Rethinking Inductive Biases for Surface Normal Estimation

Gwangbin Bae Andrew J. Davison
Dyson Robotics Lab, Imperial College London
{g.bae, a.davison}@imperial.ac.uk

Figure 1. **Examples of challenging in-the-wild images and their surface normals predicted by our method.**

Abstract

Despite the growing demand for accurate surface normal estimation models, existing methods use general-purpose dense prediction models, adopting the same inductive biases as other tasks. In this paper, we discuss the inductive biases needed for surface normal estimation and propose to (1) utilize the per-pixel ray direction and (2) encode the relationship between neighboring surface normals by learning their relative rotation. The proposed method can generate crisp — yet, piecewise smooth — predictions for challenging in-the-wild images of arbitrary resolution and aspect ratio. Compared to a recent ViT-based state-of-the-art model, our method shows a stronger generalization ability, despite being trained on an orders of magnitude smaller dataset. The code is available at https://github.com/baegwangbin/DSINE.

1. Introduction

We address the problem of estimating per-pixel surface normal from a single RGB image. This task, unlike monocular depth estimation, is not affected by scale ambiguity and has a compact output space (a unit sphere vs. positive real value), making it feasible to collect data that densely covers the output space. As a result, learning-based surface normal estimation methods show strong generalization capability for out-of-distribution images, despite being trained on relatively small datasets [2].

Despite their essentially local property, predicted surface normals contain rich information about scene geometry. In recent years, their usefulness has been demonstrated for various computer vision tasks, including image generation [62], object grasping [60], multi-task learning [36], depth estimation [3, 41], simultaneous localization and mapping [63], human body shape estimation [5, 54, 55], and CAD model alignment [33]. However, despite the growing demand for accurate surface normal estimation models, there has been little discussion on the right inductive biases needed for the task.

State-of-the-art surface normal estimation methods [2, 14, 29, 56] use general-purpose dense prediction models, adopting the same inductive biases as other tasks (e.g. depth estimation and semantic segmentation). For example, CNN-based models [2, 14] assume translation equivariance and use the same set of weights for different parts of the image. While such weight-sharing can improve sample efficiency [34], it is sub-optimal for surface normal estimation as a pixel's *ray direction* provides important cues and constraints for its surface normal. This has limited the accu-

racy of the prediction and the ability to generalize to images taken with out-of-distribution cameras.

Another important aspect of surface normal estimation overlooked by existing methods is that there are common typical relationships between the normals at nearby image pixels. It is well understood that many 3D objects in a scene are piece-wise smooth [24] and that neighboring normals often have similar values. There is also a very frequently occurring relationship between groups of nearby pixels on two surfaces in contact, or between groups of pixels on a continuously curving surface: their normals are related by a rotation through a certain angle, about an axis lying within the surface and which is sometimes visible in the image as an edge.

In this paper, we provide a thorough discussion of the inductive biases needed for deep learning-based surface normal estimation and propose three architectural changes to incorporate such biases:

- We supply *dense pixel-wise ray direction* as input to the network to enable camera intrinsics-aware inference and hence improve the generalization ability.
- We propose a *ray direction-based activation function* to ensure the visibility of the prediction.
- We recast surface normal estimation as *rotation estimation*, where the relative rotation with respect to the neighboring pixels is estimated in the form of axis-angle representation. This allows the model to generate predictions that are piece-wise smooth, yet crisp at the intersection between surfaces.

The proposed method shows strong generalization ability. It can generate highly detailed predictions even for challenging in-the-wild images of arbitrary resolution and aspect ratio (see Fig. 1). We outperform a recent ViT-based state-of-the-art method [14, 29] — both quantitatively and qualitatively — despite being trained on an orders of magnitude smaller dataset.

2. Related work

Hoiem et al. [22, 23] were among the first to propose a learning-based approach for monocular surface normal estimation. The output space was discretized and handcrafted features were extracted to classify the normals. Fouhey et al. [18] took a different approach and tried to detect geometrically informative *primitives* from data. For detected primitives, the normal maps of the corresponding training patches were aligned to recover a dense prediction. Another common approach was to assume a Manhattan World [10] to adjust the initial prediction [18] or generate candidate normals from pairs of vanishing points [19].

Following the success of deep convolutional neural networks in image classification [32], many deep learning-based methods [4, 15, 53] were introduced. Since then, notable contributions have been made by exploiting the surface normals computed from Manhattan lines [51], introducing a spatial rectifier to handle tilted images [12], and estimating the aleatoric uncertainty to improve the performance on small structures and near object boundaries [2].

Eftekhar et al. [14] trained a U-Net [45] on more than 12 million images covering diverse scenes and camera intrinsics. They recently released an updated model by training a transformer-based model [42] with sophisticated 3D data augmentation [29] and cross-task consistency [59]. This model is the current state-of-the-art in surface normal estimation and will be the main comparison for our method.

3. Inductive bias for surface normal estimation

In this section, we discuss the inductive biases needed for surface normal estimation. Throughout the rest of this paper, we use the right-hand convention for camera-centered coordinates, where the X, Y, and Z axes point right, down, and front, respectively.

3.1. Encoding per-pixel ray direction

Under perspective projection, each pixel is associated with a ray that passes through the camera center and intersects the image plane at the pixel. Assuming a pinhole camera, a ray of unit depth for a pixel at (u, v) can be written as

$$\mathbf{r}(u,v) = \begin{bmatrix} \frac{u-c_u}{f_u} & \frac{v-c_v}{f_v} & 1 \end{bmatrix}^\intercal, \tag{1}$$

where f_u and f_v are the focal lengths and (c_u, c_v) are the pixel coordinates of the principal point.

Per-pixel ray direction is essential for surface normal estimation. For rectangular structures (e.g. buildings), we can identify sets of parallel lines and their respective vanishing points. The ray direction at the vanishing point then gives us the 3D orientation of the lines and hence the surface normals [21]. Early works on single-image 3D reconstruction [19, 25, 31, 35] made explicit use of such cues.

Now consider an occluding boundary created by a smooth (i.e. infinitely differentiable) surface. As Marr [38] pointed out, the surface normals at an occluding boundary can be determined uniquely by forming a generalized cone (whose apex is at the camera center) that intersects the image plane at the boundary. In other words, the normals at the boundary should be perpendicular to the ray direction (see Fig. 2-a). Such insights have been widely adopted for under-constrained 3D reconstruction tasks such as single-image shape from shading [28].

Lastly, the ray direction decides the range of normals that would be *visible* in that pixel, effectively halving the output space (see Fig. 2-b). This is analogous to the case of depth estimation, where the output should be positive. Such an inductive bias is often adopted by interpreting the network output as log depth [16] or by using a ReLU activation [61].

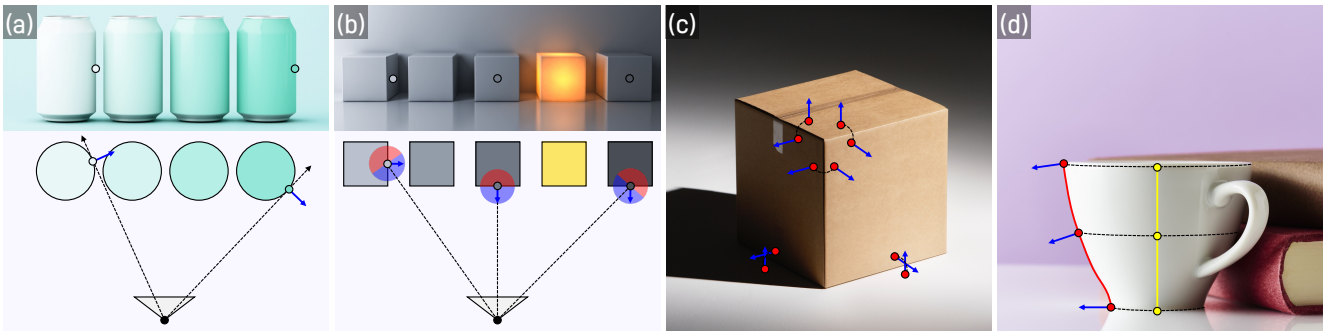

Figure 2. **Motivation.** In this paper, we propose to utilize the *per-pixel ray direction* and estimate the surface normals by learning the *relative rotation between nearby pixels*. **(a)** Ray direction serves as a useful cue for pixels near occluding boundaries as the normal should be perpendicular to the ray. **(b)** It also gives us the range of normals that would be visible, effectively halving the output space. **(c)** The surface normals of certain scene elements — in this case, the floor — may be difficult to estimate due to the lack of visual cues. Nonetheless, we can infer their normals by learning the pairwise relationship between nearby normals (e.g. which surfaces should be perpendicular). **(d)** Modeling the relative change in surface normals is not just useful for flat surfaces. In this example, the relative angle between the normals of the yellow pixels can be inferred from that of the red pixels assuming circular symmetry.

Despite the aforementioned usefulness, state-of-the-art methods [2, 14] do not encode the ray direction and use CNNs with translational weight sharing, preventing the model from learning ray direction-aware inference. While recent transformer-based models [29, 56] have the capability of encoding the ray direction in the form of learned positional embedding, it is not trivial to inter/extrapolate the positional embedding when testing the model on images taken with out-of-distribution intrinsics.

3.2. Modeling inter-pixel constraints

Consider a pixel i and its neighboring pixel j. Since their surface normals have unit length and share the same origin (camera center), they are related by a 3D rotation matrix $R \in SO(3)$. While there are different ways to parameterize R, we choose the *axis-angle* representation, $\boldsymbol{\theta} = \theta \mathbf{e}$, where a unit vector $\mathbf{e}$ represents the axis of rotation and θ is the angle of rotation. Then, the exponential map $\exp : \mathfrak{so}(3) \rightarrow SO(3)$ — which is readily available in modern deep learning libraries [1, 43] — can map $\boldsymbol{\theta}$ back to R.

Within flat surfaces (which are prevalent in man-made scenes/objects), θ would be zero and R would simply be the identity. In a typical indoor scene, the surfaces of objects are often perpendicular or parallel to the ground plane, creating lines across which the normals should rotate by $90°$ (see Fig. 2-c). For a curved surface, the relative angle between the pixels can be inferred from the occluding boundaries by assuming a certain level of symmetry (see Fig. 2-d).

But why should we learn R instead of directly estimating the normals? Firstly, learning the relative rotation is much easier, as the angle between the normals, unlike the normals themselves, is independent of the viewing direction (it is also zero — or close to zero — for most pixel pairs). Finding the axis of rotation is also straightforward. When two (locally) flat surfaces intersect at a line, the normals rotate around that intersection. As the image intensity generally changes sharply near such intersections, the task can be as simple as edge detection.

Secondly, the estimated rotation can help improve the accuracy for surfaces with limited visual cues. For instance, while it is difficult to estimate the normal of a texture-less surface, the objects that are in contact with the surface can provide evidence for its normal (see Fig. 2-c).

Lastly, as long as the relative rotations between the normals are captured correctly, any misalignment between the prediction and the ground truth can be resolved via a single global rotation. For example, in the case of a flat surface, estimating inaccurate but constant normals is better than estimating accurate but noisy normals, as the orientation can easily be corrected (e.g. via sparse depth measurements or visual odometry). This is again analogous to depth estimation where a relative depth map is easier to learn and can be aligned via a global scaling factor.

4. Our approach

From Sec. 4.1 to 4.3, we explain how a dense prediction network can be modified to encode the inductive biases discussed in Sec. 3. We then explain the network architecture and our training dataset in Sec. 4.4 and 4.5.

4.1. Ray direction encoding

To avoid having to learn ray direction-aware prediction, one can crop and zero-pad the images such that the principal point is at the center and the field of view is always $\theta°$, where θ is set to some high value. Then, a pixel at (u, v) will always have the same ray direction, allowing the network to encode it, e.g., in the form of position embedding. However, such an approach (1) wastes the compute for the

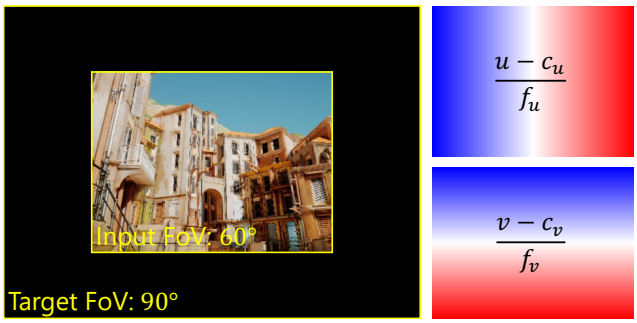

Figure 3. **Encoding camera intrinsics. (left)** To avoid having to learn camera intrinsics-aware prediction, one can zero-pad or crop the images such that they always have the same intrinsics. **(right)** Instead, we compute the focal length-normalized image coordinates and provide them as additional input to the network.

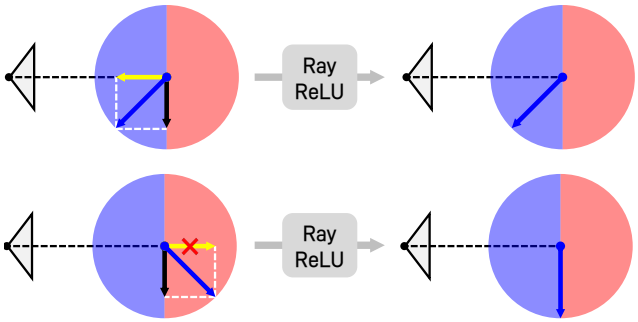

Figure 4. **Ray ReLU activation.** An important constraint for surface normal estimation is that the predicted normal should be *visible*. We achieve this by zeroing out the component that is in the direction of the ray.

zero-padded regions, (2) loses high-frequency details from downsampling, and (3) cannot be applied to images with wider field-of-view. Instead, we compute the focal length-normalized image coordinates — i.e. Eq. 1 — and provide this as an additional input to the intermediate layers of the network (see Fig. 3). This encoding is similar to that of CAM-Convs [17] which was designed for depth estimation. Unlike [17], we do not encode the image coordinates themselves and only encode the ray direction.

4.2. Ray direction-based activation

A surface that is facing *away* from the camera would simply not be visible in the image. An important constraint for surface normal estimation should thus be that the angle between the ray direction and the estimated normal vector must be greater than 90°. To incorporate such a bias, we propose a ray direction-based activation function analogous to ReLU. Given the estimated normal $\mathbf{n}$ and ray direction $\mathbf{r}$ (both are normalized), the activation can be written as

$$\sigma_{\text{ray}}(\mathbf{n}, \mathbf{r}) := \frac{\mathbf{n} + (\min(0, \mathbf{n} \cdot \mathbf{r}) - \mathbf{n} \cdot \mathbf{r})\,\mathbf{r}}{\|\mathbf{n} + (\min(0, \mathbf{n} \cdot \mathbf{r}) - \mathbf{n} \cdot \mathbf{r})\,\mathbf{r}\|}. \quad (2)$$

Eq. 2 ensures that $\mathbf{n} \cdot \mathbf{r} = \cos\theta$ (i.e. the magnitude of $\mathbf{n}$ along $\mathbf{r}$) is less than or equal to zero. The rectified normal is then re-normalized to have a unit length. This is illustrated in Fig. 4.

4.3. Recasting surface normal estimation as rotation estimation

For pixel i, we can define its local neighborhood $\mathcal{N}_i = \{j : |u_i - u_j| \leq \beta \text{ and } |v_i - v_j| \leq \beta\}$. We can then learn the pairwise relationship between the surface normals $\mathbf{n}_i$ and $\mathbf{n}_j$ in the form of a rotation matrix R_{ij}.

For each pair of pixels, three quantities should be estimated: First is the angle θ_{ij} between the two normals. This is easy to learn as θ_{ij} is independent of the viewing direction and is 0° or 90° for many pixel pairs. Secondly, we need to estimate the axis of rotation $\mathbf{e}_{ij}$ (i.e. a 3D unit vector around which the normals rotate). While directly learning $\mathbf{e}_{ij}$ requires complicated 3D reasoning, we propose a simpler approach that only requires 2D information.

Let's first consider the case where $\mathbf{n}_i$ and $\mathbf{n}_j$ are on the same smooth surface. As the surface can locally be approximated as a plane, the angle should be close to zero. In such a case, finding the axis is less important as the rotation matrix will be close to the identity.

Another possibility is that the two points are on different smooth surfaces that are intersecting with each other. In this case, we only need to estimate the *2D projection* of $\mathbf{e}_{ij}$. Suppose that this 2D projection is a vector whose endpoints are (u_j, v_j) and $(u_j + \delta u_{ij}, v_j + \delta v_{ij})$. The 3D vector $\mathbf{e}_{ij}$ should then lie on a plane that passes through the camera center and the two endpoints (see Fig 5-right). Formally, this can be written as,

$$\begin{aligned}\mathbf{e}_{ij} &= X^c(u_j + \delta u_{ij}, v_j + \delta v_{ij}) - X^c(u_j, v_j) \\ &= \begin{bmatrix} (z_j + \delta z_{ij}) \cdot \frac{u_j + \delta u_{ij} - c_u}{f_u} - z_j \cdot \frac{u_j - c_u}{f_u} \\ (z_j + \delta z_{ij}) \cdot \frac{v_j + \delta v_{ij} - c_v}{f_v} - z_j \cdot \frac{v_j - c_v}{f_v} \\ \delta z_{ij} \end{bmatrix}, \end{aligned} \quad (3)$$

where $X^c(\cdot, \cdot)$ are the camera-centered coordinates corresponding to the pixel and δz represents the change in depth. There are two unknowns in Eq. 3: z and δz. The first constraint for solving Eq. 3 is that $\|\mathbf{e}_{ij}\| = 1$. We can then assume that the surface normal $\mathbf{n}_j$ is perpendicular to $\mathbf{e}_{ij}$ (i.e. $\mathbf{n}_j \cdot \mathbf{e}_{ij} = 0$), which is true for pixels near the intersection between two (locally) flat surfaces. Such an approach is appealing for 2D CNNs — whose initial layers are known to be oriented edge filters — as the image intensity tends to change sharply near the intersection between two surfaces.

The final remaining possibility for $\mathbf{n}_i$ and $\mathbf{n}_j$ is that they are on two disconnected surfaces or on non-smooth sur-

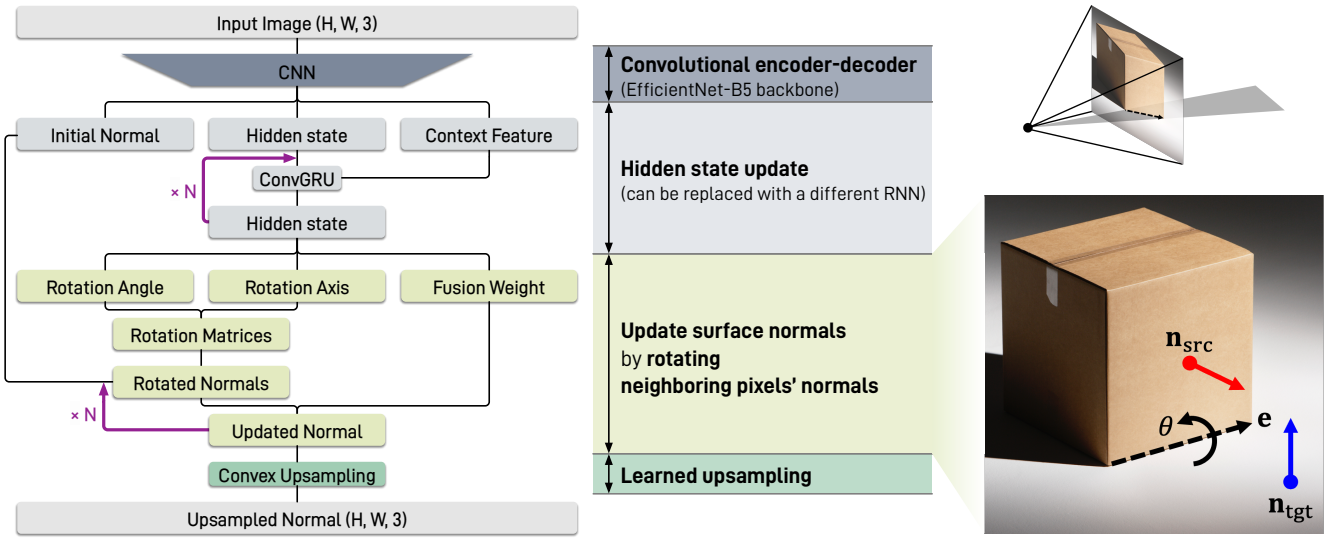

Figure 5. **Network architecture.** A lightweight CNN extracts a low-resolution feature map, from which the initial normal, hidden state and context feature are obtained. The hidden state is then recurrently updated using a ConvGRU [9] unit. From the updated hidden state, we estimate three quantities: rotation angle and axis to define a pairwise rotation matrix for each neighboring pixel; and a set of weights that will be used to fuse the rotated normals.

faces. As estimating R_{ij} in such a case is significantly more challenging than the other two scenarios, we choose to down-weight them with a set of weights $\{w_{ij}\}$, which is the last quantity we learn. The updated normal of pixel i can be written as

$$\mathbf{n}_i^{t+1} = \frac{\sum_j w_{ij}\sigma_{\text{ray}}(R_{ij}\mathbf{n}_j^t, \mathbf{r}_i)}{\|\sum_j w_{ij}\sigma_{\text{ray}}(R_{ij}\mathbf{n}_j^t, \mathbf{r}_i)\|} \qquad (4)$$
$$R_{ij} = \exp\left(\theta_{ij}[\mathbf{e}_{ij}]_\times\right).$$

where the ray-ReLU activation, introduced in Sec. 4.2, is used to ensure that the rotated normals are in the visible range for the target pixel i. We also added a superscript for the normals to represent an iterative update.

To summarize, given some initial surface normal prediction $\mathbf{n}^t$, the network should estimate the following three quantities — for each pixel i — in order to obtain the updated normal map $\mathbf{n}^{t+1}$:

- The rotation angles $\{\theta_{ij}\}$ for the neighboring pixels. For the output, we use a sigmoid activation followed by a multiplication of π.
- 2D unit vectors $\{(\delta u_{ij}, \delta v_{ij})\}$ whose orientation represents the 2D projection of the rotation axes $\{\mathbf{e}_{ij}\}$. We use two output channels followed by an L2 normalization. This, combined with $\{\mathbf{n}_j^t\}$ gives us the axes of rotation.
- The weights $\{w_{ij}\}$ to fuse the rotated normals.

The process is then repeated for N_{iter} times. In the following section, we explain how such inference can be done in a convolutional recurrent neural network.

4.4. Network architecture

The components described in Sec. 4.1-4.3 are general and can be adopted by most dense prediction neural networks with minimal architectural changes. We use a light-weight CNN with a bottleneck recurrent unit (see Fig. 5). The architecture is the same as that of [3] except for the quantities that are estimated from the updated hidden state.

The initial prediction and the hidden state have the resolution of $(H/8 \times W/8)$, where H and W are the input height and width. Updating the normals in a coarse resolution allows us to model long-range relationships with small compute. We set the neighborhood size β (mentioned in Sec. 4.3) to 2 (i.e. 5×5 neighborhood). The number of surface normal updates N_{iter} is set to 5, as it gave a good balance between accuracy and computational efficiency. As a result, each forward pass returns $N_{\text{iter}} + 1$ predictions (initial prediction obtained via direct regression + N_{iter} updates). We then apply convex upsampling [50] to recover full-resolution outputs (more details regarding the network architecture are provided in the supplementary material). The network is trained by minimizing the weighted sum of their angular losses. The loss for pixel i can be written as

$$\mathcal{L}_i = \sum_{t=0}^{N_{\text{iter}}} \gamma^{N_{\text{iter}}-t} \cos^{-1}(\mathbf{n}_i^{\text{gt}} \cdot \mathbf{n}_i^t) \qquad (5)$$

where $0 < \gamma < 1$ puts a bigger emphasis on the final prediction. We set $\gamma = 0.8$ following RAFT [50].

Dataset	Train		Val	
	# scenes	# imgs	# scenes	# imgs
Cleargrasp [46]	9	900	9	45
3D Ken Burns [39]	23	4600	23	230
Hypersim [44]	407	38744	407	2035
SAIL-VOS 3D [26]	170	16262	170	850
TartanAir [52]	16	3200	16	160
MVS-Synth [27]	120	11400	120	600
BlendedMVG [57]	495	44070	7	35
Taskonomy* [58]	375	37500	73	365
Replica* [49]	10	1000	4	20
Replica + GSO* [14, 49]	30	3000	12	60
Total	1655	160676	841	4400

Table 1. **Dataset statistics.** We created a small meta-dataset that covers diverse scenes (*: downloaded from Omnidata [14]).

4.5. Dataset

The proposed model is designed to have high sample efficiency. Firstly, we use a fully convolutional design to allow translational weight sharing, which is known to improve the sample efficiency [13]. Secondly, we estimate the rotation matrices by decomposing them into *angles* and *axes*. The angle between two normals is unaffected by the camera pose. While the axis of rotation does change with the camera pose, we estimate the *2D orientation* of its projection on the image plane, which can be as simple as edge detection (its 3D orientation is then recovered from the surface normal). For such reasons, we do not need a large number of images from the same scene. Rendering synthetic scenes with diverse camera intrinsics is also unnecessary as the intrinsics are explicitly encoded in the input.

To this end, we created a small meta-dataset consisting of images extracted from 10 RGB-D datasets (see Tab. 1 for dataset composition). Our dataset, compared to Omnidata [14], has a similar number of scenes (1655 vs. 1905) but a significantly smaller number of images (160K vs. 12M).

5. Experiments

After providing details regarding the experimental setup (Sec. 5.1), we compare the generalization capability of our method to that of the state-of-the-art methods (Sec. 5.2) and perform an ablation study to demonstrate the effectiveness of the proposed usage of additional inductive biases (Sec. 5.3).

5.1. Experimental setup

Evaluation protocol. We measure the angular error for the pixels with ground truth and report the mean and median (lower the better). We also report the percentage of pixels with an error below $t \in [5.0°, 7.5°, 11.25°, 22.5°, 30.0°]$ (higher the better).

Data preprocessing. The training images are randomly resized and cropped with a random aspect ratio to facilitate the learning of ray direction-aware prediction. As many of our training images are synthetic, we also add aggressive 2D data augmentation — e.g., Gaussian blur, Gaussian noise, motion blur, and color — to minimize the domain gap. Full details regarding data preprocessing are provided in the supplementary material.

Implementation details. Our model is implemented in PyTorch [40]. In all our experiments, the network and its variants are trained on our meta-dataset (Sec. 4.5) for five epochs. We use the AdamW optimizer [37] and schedule the learning rate using 1cycle policy [48] with $lr_{\text{max}} = 3.5 \times 10^{-4}$. The batch size is set to 4 and the gradients are accumulated every 4 batches. The training approximately takes 12 hours on a single NVIDIA 4090 GPU.

5.2. Comparison to the state-of-the-art

We select six datasets to compare our method's generalization capability against the state-of-the-art methods. NYUv2 [47], ScanNet [11], and iBims-1 [30] all contain images of real indoor scenes captured with cameras of standard intrinsics (480 $\times$ 640 resolution and approximately 60° field of view). Generalizing to such datasets is straightforward as the scenes and the cameras are similar to those of commonly-used training datasets (e.g. taskonomy [58]). While our method outperforms other methods on most metrics, the improvement is relatively small for this reason.

On the contrary, Sintel [6] and Virtual KITTI [20] contain highly dynamic outdoor scenes and have less common fields of views (e.g. 18° to 83° for Sintel). The aspect ratios — 436×1024 and 375×1242, respectively — are also out of distribution. For such datasets, our approach significantly outperforms the other methods across all metrics. This is mainly due to the explicit encoding of the ray direction in the input.

Lastly, we evaluate the methods on the validation set of OASIS [8], which contains 10,000 in-the-wild images collected from the internet. Two things should be noted for this dataset. Firstly, the ground truth surface normals only exist for small patches of the images and are annotated by *humans*. Plus, the ground truth is generally available only for large flat regions. The accuracy metrics thus do not faithfully represent the performance of the methods. Secondly, unlike most RGB-D datasets, the camera intrinsics are not available for the input images. We thus approximated the intrinsics by using the focal length recorded in the image metadata, and assuming that the principal point is at the center and that there is zero distortion. Despite such approximation, our method performs on par with the other methods.

For OASIS, we provide a qualitative comparison against Omnidata v2 [29] in Fig. 6. While the quantitative accuracy was better for [29], the predictions made by our method

Method	NYUv2 [47]							ScanNet [11]							iBims-1 [30]						
	mean	med	5.0°	7.5°	11.25°	22.5°	30°	mean	med	5.0°	7.5°	11.25°	22.5°	30°	mean	med	5.0°	7.5°	11.25°	22.5°	30°
OASIS [8]	29.2	23.4	7.5	14.0	23.8	48.4	60.7	32.8	28.5	3.9	8.0	15.4	38.5	52.6	32.6	24.6	7.6	13.8	23.5	46.6	57.4
EESNU [2]	**16.2**	8.5	**32.8**	46.0	58.6	77.2	**83.5**	11.8	5.7	45.2	59.7	71.3	85.5	89.9	20.0	8.4	32.0	46.1	58.5	73.4	78.2
Omnidata v1 [14]	23.1	12.9	21.6	33.4	45.8	66.3	73.6	22.9	12.3	21.5	34.5	47.4	66.1	73.2	19.0	7.5	37.2	50.0	62.1	76.1	80.1
Omnidata v2 [29]	17.2	9.7	25.3	40.2	55.5	76.5	83.0	**16.2**	8.5	29.1	44.9	60.2	**79.5**	**84.7**	18.2	7.0	38.9	52.2	63.9	77.4	81.1
Ours	16.4	**8.4**	**32.8**	**46.3**	**59.6**	**77.7**	**83.5**	**16.2**	**8.3**	**29.8**	**45.9**	**61.0**	78.7	84.4	**17.1**	**6.1**	**43.6**	**56.5**	**67.4**	**79.0**	**82.3**

Method	Sintel [6]							Virtual KITTI [20]							OASIS [8]						
	mean	med	5.0°	7.5°	11.25°	22.5°	30°	mean	med	5.0°	7.5°	11.25°	22.5°	30°	mean	med	5.0°	7.5°	11.25°	22.5°	30°
OASIS [8]	43.1	39.5	1.4	3.1	7.0	24.1	35.7	41.8	34.6	2.7	10.1	23.6	40.8	46.7	23.9	18.2	-	-	31.2	59.5	71.8
EESNU [2]	42.1	36.5	3.0	6.1	11.5	29.8	41.2	51.9	53.3	1.3	4.5	14.9	29.1	34.0	27.7	21.0	-	-	24.0	53.2	66.6
Omnidata v1 [14]	41.5	35.7	3.0	5.8	11.4	30.4	42.0	41.2	34.0	21.5	29.3	34.7	43.0	47.6	24.9	**18.0**	-	-	**31.0**	59.5	71.4
Omnidata v2 [29]	40.5	35.1	4.6	7.9	14.7	33.0	43.5	37.5	27.4	30.7	36.1	39.7	47.1	51.5	**24.2**	18.2	-	-	27.7	**61.0**	**74.2**
Ours	**34.9**	**28.1**	**8.9**	**14.1**	**21.5**	**41.5**	**52.7**	**28.9**	**9.9**	**43.7**	**47.5**	**51.3**	**59.2**	**63.2**	24.4	18.8	**10.5**	**17.5**	28.8	58.5	72.0

Table 2. **Quantitative evaluation of the generalization capabilities possessed by different methods.** For each metric, the best results are colored in green. For evaluation on [6, 11, 20, 30, 47], we used the official code and model weights to generate predictions and measured their accuracies. For methods that assume a specific aspect ratio and resolution, the images were zero-padded and resized accordingly to match the requirements. The numbers in red mean that the method was trained on the same dataset. We excluded such methods in ranking to ensure a fair comparison.

Figure 6. **Comparison to Omnidata v2 [14] (DPT [42] model trained on 12 million images using 3D data augmentation [29] and cross-task consistency [59]).** Our method shows a stronger generalization capability for challenging in-the-wild objects. For texture-less regions (e.g. sky in the fourth column), our model resolves any inconsistency in the prediction and outputs a flat surface, while preserving sharp boundaries around other objects.

show a significantly higher level of detail.

One notable advantage of our method over ViT-based models (e.g. [29]) lies in the simplicity and efficiency of network training. For example, Omnidata v2 [29] was trained for 2 weeks on four NVIDIA V100 GPUs. A set of sophisticated 3D data augmentation functions [29] were used to improve the generalization performance and cross-task consistency [59] was enforced by utilizing other ground truth labels. On the contrary, our model can be trained in just 12 hours on a single NVIDIA 4090 GPU, does not require geometry-aware 3D augmentations, and does not require any additional supervisory signal. Our model also has 40% fewer parameters compared to [29] (72M vs 123M).

5.3. Ablation study

We now run an ablation study to examine the effectiveness of the proposed usage of two new inductive biases — utilizing dense per-pixel ray direction and modeling the pair-

Method	NYUv2 [47]				ScanNet [11]				iBims-1 [30]				Sintel [6]				Virtual KITTI [20]			
	mean	11.25°	22.5°	30°	mean	11.25°	22.5°	30°	mean	11.25°	22.5°	30°	mean	11.25°	22.5°	30°	mean	11.25°	22.5°	30°
baseline	16.6	59.1	76.8	82.9	16.5	60.5	77.7	83.5	18.0	66.0	77.7	81.2	36.6	18.4	38.3	50.0	30.5	48.0	55.9	60.6
baseline + ray	16.6	59.2	77.0	82.9	16.4	61.2	77.9	83.6	17.6	66.4	78.1	81.4	36.0	19.8	38.8	50.4	29.4	50.6	58.4	62.4
baseline + ray + rot	**16.4**	**59.6**	**77.7**	**83.5**	**16.2**	**61.0**	**78.7**	**84.4**	**17.1**	**67.4**	**79.0**	**82.3**	**34.9**	**21.5**	**41.5**	**52.7**	**28.9**	**51.3**	**59.2**	**63.2**

Table 3. **Ablation study - quantitative results.** Adding per-pixel ray direction as input (+ ray) and updating the initial prediction via iterative rotation estimation (+ rot) both lead to an overall improvement in the metrics. The benefit of using ray direction encoding is clearer for datasets with out-of-distribution camera intrinsics (Sintel and Virtual KITTI).

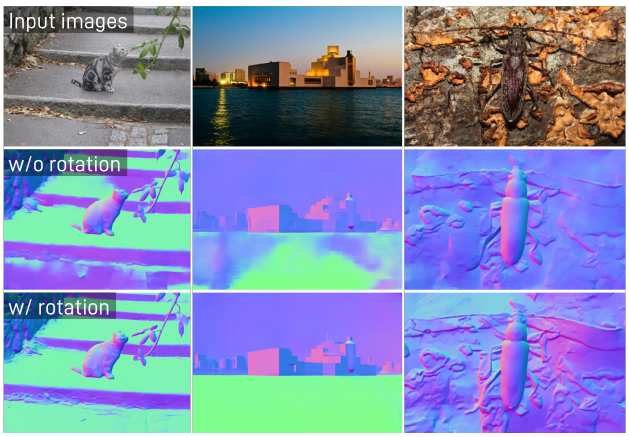

Figure 7. **Ablation study - qualitative results.** Here we compare the predictions made by two models with and without the iterative rotation estimation (both are using per-pixel ray information). When the model is trained to directly estimate the normals, the prediction is often inconsistent within smooth surfaces, leading to *bleeding* artifacts. The proposed refinement via rotation estimation leads to piece-wise smooth surfaces that are crisp near surface boundaries.

wise relative rotation between nearby pixels. As can be seen from Tab. 3, encoding the ray direction helps improve the accuracy, especially for out-of-distribution camera intrinsics (Sintel and Virtual KITTI). This is in line with our observations in Tab. 2.

On the other hand, the improvement coming from the rotation estimation is not big. As the accuracy metrics are dominated by the pixels belonging to large planar surfaces, they do not convey the improvements near surface boundaries. The metrics also do not penalize the inconsistencies within piece-wise smooth surfaces. Qualitative comparison in Fig. 7 clearly shows that the proposed refinement via rotation estimation improves the piece-wise consistency and the sharpness of the prediction near surface boundaries.

6. Conclusion

In this paper, we discussed the inductive biases needed for surface normal estimation and introduced how per-pixel ray direction and the relative rotational relationship between neighboring pixels can be encoded in the output. Per-pixel ray direction allows camera intrinsics-aware inference and thus improves the generalization ability, especially when tested on images taken with out-of-distribution cameras. Explicit modeling of inter-pixel constraints — implemented in the form of rotation estimation — leads to piece-wise smooth predictions that are crisp near object boundaries.

Compared to a recent transformer-based state-of-the-art method, our method shows stronger generalization capability and a significantly higher level of detail in the prediction, despite being trained on an orders of magnitude smaller dataset. Thanks to its fully convolutional architecture, our model can be applied to images of arbitrary resolution and aspect ratio, without the need for image resizing or position encoding inter/extrapolation. We believe that the domain- and camera-agnostic generalization capability of our method makes it a strong front-end perception that can benefit many downstream 3D computer vision tasks.

7. Limitation and future work

Surface normal estimation is an inherently ambiguous task when the camera intrinsics are not known. This was why we proposed to encode the camera intrinsics in the form of dense per-pixel ray direction. While this helped us push the limits of single-image surface normal estimation, it also means that the model requires prior knowledge about the camera.

Note, however, that most RGB-D datasets already provide pre-calibrated camera parameters, and that monocular cameras can be calibrated easily using patterns with known relative coordinates. For in-the-wild images, we demonstrated in Sec. 5.2 that the intrinsics can be approximated using the image metadata. If no information is available, we can attempt to *estimate* the camera intrinsics from a single image. For instance, vanishing points with known relative angles can be used to recover the camera parameters [7]. As our model is designed to learn the relative angle between surfaces, it can in turn be used for camera calibration. This will be explored in our future work.

8. Acknowledgement

Research presented in this paper was supported by Dyson Technology Ltd. The authors would like to thank Shikun Liu, Eric Dexheimer, Callum Rhodes, Aalok Patwardhan, Riku Murai, Hidenobu Matsuki, and members of the Dyson Robotics Lab for insightful feedback and discussions.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 3

[2] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *ICCV*, pages 13137–13146, 2021. 1, 2, 3, 7

[3] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Irondepth: Iterative refinement of single-view depth using surface normal and its uncertainty. *arXiv preprint arXiv:2210.03676*, 2022. 1, 5

[4] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, pages 5965–5974, 2016. 2

[5] Oliver Boyne, Gwangbin Bae, James Charles, and Roberto Cipolla. Found: Foot optimization with uncertain normals for surface deformation using synthetic data. *arXiv preprint arXiv:2310.18279*, 2023. 1

[6] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 611–625, 2012. 6, 7, 8

[7] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International journal of computer vision*, 4(2):127–139, 1990. 8

[8] Weifeng Chen, Shengyi Qian, David Fan, Noriyuki Kojima, Max Hamilton, and Jia Deng. Oasis: A large-scale dataset for single image 3d in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 679–688, 2020. 6, 7, 3

[9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014. 5

[10] James Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *NeurIPS*, pages 845–851, 2000. 2

[11] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6, 7, 8

[12] Tien Do, Khiem Vuong, Stergios I Roumeliotis, and Hyun Soo Park. Surface normal estimation of tilted images via spatial rectifier. In *Proceedings of the European Conference on Computer Vision (ECCV), Part IV*, pages 265–280, 2020. 2

[13] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021. 6

[14] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, pages 10786–10796, 2021. 1, 2, 3, 6, 7

[15] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015. 2

[16] 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. 2

[17] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Camconvs: Camera-aware multi-scale convolutions for single-view depth. In *CVPR*, pages 11826–11835, 2019. 4

[18] David F Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3d primitives for single image understanding. In *ICCV*, pages 3392–3399, 2013. 2

[19] David Ford Fouhey, Abhinav Gupta, and Martial Hebert. Unfolding an indoor origami world. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 687–702, 2014. 2

[20] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, pages 4340–4349, 2016. 6, 7, 8

[21] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 2

[22] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, pages 577–584. 2005. 2

[23] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *IJCV*, 75:151–172, 2007. 2

[24] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302, 1994. 2

[25] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make anima-

tion from a single image. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 225–232, 1997. 2

[26] Yuan-Ting Hu, Jiahong Wang, Raymond A Yeh, and Alexander G Schwing. Sail-vos 3d: A synthetic dataset and baselines for object detection and 3d mesh reconstruction from video data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1418–1428, 2021. 6

[27] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[28] Katsushi Ikeuchi and Berthold KP Horn. Numerical shape from shading and occluding boundaries. *Artificial intelligence*, 17(1-3):141–184, 1981. 2

[29] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18963–18974, 2022. 1, 2, 3, 6, 7

[30] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Korner. Evaluation of cnn-based single-image depth estimation methods. 2018. 6, 7, 8

[31] Jana Košecká and Wei Zhang. Extraction, matching, and pose recovery based on dominant rectangular structures. *Computer Vision and Image Understanding*, 100(3):274–293, 2005. 2

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1106–1114, 2012. 2

[33] Florian Langer, Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Sparc: Sparse render-and-compare for cad model alignment in a single rgb image. *arXiv preprint arXiv:2210.01044*, 2022. 1

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[35] David C Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *2009 IEEE conference on computer vision and pattern recognition*, pages 2136–2143. IEEE, 2009. 2

[36] Shikun Liu, Linxi Fan, Edward Johns, Zhiding Yu, Chaowei Xiao, and Anima Anandkumar. Prismer: A vision-language model with an ensemble of experts. *arXiv preprint arXiv:2303.02506*, 2023. 1

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[38] David Marr. Analysis of occluding contour. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 197(1129):441–475, 1977. 2

[39] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics (ToG)*, 38(6):1–15, 2019. 6

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6, 1

[41] Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *CVPR*, pages 283–291, 2018. 1

[42] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021. 2, 7

[43] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 3

[44] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV) 2021*, 2021. 6

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), Part III*, pages 234–241, 2015. 2

[46] Shreeyak Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642. IEEE, 2020. 6

[47] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision (ECCV), Part V*, pages 746–760, 2012. 6, 7, 8

[48] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2018. 6

[49] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 6

[50] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 5, 1

[51] Rui Wang, David Geraghty, Kevin Matzen, Richard Szeliski, and Jan-Michael Frahm. Vplnet: Deep single view normal estimation with vanishing points and lines. In *CVPR*, pages 689–698, 2020. 2

[52] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020. 6

[53] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *CVPR*, pages 539–547, 2015. 2

[54] Yuliang Xiu, Jinlong Yang, Dimitrios Tzionas, and Michael J Black. Icon: Implicit clothed humans obtained from normals. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13286–13296. IEEE, 2022. 1

[55] Yuliang Xiu, Jinlong Yang, Xu Cao, Dimitrios Tzionas, and Michael J Black. Econ: Explicit clothed humans optimized via normal integration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 512–523, 2023. 1

[56] Guanglei Yang, Hao Tang, Mingli Ding, Nicu Sebe, and Elisa Ricci. Transformer-based attention networks for continuous pixel-wise prediction. In *ICCV*, pages 16269–16279, 2021. 1, 3

[57] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[58] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 6

[59] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020. 2, 7

[60] Guangyao Zhai, Dianye Huang, Shun-Cheng Wu, HyunJun Jung, Yan Di, Fabian Manhardt, Federico Tombari, Nassir Navab, and Benjamin Busam. Monograspnet: 6-dof grasping with a single rgb image. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1708–1714. IEEE, 2023. 1

[61] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 340–349, 2018. 2

[62] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1

[63] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023. 1

Rethinking Inductive Biases for Surface Normal Estimation

Supplementary Material

9. Network architecture

Tab. 4 shows the architecture of the CNN used to extract the initial surface normals, initial hidden state, and context feature. For the ConvGRU cell and convex upsampling layer, we use the architecture of [3] and [50], respectively.

10. Data preprocessing

During training, the input image goes through the following set of data augmentation (p: the probability of applying each augmentation).

- **Downsample-and-upsample** ($p = 0.1$). Bilinearly downsample the image $(H \times W)$ into $(rH \times rW)$, where $r \sim \mathcal{U}(0.2, 1.0)$. Then upsample it back to $(H \times W)$.
- **JPEG compression** ($p = 0.1$). Apply JPEG compression with quality $q \sim \mathcal{U}(10, 90)$.
- **Gaussian blur** ($p = 0.1$). Add Gaussian blur with kernel size (11×11) and $\sigma \sim \mathcal{U}(0.1, 5.0)$.
- **Motion blur** ($p = 0.1$). Simulate motion blur by convolving the image with a 2D kernel whose value is 1.0 along a line that passes through the center and is 0.0 elsewhere. The kernel is then normalized such that its sum equals 1.0. The kernel size is drawn randomly from $[1, 3, 5, 7, 9, 11]$.
- **Gaussian noise** ($p = 0.1$). Add Gaussian noise $x \sim \mathcal{N}(0, \sigma)$ where $\sigma \sim \mathcal{U}(0.01, 0.05)$. Note that the image is pre-normalized to $[0.0, 1.0]$.
- **Color** ($p = 0.1$). Use ColorJitter in PyTorch [40] with (brightness=0.5, contrast=0.5, saturation=0.5, hue=0.2).
- **Grayscale** ($p = 0.01$). Change the image into grayscale.

We also randomize the aspect ratio of the input image. Suppose that the input has a resolution of $H \times W$. We first randomize the target aspect ratio $H^{\text{target}} \times W^{\text{target}}$, while making sure that the total number of pixels is roughly 300K (to maintain GPU memory usage). We then resize the input into $rH \times rW$, such that $rH \sim \mathcal{U}(\min(H, H^{\text{target}}), \max(H, H^{\text{target}}))$. The resized input is then cropped based on the target resolution.

11. Additional figures and video

We provide an additional qualitative comparison to Omnidata v2 [14] in Fig. 8.

Input	Layer	Output	Output Dimension
image	-	-	$H \times W \times 3$
Encoder			
image	EfficientNet B5	F_8 F_{16} F_{32}	$H/8 \times W/8 \times 64$ $H/16 \times W/16 \times 176$ $H/32 \times W/32 \times 2048$
Decoder			
F_{32} + $\mathbf{r}_{32}$	Conv2D(ks=1, C_{out}=2048, padding=0)	x_0	$H/32 \times W/32 \times 2048$
$\text{up}(x_0) + F_{16} + \mathbf{r}_{16}$	$\begin{pmatrix}\text{Conv2D(ks=3, } C_{\text{out}}\text{=1024, padding=1),}\\ \text{GroupNorm}(n_{\text{groups}} = 8),\\ \text{LeakyReLU()}\end{pmatrix} \times 2$	x_1	$H/16 \times W/16 \times 1024$
$\text{up}(x_1) + F_8 + \mathbf{r}_8$	$\begin{pmatrix}\text{Conv2D(ks=3, } C_{\text{out}}\text{=512, padding=1),}\\ \text{GroupNorm}(n_{\text{groups}} = 8),\\ \text{LeakyReLU()}\end{pmatrix} \times 2$	x_2	$H/8 \times W/8 \times 512$
Prediction Heads			
$x_2 + \mathbf{r}_8$	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=3, padding=0), Normalize(), viewReLU()	$\mathbf{n}^{t=0}$	$H/8 \times W/8 \times 3$
$x_2 + \mathbf{r}_8$	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=64, padding=0)	$\mathbf{f}$	$H/8 \times W/8 \times 64$
$x_2 + \mathbf{r}_8$	Conv2D(ks=3, C_{out}=128, padding=1), ReLU(), Conv2D(ks=1, C_{out}=128, padding=0), ReLU(), Conv2D(ks=1, C_{out}=64, padding=0)	$\mathbf{h}^{t=0}$	$H/8 \times W/8 \times 64$

Table 4. **Network architecture.** In each 2D convolutional layer, "ks" and C_{out} are the kernel size and the number of output channels, respectively. F_N represents the feature-map of resolution $H/N \times W/N$, and $\mathbf{r}_N$ is a dense map of per-pixel ray direction in the same resolution. $X + Y$ means that the two tensors are concatenated, and $\text{up}(\cdot)$ is bilinear upsampling by a factor of 2.

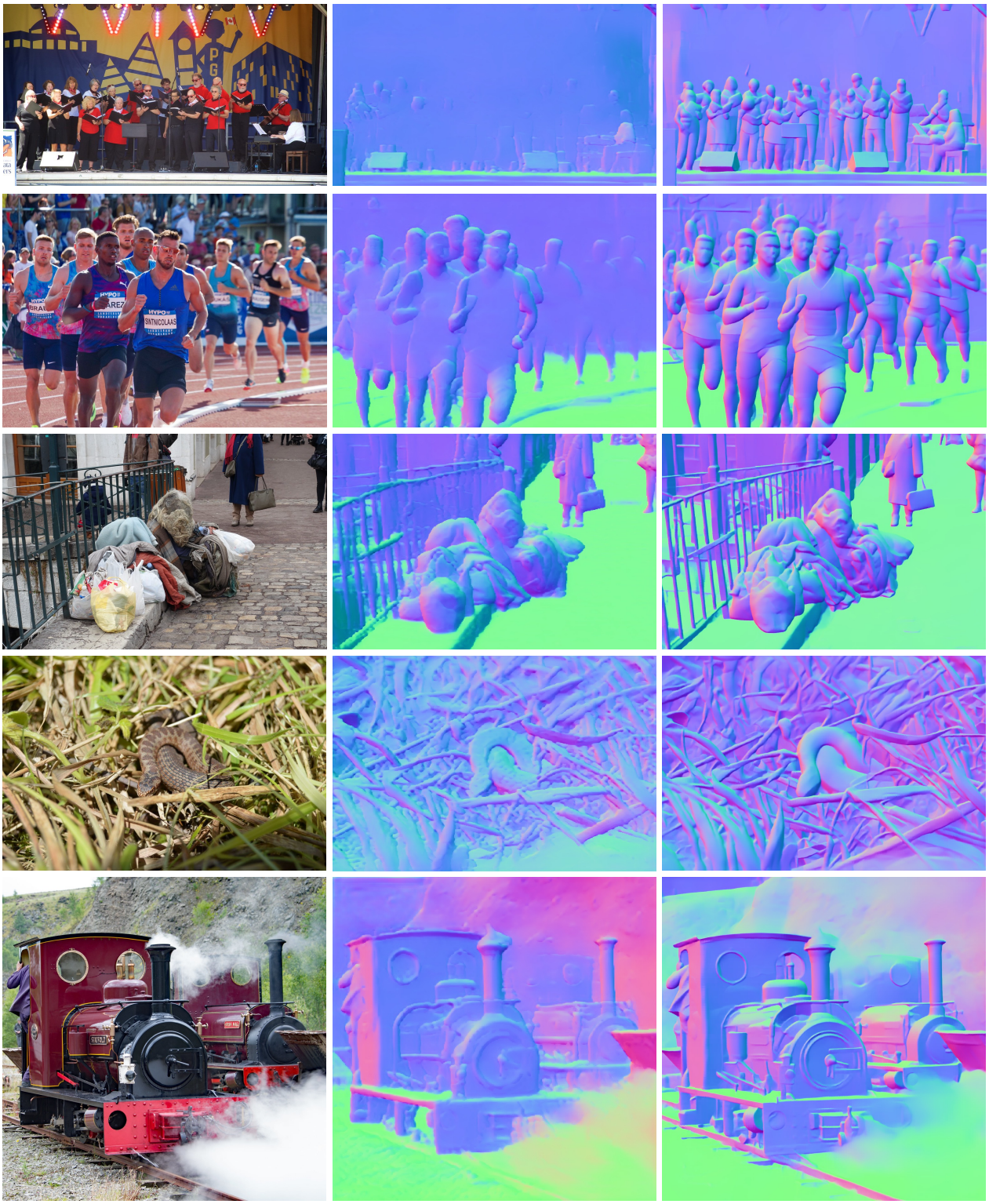

Figure 8. **Additional comparison to Omnidata v2 [14] on in-the-wild images from the OASIS dataset [8].**