldsymbol{\delta p}(\boldsymbol{p})$ a 2D displacement that depends on $\boldsymbol{p}$. This formulation allows the depth values on the two sides of occlusion boundaries to be "stitched" together and replace the over-smoothed depth values. While this method shares some insights with Deformable CNNs [28] where the convolutional kernels are deformed to adapt to different shapes, our method is fundamentally different as we displace depth values using a predicted 2D vector for each image location.

Another option to improve depth values would be to predict an additive residual depth, which can be formalized as, for comparison:

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p}) + \Delta D(\boldsymbol{p}) \,. \tag{2}$$

We argue that updating the predicted depth $\widehat{\boldsymbol{D}}$ using predicted pixel shifts to recover sharp occlusion boundaries in depth images works better than predicting the residual depth. We validate this claim with the experiments presented below on toy problems, and on real predicted depth maps in Section 4.

3.2. Testing the Optimal Displacements

To first validate our assumption that a displacement field $\boldsymbol{\delta p}(\boldsymbol{p})$ can improve the reconstructions of occlusion boundaries, we estimate the optimal displacements using ground truth depth for several predicted depth maps as:

$$\forall \boldsymbol{p} \in \Omega, \boldsymbol{\delta p}^* = \underset{\boldsymbol{\delta p}:\ \boldsymbol{p}+\boldsymbol{\delta p}\ \in \mathcal{N}(\boldsymbol{p})}{\arg\min} (D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p} + \boldsymbol{\delta p}))^2 \,. \tag{3}$$

In words, solving this problem is equivalent to finding for each pixel the optimal displacement $\boldsymbol{\delta p}^*$ that reconstructs the ground truth depth map D from a predicted depth map $\widehat{D}$. We solve Eq. (3) by performing for all pixels $\boldsymbol{p}$ an exhaustive search of $\boldsymbol{\delta p}$ within a neighborhood $\mathcal{N}(\boldsymbol{p})$ of size 50×50. Qualitative results are shown in Fig. 4. The depth map obtained by applying this optimal displacement field is clearly much better.

In practice, we will have to predict the displacement field to apply this idea. This is done by training a deep network, which is detailed in the next subsection. We then check on a toy problem that this yields better performance than predicting residual depth with a deep network of similar capacity.

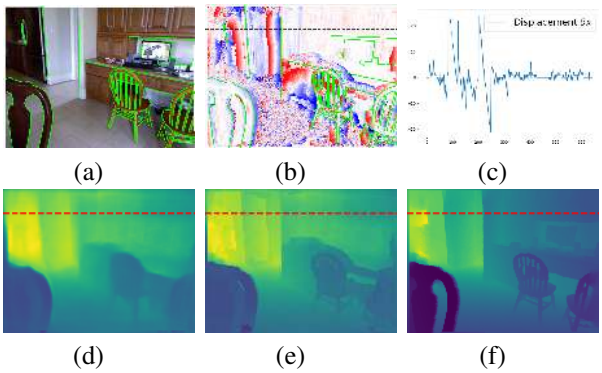


Figure 4. Refinement results using the gold standard method described in Section 3.2 to recover the optimal displacement field (best seen in color). (a) is the input RGB image with superimposed NYUv2-OC++ annotation in green and (d) its associated Ground Truth depth. (e) is the prediction using [35] with pixel displacements $\boldsymbol{\delta p}$ from Eq. (3) and (f) the refined prediction. (b) is the horizontal component of the displacement field $\boldsymbol{\delta p}^*$ obtained by Eq. (3). Red and blue color indicate positive and negative values respectively. (c) is the horizontal component δx of displacement field $\boldsymbol{\delta p}^*$ along the dashed red line drawn in (b,c,d,e).

3.3. Method Overview

Based on our model, we propose to learn the displacements of pixels in predicted depth images using CNNs. Our approach is illustrated in Fig. 6: Given a predicted depth image $\widehat{D}$, our network predicts a displacement field $\boldsymbol{\delta p}$ to resample the image locations in depth map $\widehat{D}$ according to Eq. (1). This approach can be implemented as a Spatial Transformer Network [29].

Image guidance can be helpful to improve the occlusion boundary precision in refined depth maps and can also help to discover edges that were not visible in the initial predicted depth map $\widehat{D}$. However, it should be noted that our network can still work even without image guidance.

3.4. Training Our Model on Toy Problems

In order to verify that displacement fields $\boldsymbol{\delta p}$ presented in Section 3.1 can be learned, we first define a toy problem in 1D.

In this toy problem, as shown in Fig. 5, we model the signals D to be recovered as piecewise continuous functions, generated as sequences of basic functions such as step, affine and quadratic functions. These samples exhibit strong discontinuities at junctions and smooth variations everywhere else, which is a property similar to real depth maps. We then convolve the D signals with random-size (blurring) Gaussian kernels to obtain smooth versions $\widehat{D}$. This gives us a training set $\mathcal{T}$ of $(\widehat{D}, D)$ pairs.

We use $\mathcal{T}$ to train a network $f(.; \Theta_f)$ of parameters to

Figure 5. Comparison between displacement and residual update learning. Both residual and displacement learning can predict sharper edges, however residual updates often produce artifacts along the edges while displacements update does not.

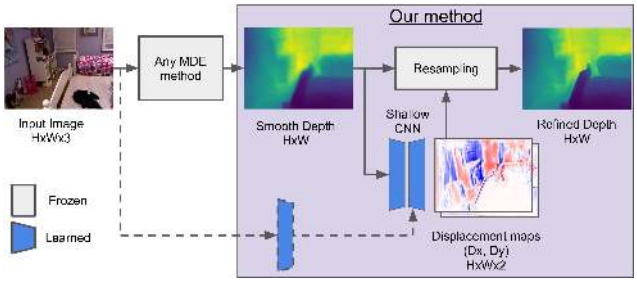


Figure 6. Our proposed pipeline for depth edge sharpening. The dashed lines define the optional guidance with RGB features for our shallow network.

predict a displacement field:

$$\min_{\Theta_f} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}\left(\boldsymbol{p} + f(\widehat{D};\Theta_f)(\boldsymbol{p})\right)\right) . \tag{4}$$

and a network $g(.;\Theta_g)$ of parameters to predict a residual depth map:

$$\min_{\Theta_g} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p}) + g(\widehat{D};\Theta_g)(\boldsymbol{p})\right) , \tag{5}$$

where $L(.)$ is some loss. In our experiments, we evaluate the l_1, l_2, and Huber losses.

As shown in Fig. 5, we found that predicting a residual update produces severe artifacts around edges such as overshooting effects. We argue that these issues arise because the residual CNN g is also trained on regions where the values of $\widehat{D}$ and D are different even away from edges, thus encouraging network g to also correct these regions. By contrast, our method does not create such overshooting effects as it does not alter the local range of values around edges.

It is worth noticing that even when we allow $\widehat{D}$ and D to have slightly different values in non-edge areas -which simulates residual error between predicted and ground truth depth-, our method still converges to a good solution, compared to the residual CNN.

We extend our approach validation from 1D to 2D, where the 1D signal is replaced by 2D images with different polygons of different values. We apply the same operation to smooth the images and then use our network to recover the original sharp images from the smooth ones. We observed similar results in 2D: The residual CNN always generates artifacts. Some results of our 2D toy problem can be found in supplementary material.

3.5. Learning to Sharpen Depth Predictions

To learn to produce sharper depth predictions using displacement fields, we first trained our method in a similar fashion to the toy problem described in Section 3.4. While this already improves the quality of occlusion boundaries of all depth map predictions, we show that we can further improve quantitative results by training our method using predictions of an MDE algorithm on the NYUv2-Depth dataset as input and the corresponding ground truth depth as target output. We argue that this way, the network learns to correct more complex distortions of the ground truth than Gaussian blurring. In practice, we used the predictions of [48] to demonstrate this, as it is state-of-the-art on occlusion boundary sharpness. We show that this not only improves the quality of depth maps predicted by [48] but all other available MDE algorithms. Training our network using the predictions of other algorithms on the NYUv2 would further improve their result, however not all of them provide their code or their predictions on the official *training set*. Finally, our method is fully differentiable. While we do not do it in this paper, it is also possible to train an MDE jointly with our method, which should yield even better results.

4. Experiments

In this section, we first detail our implementation, describe the metrics we use to evaluate the reconstruction of the occlusion boundaries and the accuracy of the depth prediction, and then present the results of the evaluations of our method on the outputs from different MDE methods.

4.1. Implementation Details

We implemented our network using the Pytorch framework [46]. Our network is a light-weight encoder-decoder network with skip connections, which has one encoder for depth, an optional one for guidance with the RGB image, and a shared decoder. Details on each component can be found in the supplementary material. We trained our network on the output of a MDE method [48], using Adam optimization with an initial learning rate of 5e-4 and weight decay of 1e-6, and the *poly* learning rate policy [65], during 32k iterations on NYUv2 [52]. This dataset contains 1449 pairs RGB and depth images, split into 795 samples for training and 654 for testing. Batch size was set to 1. The input images were resized with scales $[0.75, 1, 1, 5, 2]$

(a) (b) (c) (d) (e) (f)

Figure 7. Refinement results using our method (best seen in color). From left to right: (a) input RGB image with NYUv2-OC++ annotation in green, (b) SharpNet [48] depth prediction, (c) Refined prediction, (d) Ground truth depth, (e) Horizontal and (f) Vertical components of the displacement field. Displacement fields are clipped between ± 15 pixels. Although SharpNet is used as an example here because it is currently state-of-the-art on occlusion boundary accuracy, similar results can be observed when refining predictions from other methods.

and then cropped and padded to 320×320. We tried to learn on the *raw* depth maps from the NYUv2 dataset, without success. This is most likely due to the fact that missing data occurs mostly around the occlusion boundaries. Dense depth maps are therefore needed which is why we use [37] to inpaint the missing data in the raw depth maps.

4.2. Evaluation metrics

4.2.1 Evaluation of monocular depth prediction

As for previous work [7, 6, 35], we evaluate the monocular depth predictions using the following metrics: Root Mean Squared linear Error (R_{lin}), mean absolute relative error (rel), mean $\log_{10}$ error ($\log_{10}$), Root Mean Squared log Error (R_{log}), and the accuracy under threshold ($\sigma_i < 1.25^i$, $i = 1, 2, 3$).

4.2.2 Evaluation of occlusion boundary accuracy

Following the work of Koch *et al.* [34], we evaluate the accuracy of occlusion boundaries (OB) using the proposed depth boundary errors, which evaluate the accuracy ϵ_a and completion ϵ_c of predicted occlusion boundaries. The boundaries are first extracted using a Canny edge detector [3] with predefined thresholds on a normalized predicted depth image. As illustrated in Fig. 8, ϵ_a is taken as the average Chamfer distance in pixels [9] from the predicted boundaries to the ground truth boundaries; ϵ_{com} is taken as the average Chamfer distance from ground truth boundaries to the predicted boundaries.

4.3. Evaluation on the NYUv2 Dataset

We evaluate our method by refining the predictions of different state-of-the-art methods [7, 35, 10, 48, 31, 63].

(a) (b)

Figure 8. Occlusion boundary evaluation metrics based on the Chamfer distance, as introduced in [34]. The blue lines represent the ground truth boundaries, the red curve the predicted boundary. Only the boundaries in the green area are taken into account during the evaluation of accuracy (a), and only the yellow area are taken into account during the evaluation of completeness (b). (a) The accuracy is evaluated from the distances of points on the predicted boundaries to the ground truth boundaries. (b) The completeness is evaluated from the distances of points on the ground truth boundaries to the predicted boundaries.

Our network is trained using the 795 labeled NYUv2 depth images of training dataset with corresponding RGB images as guidance.

To enable a fair comparison, we evaluate only pixels inside the crop defined in [7] for all methods. Table 1 shows the evaluation of refined predictions of different methods using our network. With the help of our proposed network, the occlusion boundary accuracy of all methods can be largely improved, without degrading the global depth estimation accuracy. We also show qualitative results of our refinement method in Fig. 12.

4.4. Evaluation on the iBims Dataset

We applied our method trained on NYUv2 dataset to refine the predictions from various methods [7, 6, 35, 41, 38, 40, 48] on the iBims dataset [34]. Table 2 shows that our network significantly improves the accuracy and complete-

Method	Refine	Depth error (↓)				Depth accuracy (↑)			OBs (↓)	
		rel	$\log_{10}$	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Eigen *et al.* [7]	-	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
	✓	0.232	0.094	0.758	0.263	0.615	0.889	0.971	2.168	8.173
Laina *et al.* [35]	-	0.142	0.059	0.510	0.181	0.818	0.955	0.988	4.702	8.982
	✓	0.140	0.059	0.509	0.180	0.819	0.956	0.989	2.372	7.041
Fu *et al.* [10]	-	0.131	0.053	0.493	0.174	0.848	0.956	0.984	3.872	8.117
	✓	0.136	0.054	0.502	0.178	0.844	0.954	0.983	3.001	7.242
Ramamonjisoa and Lepetit [48]	-	0.116	0.053	0.448	0.163	0.853	0.970	0.993	3.041	8.692
	✓	0.117	0.054	0.457	0.165	0.848	0.970	0.993	1.838	6.730
Jiao *et al.* [31]	-	0.093	0.043	0.356	0.134	0.908	**0.981**	**0.995**	8.730	9.864
	✓	**0.092**	**0.042**	**0.352**	**0.132**	**0.910**	**0.981**	**0.995**	2.410	8.230
Yin *et al.* [63]	-	0.112	0.047	0.417	0.144	0.880	0.975	0.994	1.854	7.188
	✓	0.112	0.047	0.419	0.144	0.879	0.975	0.994	**1.762**	**6.307**

Table 1. Evaluation of our method on the output of several state-of-the-art methods on NYUv2. Our method significantly improves the occlusion boundaries metrics ϵ_a and ϵ_c without degrading the other metrics related to the overall depth accuracy. These results were computed using available depth maps predictions (apart from Jiao *et al.* [31] who sent us their predictions) within the image region proposed in [7]. (↓: Lower is better; ↑: Higher is better).

ness metrics for the occluding boundaries of all predictions on this dataset as well.

4.5. Comparison with Other Methods

To demonstrate the efficiency of our proposed method, we compare our method with existing filtering methods [54, 20, 2, 57]. We use the prediction of [7] as input, and compare the accuracy of depth estimation and occlusion boundaries of each method. Note that for the filters with hyper-parameters, we tested each filter with a series of hyper-parameters and select the best refined results. For the Fast Bilateral Solver (FBS) [2] and the Deep Guided Filter (GF) [57], we use their default settings from the official implementation. We keep the same network with and without Deep GF, and train both times with the same learning rate and data augmentation.

As shown in Table 3, our method achieves the best accuracy for occlusion boundaries. Finally, we compare our method against the additive residual prediction method discussed in 3.1. We keep the same U-Net architecture, but replace the displacement operation with an addition as described in Eq. (2), and show that we obtain better results. We argue that the performance of the deep guided filter and additive residual are lower due to generated artifacts which are discussed in Section 3.4. In Table 4, we also compare our network's computational efficiency against reference depth estimation and refinement methods.

4.6. Influence of the Loss Function and the Guidance Image

In this section, we report several ablation studies to analyze the favorable factors of our network in terms of performances. Fig. 6 shows the architecture of our baseline network. All the following networks are trained using the same setting detailed in Section 4.1 and trained on the official 795 RGB-D images of the NYUv2 training set. To evaluate the effectiveness of our method, we choose the predictions of [7] as input, but note that the conclusion is still valid with other MDE methods. Please see supplementary material for further results.

4.6.1 Loss Functions for Depth Prediction

In Table 5, we show the influence of different loss functions. We apply the Pytorch official implementation of l_1, l_2, and the Huber loss. The Disparity loss supervises the network with the reciprocal of depth, the target depth y_{target} is defined as $y_{target} = M/y_{original}$, where M here represents the maximum of depth in the scene. As shown in Table 5, our network trained with l_1 loss achieves the best accuracy for the occlusion boundaries.

4.6.2 Guidance image

We explore the influence of different types of guidance image. The features of guidance images are extracted using an encoder with the same architecture as the depth encoder, except that Leaky ReLU [42] activations are all replaced by standard ReLU [15]. We perform feature fusion of guidance and depth features using skip connections from the guidance and depth decoder respectively at similar scales.

Table 6 shows the influence of different choices for the guidance image. The edge images are created by accumulating the detected edges using a series of Canny detector with different thresholds. As shown in Table 6, using the original RGB image as guidance achieves the highest accuracy, while using the image converted to grayscale achieves

Method	Refine	Depth error (↓) rel	$\log_{10}$	R_{lin}	Depth accuracy (↑) σ_1	σ_2	σ_3	OB (↓) ϵ_a	ϵ_c
Eigen *et al.* [7]	-	0.32	0.17	1.55	0.36	0.65	0.84	9.97	9.99
	✓	0.32	0.17	1.54	0.37	0.66	0.85	4.83	8.78
Eigen and Fergus (AlexNet) [6]	-	0.30	0.15	1.38	0.40	0.73	0.88	4.66	8.68
	✓	0.30	0.15	1.37	0.41	0.73	0.88	4.10	7.91
Eigen and Fergus (VGG) [6]	-	0.25	0.13	1.26	0.47	0.78	0.93	4.05	8.01
	✓	0.25	0.13	1.25	0.48	0.78	0.93	3.95	7.57
Laina *et al.* [35]	-	0.26	0.13	1.20	0.50	0.78	0.91	6.19	9.17
	✓	0.25	0.13	1.18	0.51	0.79	0.91	3.32	7.15
Liu *et al.* [41]	-	0.30	0.13	1.26	0.48	0.78	0.91	2.42	7.11
	✓	0.30	0.13	1.26	0.48	0.77	0.91	2.36	7.00
Li *et al.* [38]	-	**0.22**	**0.11**	1.09	0.58	**0.85**	**0.94**	3.90	8.17
	✓	**0.22**	**0.11**	1.10	0.58	0.84	**0.94**	3.43	7.19
Liu *et al.* [40]	-	0.29	0.17	1.45	0.41	0.70	0.86	4.84	8.86
	✓	0.29	0.17	1.47	0.40	0.69	0.86	2.78	7.65
Ramamonjisoa and Lepetit [48]	-	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	3.69	7.82
	✓	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	**2.13**	**6.33**

Table 2. Evaluation of our method on the output of several state-of-the-art methods on iBims.

Method	Depth error (↓) rel	log10	R_{lin}	R_{log}	Depth accuracy (↑) σ_1	σ_2	σ_3	OB (↓) ϵ_a	ϵ_c
Baseline [7]	0.234	0.095	0.766	0.265	0.610	0.886	*0.971*	9.926	9.993
Bilateral Filter [54]	0.236	0.095	0.765	0.265	0.611	0.887	*0.971*	9.313	9.940
GF [20]	0.237	0.095	0.767	0.265	0.610	0.885	*0.971*	6.106	9.617
FBS [2]	0.236	0.095	0.765	0.264	0.611	0.887	*0.971*	5.428	9.454
Deep GF [57]	0.306	0.116	0.917	0.362	0.508	0.823	0.948	4.318	9.597
Residual	0.286	0.132	0.928	0.752	0.508	0.807	0.931	5.757	9.785
Our Method	**0.232**	**0.094**	**0.757**	**0.263**	**0.615**	**0.889**	*0.971*	**2.302**	**8.347**

Table 3. Comparison with existing methods for image enhancement, adapted to the depth map prediction problems. Our method performs the best for this problem over all the different metrics.

Method	SharpNet [48]	VNL [63]	Deep GF[57]	Ours
FPS - GPU	83.2 ± 6.0	32.2± 2.1	70.5 ± 7.5	**100.0** ± 7.3
FPS - CPU	2.6 ± 0.0	*	4.0 ± 0.1	**5.3** ± 0.15

Table 4. Speed comparison with other reference methods implemented using Pytorch. Those numbers were computed using a single GTX Titan X and Intel Core i7-5820K CPU. using 320x320 inputs. Runtime statistics are computed over 200 runs.

the lowest accuracy, as information is lost during the conversion. Using the Canny edge detector can help to alleviate this problem, as the network achieves better results when switching from gray image to binary edge maps.

5. Conclusion

We showed that by predicting a displacement field to resample depth maps, we can significantly improve the reconstruction accuracy and the localization of occlusion boundaries of any existing method for monocular depth prediction. To evaluate our method, we also a new dataset of precisely labeled occlusion boundaries. Beyond evaluation of occlusion boundary reconstruction, this dataset should be extremely valuable for future methods to learn to detect more precisely occlusion boundaries.

Method	Depth error (↓) rel	log10	R_{lin}	R_{log}	Depth accuracy (↑) σ_1	σ_2	σ_3	OB (↓) ϵ_a	ϵ_c
Baseline [7]	0.234	0.095	0.766	0.265	0.610	0.886	0.971	9.926	9.993
l_1	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**
l_2	*0.232*	*0.094*	**0.757**	*0.263*	*0.615*	*0.889*	0.971	2.302	8.347
Huber	*0.232*	0.095	0.758	*0.263*	*0.615*	*0.889*	**0.972**	2.225	8.282
Disparity	0.234	0.095	0.761	0.264	0.613	0.888	0.971	2.312	8.353

Table 5. Evaluation of different loss functions for learning the displacement field. The l_1 norm yields the best results.

Method	Depth error (↓) rel	log10	R_{lin}	R_{log}	Depth accuracy (↑) σ_1	σ_2	σ_3	OB (↓) ϵ_a	ϵ_c
Baseline [7]	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
No guidance	0.236	0.096	0.771	0.268	0.608	0.883	0.969	6.039	9.832
Gray	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.659	8.681
Binary Edges	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.466	8.483
RGB	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**

Table 6. Evaluation of different ways of using the input image for guidance. Simply using the original color image works best.

Acknowledgement

We thank Xuchong Qiu and Yang Xiao for their help in annotating the NYUv2-OC++ dataset. This project has received funding from the CHISTERA-IPALM project.

Predicting Sharp and Accurate Occluding Contours in Monocular Depth Estimation using Displacement Maps

Supplementary Material

Michaël Ramamonjisoa* Yuming Du* Vincent Lepetit
LIGM, IMAGINE, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée France
{first.lastname}@enpc.fr https://michaelramamonjisoa.github.io/projects/DisplacementFields

6. Architecture Details

In Fig. 9, we detail the architecture of our network, which is composed of two encoders, one for Depth and an optional one for Guidance. We use a single decoder which combines the respective outputs of the Depth and Guidance decoders using residual blocks and skip connections. We give full details of each block in the following.

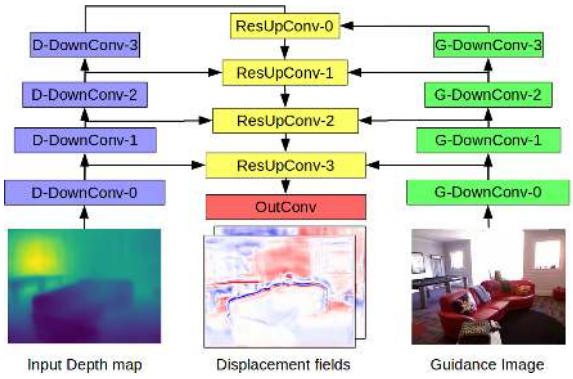


Figure 9. Detailed architecture of our displacement field prediction network. Details on the Depth and Guidance encoders are provided in Sections 6.1 and 6.2 respectively.

6.1. Depth Encoder

Our Depth is a standard encoder with a cascade of four down-convolutions, denoted *D-DownConv*. The D-DownConv blocks are composed of a convolution layer with 3x3 kernel, followed by a 2x2 MaxPooling, and a LeakyReLU [42] activation. The D-DownConv block convolution layers have respectively [32, 64, 128, 256] channels. They all use batch-normalization, are initialized using Xavier [14] initialization and a Leaky ReLU [42] activation.

6.2. Guidance Encoder

Our Guidance encoder is composed of a cascade four of down-convolutions as in [22], which we denote G-DownConv. It is identical to the D-DownConv block described in 6.1 except that it uses simple ReLU [16] activations. and batch normalization for the convolution layers. The convolution layers have respectively [32, 64, 128, 256] channels.

6.3. Displacement Field Decoder

The displacement field decoder is composed of a cascade of four ResUpConv blocks detailed in Fig 6.3.1, and a convolution block OutConv.

6.3.1 ResUpConv

The ResUpConv block is the main component of our decoder. It fuses depth and guidance features at multiple scales using skip connections. The block architecture is detailed in Fig 10. Guidance features are refined using a 3x3 residual convolution layer [22] denoted ResConv3x3. All blocks use batch-normalization, Leaky ReLU [42] activation and filters weights are initialized using Xavier initialization.

6.3.2 Output layers

The final output block *OutConv* is composed of two Conv3x3 layers with batch-normalization and ReLU [16] activation, followed by a simple 3x3 convolution layer without batch-normalization nor activation. The number of channels of those layers are respectively 32, 16, and 2. Weights are initialized using Xavier initialization.

* Authors with equal contribution.

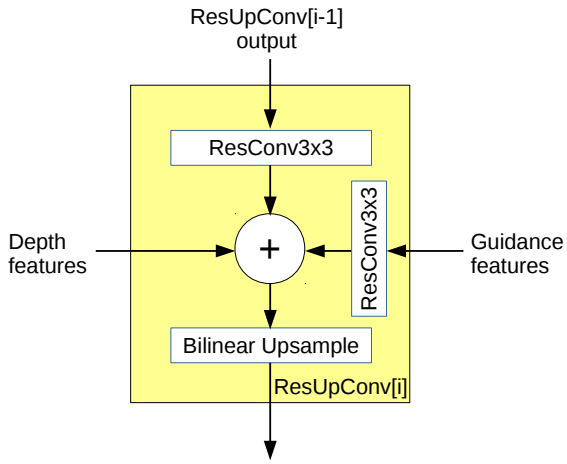


Figure 10. Details of the ResUpConv block used in our displacement field decoder.

7. Qualitative Results

7.1. Comparative Results on 2D Toy Problem

In Fig. 11, we show qualitative results on the 2D Toy Problem described in Section 3.3 of the main paper. One can see that residual update introduces severe artifacts around edges, producing large and spread out errors around them. Constrastingly, our proposed displacement update recovers sharp edges without degrading the rest of the image. To ensure fair comparison between residual and displacement update methods, identical CNN architectures were used for this experiment except for the last layer which predicts a 2-channel output for the displacement field instead of the 1D-channel output residual.

7.2. Comparative Results on NYUv2 Using Different MDE Methods

In Fig 12, we show qualitative results of our proposed refinement method on different MDE methods [7, 35, 10, 48, 63, 31] evaluated in the main paper. Our method always improves the sharpness of initial depth map prediction, without degrading the global depth reconstruction.

8. More NYUv2-OC++ Samples

In Fig. 13 we show several examples of our manually annotated 654 images NYUv2-OC++ dataset, which extends [48]. This dataset is based on the official test split of the popular NYUv2-Depth [52] depth estimation benchmark.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *PAMI*, 33(5):898–916, 2011. 3

[2] J.T. Barron and B. Poole. The Fast Bilateral Solver. In *ECCV*, 2016. 3, 7, 8

[3] J. Canny. A Computational Approach to Edge Detection. *PAMI*, 8(6), 1986. 6

[4] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu. Learning to Predict Crisp Boundaries. In *ECCV*, 2018. 3

[5] Piotr Dollár and C. Lawrence Zitnick. Structured Forests for Fast Edge Detection. In *ICCV*, 2013. 3

[6] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *ICCV*, 2015. 1, 2, 6, 8

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In *NIPS*, 2014. 1, 2, 6, 7, 8, 10, 14

[8] M. Everingham, L. Van Gool, C.K.I. Williams, J. M. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. 3

[9] H. Fan, H. Su, and L. J. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*, 2017. 6

[10] H. Fu, A. M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. In *CVPR*, 2018. 1, 2, 6, 7, 10, 14

[11] P. Fua. Combining Stereo and Monocular Information to Compute Dense Depth Maps That Preserve Depth Discontinuities. In *IJCAI*, pages 1292–1298, August 1991. 2

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision Meets Robotics: the KITTI Dataset. *International Journal of Robotics Research*, 2013. 2

[13] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and Binocular Stereo. *IJCV*, 14:211–226, 1995. 2

[14] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010. 9

[15] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2011. 7

[16] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, pages 315–323, 2011. 9

[17] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017. 1, 2

[18] S. Gupta, P. Arbelez, and J. Malik. Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images. In *CVPR*, 2013. 3

[19] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *ECCV*, 2014. 3

[20] K. He, J. Sun, and X. Tang. Guided Image Filtering. *PAMI*, 35:1397–1409, 06 2013. 3, 7, 8

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 2

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 9

[23] P. Heise, S. Klose, B. Jensen, and A. Knoll. PM-Huber: Patchmatch with Huber Regularization for Stereo Matching. In *ICCV*, 2013. 2

[24] M. Heo, J. Lee, K.-R. Kim, H.-U. Kim, and C.-S. Kim. Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement. In *ECCV*, 2018. 2

[25] D. Hoiem, A.A. Efros, and M. Hebert. Recovering Occlusion Boundaries from an Image. *IJCV*, 91(3), 2011. 3

[26] D. Hoiem, A. A. Efros, and M. Hebert. Recovering Surface Layout from an Image. *IJCV*, 75(1):151–172, October 2007. 3

[27] S.S. Intille and A.F. Bobick. Disparity-Space Images and Large Occlusion Stereo. In *ECCV*, 1994. 2

[28] H. Qi J. Dai, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable Convolutional Networks. In *ICCV*, 2017. 4

[29] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, 2015. 4

[30] J. Jeon and S. Lee. Reconstruction-Based Pairwise Depth Dataset for Depth Image Enhancement Using CNN. In *ECCV*, 2018. 2

[31] J. Jiao, Y. Cao, Y. Song, and R. W. H. Lau. Look Deeper into Depth: Monocular Depth Estimation with Semantic Booster and Attention-Driven Loss. In *ECCV*, 2018. 1, 2, 6, 7, 10, 14

[32] T. Kanade and M. Okutomi. A Stereo Matching Algorithm with an Adaptative Window: Theory and Experiment. *PAMI*, 16(9):920–932, September 1994. 2

[33] K. Karsch, Z. Liao, J. Rock, J. T. Barron, and D. Hoiem. Boundary Cues for 3D Object Shape Recovery. In *CVPR*, 2013. 1

[34] T. Koch, L. Liebel, F. Fraundorfer, and M. Körner. Evaluation of CNN-Based Single-Image Depth Estimation Methods. In *ECCV*, 2018. 2, 3, 6

[35] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *3DV*, 2016. 1, 2, 4, 6, 7, 8, 10, 14

[36] J.-H. Lee and C.-S. Kim. Monocular Depth Estimation Using Relative Depth Maps. In *CVPR*, June 2019. 2

[37] A. Levin, D. Lischinski, and Y. Weiss. Colorization Using Optimization. In *ACM Transactions on Graphics*, 2004. 6

[38] J. Li, R. Klein, and A. Yao. A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images. In *ICCV*, 2017. 6, 8

[39] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep Joint Image Filtering. In *ECCV*, 2016. 3

[40] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. PlaneNet: Piece-Wise Planar Reconstruction from a Single RGB Image. In *CVPR*, 2018. 6, 8

[41] F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *CVPR*, 2015. 6, 8

[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML*, 2013. 7, 9

[43] D. Martin, C. Fowlkes, and J Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *PAMI*, 26(5), 2004. 3

[44] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV*, 2001. 3

[45] B. Owen. A Robust Hybrid of Lasso and Ridge Regression. *Contemp. Math.*, 443, 2007. 2

[46] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in Pytorch. In *NIPS Workshop*, 2017. 5

[47] M. Poggi, A. F. Tosi, and S. Mattoccia. Learning Monocular Depth Estimation with Unsupervised Trinocular Assumptions. In *3DV*, pages 324–333, 2018. 1

[48] M. Ramamonjisoa and V. Lepetit. Sharpnet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. In *ICCV Workshop*, 2019. 1, 2, 3, 5, 6, 7, 8, 10, 14

[49] X. Ren and L. Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *NIPS*, 2012. 3

[50] X. Ren, C.C. Fowlkes, and J. Malik. Figure/ground Assignment in Natural Images. In *ECCV*, 2006. 3

[51] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. 2

[52] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012. 2, 3, 5, 10, 14, 15

[53] R. Szeliski and R. Weiss. Robust Shape Recovery from Occluding Contours Using a Linear Smoother. *IJCV*, 28(1):27–44, 1998. 1

[54] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *ICCV*, 1998. 7, 8

[55] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. SURGE: Surface Regularized Geometry Estimation from a Single Image. In *NIPS*, 2016. 2

[56] P. Wang and A. Yuille. DOC: Deep Occlusion Estimation from a Single Image. In *ECCV*, 2016. 3

[57] H. Wu, S. Zheng, J. Zhang, and K. Huang. Fast End-To-End Trainable Guided Filter. In *CVPR*, 2018. 3, 7, 8

[58] J. Xie, R. B. Girshick, and A. Farhadi. Deep3D: Fully Automatic 2D-To-3d Video Conversion with Deep Convolutional Neural Networks. In *ECCV*, 2016. 1

[59] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-Scale Continuous CRFs as Sequential Deep Networks for Monocular Depth Estimation. In *CVPR*, 2017. 3

[60] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Monocular Depth Estimation Using Multi-Scale Continuous CRFs as Sequential Deep Networks. *PAMI*, 2018. 3

[61] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. LEGO: Learning Edge with Geometry All at Once by Watching Videos. In *CVPR*, 2018. 2

[62] Z. Yang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised Learning of Geometry from Videos with Edge-Aware Depth-Normal Consistency. In *AAAI*, 2018. 2

[63] W. Yin, Y. Liu, C. Shen, and Y. Yan. Enforcing Geometric Constraints of Virtual Normal for Depth Prediction. In *ICCV*, 2019. 6, 7, 8, 10, 14

[64] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang. Pattern-Affinitive Propagation Across Depth, Surface Normal and Semantic Segmentation. In *CVPR*, 2019. 2

[65] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *CVPR*, 2017. 5

[66] L. Zwald and S. Lambert-Lacroix. The Berhu Penalty and the Grouped Effect. 2012. 2

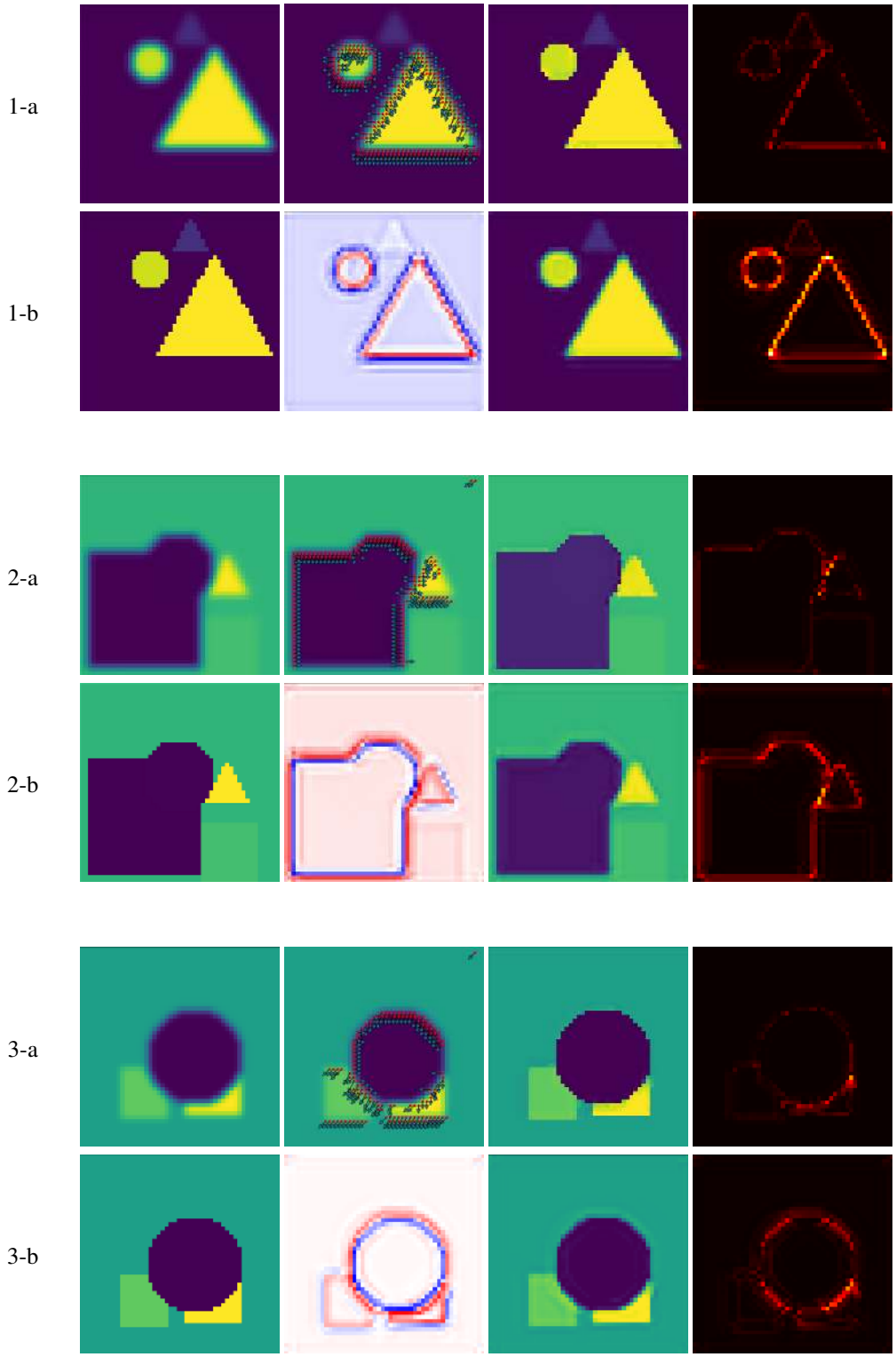
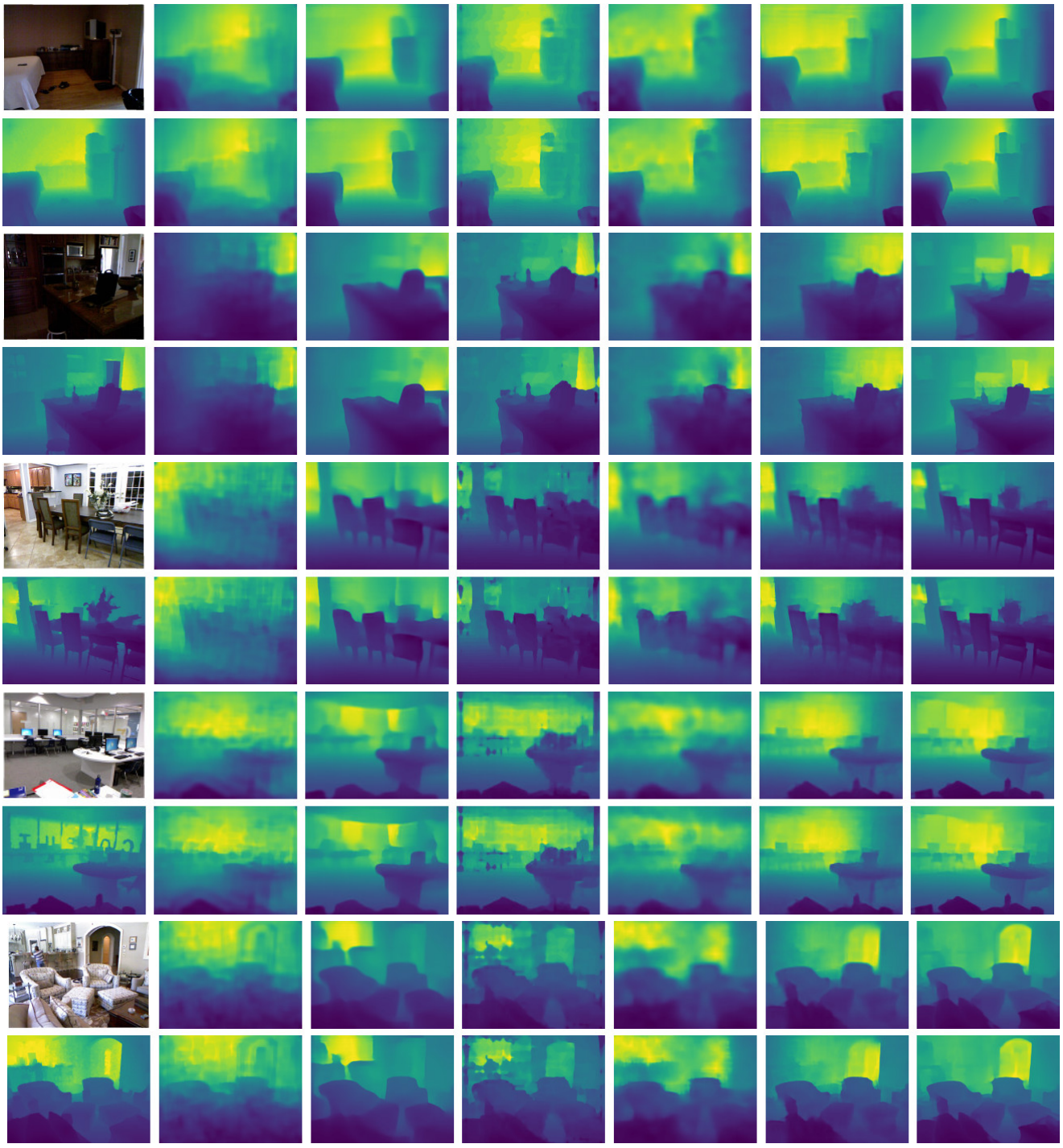


Figure 11. Comparison between residual and displacement learning on a toy image sharpening problem for three examples. A blurred input image $\tilde{I}$ is fed through a Convolutional Neural Network which learns to reconstruct the original clean image I. Lines (1,2,3)-a show from left to right: the input image, samples of the dense predicted displacement field, refinement result with displacement update, error map. Lines (1,2,3)-b show, from left to right: the ground truth image, the predicted residual (blue is negative, red is positive, white is zero), refinement result with residual update, error map. While introducing artifacts, residual learning also results in a spread out error map around edges.



RGB image
GT depth | Eigen *et al.* [7] | Laina *et al.* [35] | Fu *et al.* [10] | Jiao *et al.* [31] | Ramamonjisoa & Lepetit [48] | Yin *et al.* [63]

Figure 12. Refinement results using our method (best seen in color). Each example is represented on two rows, first row being the original predicted depth and second row being the refined depth. First column shows the RGB input images and associated ground truth depth from NYUv2 [52]. Following columns are refinement results for different methods we evaluated.

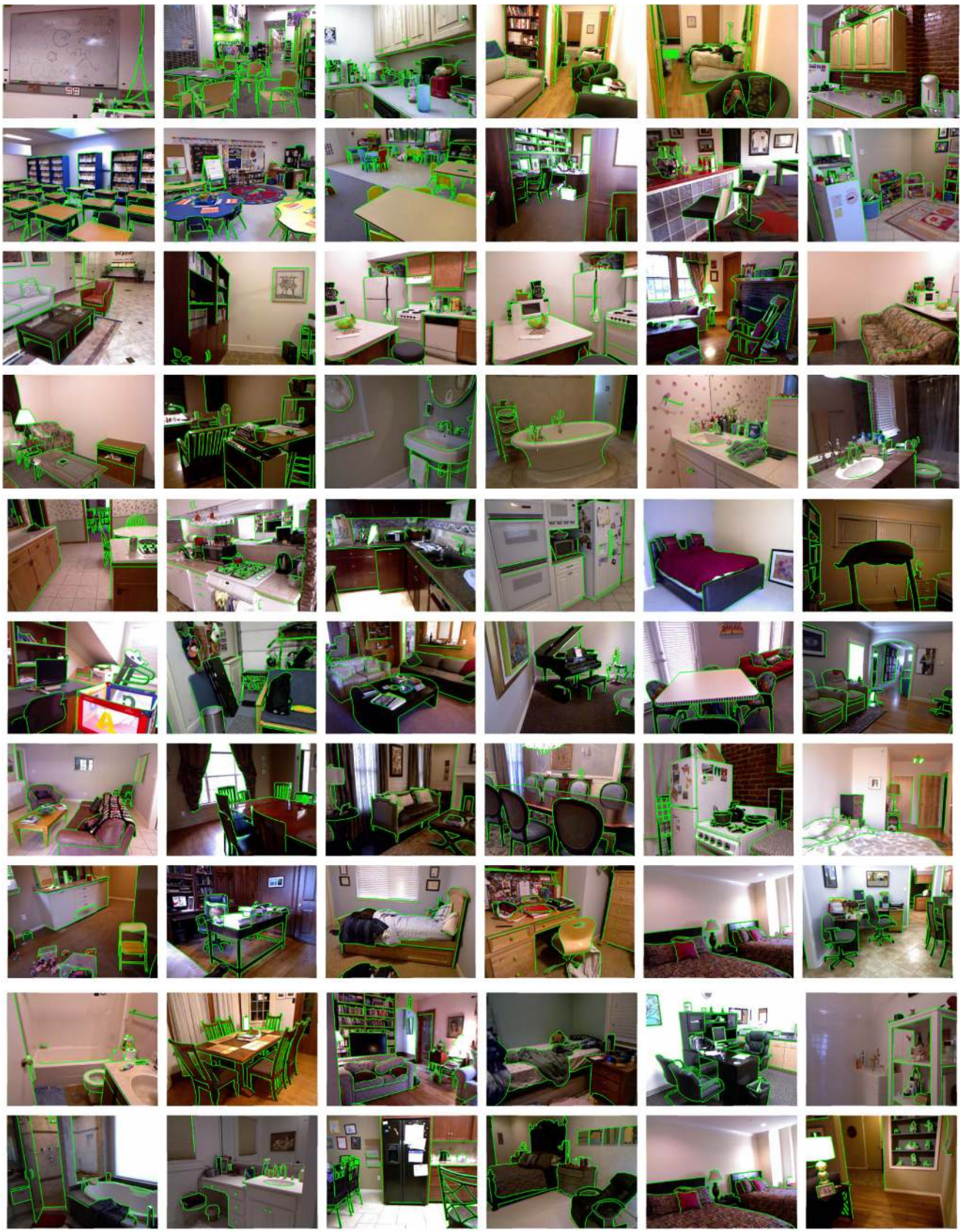


Figure 13. Samples taken from our fine-grained manually annotated NYUv2-OC++, which add occlusion boundaries to the popular NYUv2-Depth [52] benchmark. We annotated the full official 654 images test set of NYUv2-Depth.