

UprightNet: Geometry-Aware Camera Orientation Estimation from Single Images

Wenqi Xian^{*,1} Zhengqi Li^{*,1} Matthew Fisher² Jonathan Eisenmann² Eli Shechtman² Noah Snavely¹

¹ Cornell Tech, Cornell University ² Adobe Research

Abstract

We introduce UprightNet, a learning-based approach for estimating 2DoF camera orientation from a single RGB image of an indoor scene. Unlike recent methods that leverage deep learning to perform black-box regression from image to orientation parameters, we propose an end-to-end framework that incorporates explicit geometric reasoning. In particular, we design a network that predicts two representations of scene geometry, in both the local camera and global reference coordinate systems, and solves for the camera orientation as the rotation that best aligns these two predictions via a differentiable least squares module. This network can be trained end-to-end, and can be supervised with both ground truth camera poses and intermediate representations of surface geometry. We evaluate UprightNet on the single-image camera orientation task on synthetic and real datasets, and show significant improvements over prior state-of-the-art approaches.

1. Introduction

We consider the problem of estimating camera orientation from a single RGB photograph. This is an important problem with applications in robotics, image editing, and augmented reality. Classical approaches often rely on 2D projective geometric cues such as vanishing points [26]. However, more recent methods have sought to leverage the power of deep learning to directly regress from an image to extrinsic calibration parameters, by training on images with known ground truth orientation information [48, 19]. But these methods typically do not explicitly leverage the knowledge of projective geometry, treating the problem as a black-box regression or classification.

In this work, we introduce UprightNet, a novel deep network model for extrinsic camera calibration that incorporates explicit geometric principles. We hypothesize that injecting geometry will help achieve better performance and better

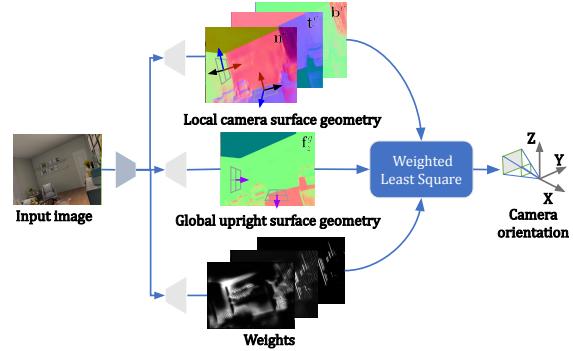


Figure 1: **UprightNet overview.** UprightNet takes a single RGB image and predicts surface geometry in both *local camera* and *global upright* coordinate systems. The camera orientation is then computed as the *alignment* between these two predictions, solved for by a differentiable least squares module, and weighted using predicted weight maps.

generalization to a broader class of images, because geometry affords generally applicable principles, and because geometric representations provide a structured intermediary in the otherwise highly non-linear relationship between raw pixels and camera orientation.

In particular, we define and use *surface frames* as an intermediate geometric representation. The orthogonal basis of the surface frames include the surface normal and two vectors that span the tangent plane of the surface. Surface frames allow us to capture useful geometric features—for instance, predicted surface normals on the ground will point directly in the up direction, and horizontal lines in the scene will point perpendicular to the up direction. However, it is not enough to know normals and other salient vectors in camera coordinates, we also need to know *which normals are on the ground*, etc. Therefore, our insight is to predict surface geometry not only in local camera coordinates, but also in *global upright* coordinates, as shown in Figure 1.

Such a global prediction is consistent across different camera views and is highly related to the semantic task of

* indicates equal contribution

predicting which pixels are horizontal surfaces (floors and ceilings), and which are vertical (walls). The camera orientation can then be estimated as the rotation that best aligns these two representations of surface geometry. This overall approach is illustrated in Figure 1. This alignment problem can be solved as a constrained least squares problem. We show in this paper that such an approach is end-to-end differentiable, allowing us to train the entire network using both supervisions on the intermediate representation of surface geometry, as well as on the final estimated orientation.

We evaluate UprightNet on both synthetic and real-world RGBD datasets, and compare it against classical geometry-based and learning-based methods. Our method shows significant improvements over prior methods, suggesting that the geometry guidance we provide to the network is important for learning better camera orientation.

2. Related Work

Single image camera calibration is a longstanding problem in computer vision. Classical geometry-based methods highly rely on low-level image cues. When only a single image is available, parallel and mutually orthogonal line segments detected in the images can be used to estimate vanishing points and the horizon lines [22, 36, 11, 26, 35, 6, 51, 5, 27, 47, 50, 3]. Other techniques based on the shape of objects such as coplanar circles [9] and repeated patterns [40] have also been proposed. When an RGB-D camera is available, one can solve for the upright orientation by assuming an ideal Manhattan world [15, 42, 43, 24]. When the scene in question has been mapped in 3D, one can also solve for the camera pose by re-localizing the cameras with respect to the 3D maps [29, 7, 39, 8, 23].

On the other hand, using machine learning methods to estimate camera orientation from a single image has gained attention. Earlier work proposes to detect and segment skylines of the images in order to estimate horizon lines [13, 2]. More recently, CNN-based techniques have been developed for horizon estimation from a single image [52, 19, 48]. Most of these methods formulate the problem as either regression or classification and impose a strong prior on the location of features correlated with the visible horizon and of corresponding camera parameters.

Single-image surface normal prediction powered by deep networks [46, 12] can provide a supervision signal for many 3D vision tasks such as planar reconstruction [33], depth completion [53] and 2D-3D alignment [4]. Recently, surface normal was used for single-image camera estimation by directly estimating a ground plane from the depth and normal estimates of segmented ground regions [34]. Unfortunately, such method assumes the ground plane is always visible in the images, and only applies to vehicle-control use cases. In addition, there are recent work making use of local surface frame representation for a variety of 3D scene understanding

tasks [20, 21]. However, our method extends beyond these ideas by estimating both local and *global aligned* surface geometry from single images and use such correspondences to estimate camera orientation. Suwajanakorn *et al.* [44] shows that the supervision on relative pose could automatically discover consistent keypoints of 3D objects, and our method is inspired by this work on end-to-end learning of intermediate representation via pose supervision.

3. Approach

Man-made indoor scenes typically consist of prominently structured surfaces such as walls, floors, and ceilings, as well as lines and junctions arising from intersections between these structures. Prominent lines also arise from other scene features, such as oriented textures. In indoor images, such geometric cues provides rich information about camera orientation. We propose to exploit such geometric features by explicitly predicting surface geometry as an intermediate step in estimating camera orientation.

To understand the benefits of such an approach, imagine that we take an image and predict per-pixel surface normals, in camera coordinates. How do these relate to camera orientation? Surface normals on the ground and other horizontal surfaces point in the same direction as the cameras up vector—exactly the vector we wish to estimate. Similarly, surface normals on walls and other vertical surfaces are perpendicular to the up vector. Hence, finding the camera orientation can be posed as finding the vector that is most parallel to ground normals, and at the same time most perpendicular to wall normals.

However, such an approach assumes that we know which pixels are ground and which are walls. Thus, we propose to also predict normals in *global scene coordinates*. This approach is in contrast to most work that predicts surface normals, which usually predict them only in the camera reference frame (e.g., if the camera is rolled 45 degrees about its axis, the predicted normals should be rotated accordingly). Given surface geometry predicted in *both* camera and scene coordinates, the camera orientation can be found as the rotation that best *aligns* these two frames.

This approach is suitable for learning. If the alignment procedure is differentiable, then we can train a method end-to-end to predict orientation by comparing to ground-truth orientations. A key advantage is that we can also apply supervision to the intermediate geometric predictions if we have ground truth.

What kind of surface geometry should we estimate? Normals are useful as described above, but do not capture in-plane features such as junctions and other lines. Hence, we propose to estimate a full orthonormal coordinate frame at each pixel, comprised of a normal and two tangent vectors. We predict these frames as a dual surface geometry representation in two coordinate systems:

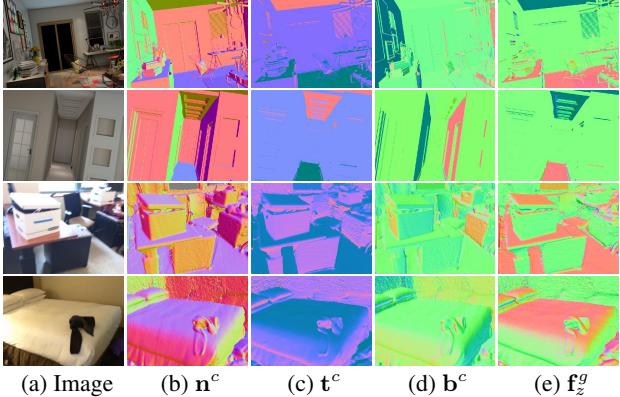


Figure 2: **Visualization of surface geometry.** From left to right: (a) image, (b-d) local camera surface frames \mathbf{F}^c , (e) the third row of global upright surface frames \mathbf{F}^g .

- \mathbf{F}^c : the surface geometry in *local camera* coordinates.
- \mathbf{F}^g : the surface geometry in *global upright* coordinates.

Surface frames. To represent surface geometry, we define a surface frame $\mathbf{F}(i)$ at every pixel location i as a 3×3 matrix formed by three mutually orthogonal unit vectors $\mathbf{F}(i) = [\mathbf{n}(i) \, \mathbf{t}(i) \, \mathbf{b}(i)]$ in either local camera coordinates or global upright coordinates:

- $\mathbf{n}^c, \mathbf{n}^g$: surface normal in camera and upright coordinates, respectively.
- $\mathbf{t}^c, \mathbf{b}^c, \mathbf{t}^g, \mathbf{b}^g$: mutually orthogonal unit vectors that span the tangent plane of the corresponding surface normal, in camera and upright coordinates respectively. (\mathbf{t} stands for tangent, and \mathbf{b} for bitangent.)

As usual, we define the camera coordinate system as a view-dependent local coordinate system, and the global upright coordinate system as the one whose camera up vector aligns with global scene up vector.

Which tangent vectors should we choose for \mathbf{t} and \mathbf{b} ? For curved surfaces, these tangent vectors are often defined in terms of local curvature. However, for man-made indoor scenes, many surfaces are planar and hence lack curvature. Instead, we define these vectors to align with the upright orientation of the scene. In particular, we define the tangent vector \mathbf{t} as a unit vector derived from the cross product between the surface normal \mathbf{n} and the camera y -axis (pointing rightward in our case). The bitangent vector \mathbf{b} is then $\mathbf{b} = \mathbf{n} \times \mathbf{t}$. This definition of tangent vectors has a degeneracy when the surface normal is parallel to the camera y -axis, in which case we instead compute \mathbf{t} to align with the up vector. However, an advantage of this choice of tangents is that this degeneracy is rare in practice. We find this choice leads to the best performance in our experiments. However, other surface frame representations could also be used [20, 21].

Camera orientation. Let \mathbf{R} be the 3×3 rotation matrix transforming local camera coordinates to global upright coordinates. \mathbf{R} maps an upright surface frame $\mathbf{F}^g(i)$ to its corresponding camera surface frame $\mathbf{F}^c(i)$ as follows:

$$\mathbf{F}^g(i) = \mathbf{R} \mathbf{F}^c(i) \quad (1)$$

Note that there is no natural reference for determining the camera heading (yaw angle) from a single image, and moreover we are most interested in determining camera roll and pitch because they are useful for graphics applications. Therefore, our problem is equivalent to finding the scene up vector in the camera coordinate system, which we denote \mathbf{u} , and which happens to be the same as the third row of \mathbf{R} . The scene up vector \mathbf{u} encodes both roll and pitch, but not yaw. It also relates the two surface frames as follows:

$$\mathbf{f}_z^g(i) = \mathbf{u}^T \mathbf{F}^c(i) \quad (2)$$

where we define $\mathbf{f}_z^g(i) = [\mathbf{n}_z^g(i) \, \mathbf{t}_z^g(i) \, \mathbf{b}_z^g(i)] \in \mathbb{R}^3$ as the third row of $\mathbf{F}^g(i)$ and it has unit length by definition.

The last column of Figure 2 shows the vectors $\mathbf{f}_z^g(i)$. Note that $\mathbf{f}_z^g(i)$ is consistent in the same supporting surfaces across images, and hence we refer to it as a *scene layout vector*. For example, \mathbf{n}_z^g for ground, wall and ceiling pixels is always fixed, to 1, 0 and -1, respectively, across *all* images, while they can differ in camera coordinates for different images according to camera orientation. Therefore, a beneficial property of the global upright frame representation is that it is similar in spirit to performing a semantic segmentation of the ground, ceiling, and other supporting structures.

To estimate 3DoF camera orientation, we could predict both camera and upright surface frames for an image, then estimate a rotation matrix that best aligns these frames. However, since we only estimate 2DoF camera orientation, it is sufficient to predict \mathbf{F}^c and \mathbf{f}_z^g .

Figure 1 shows an overview of our approach. Given a single RGB image, our network predicts per-pixel local camera surface frames \mathbf{F}^c and scene layout vectors \mathbf{f}_z^g . Using corresponding local/upright frames, we can formulate computing the best up vector as a constrained least squares problem. We show how this problem can be solved in a differentiable manner (Sec. 3.1), allowing us to train a network end-to-end by supervising it with ground truth camera orientations.

Predicting weights. A key challenge in our problem formulation is the varying uncertainty of surface geometry predictions in different image regions. We solve orientation estimation via rigid alignment as a least squares problem, which is sensitive to outliers in the predicted surface frames.

To address this problem, at each pixel i , we propose to additionally predict separate weights $w_{\mathbf{n}}(i), w_{\mathbf{t}}(i), w_{\mathbf{b}}(i)$ for each of the \mathbf{n}, \mathbf{t} and \mathbf{b} maps, and integrate these weights into the least squares solver. We have no ground truth weights

available for supervision, but because we can train our system end-to-end, the network can learn by itself to focus on only the most reliable predicted regions. Hence during training, our model jointly optimizes for surface frames, weights maps, and camera orientation.

3.1. Up vector from surface frame correspondences

Differentiable constrained least squares. Given local surface frames \mathbf{F}^c and the corresponding \mathbf{f}_z^g , our goal is to find the up vector \mathbf{u} that best aligns them. Given Eq. 2, we can write the following constrained minimization problem:

$$\begin{aligned} \min_{\mathbf{u}} \sum_{i=1}^N \|\mathbf{u}^T \mathbf{F}^c(i) - \mathbf{f}_z^g(i)\|_2^2 \\ \text{subject to } \|\mathbf{u}\|_2 = 1 \end{aligned} \quad (3)$$

where N is the number of pixels. Eq. 3 can be rewritten in matrix form as:

$$\min_{\mathbf{u}} \|\mathbf{A}\mathbf{u} - \mathbf{b}\|_2^2, \text{ subject to } \|\mathbf{u}\|_2 = 1 \quad (4)$$

where the matrix $\mathbf{A} \in \mathbb{R}^{3N \times 3}$ can be formed by vertically stacking matrices $\mathbf{F}^c(i)$ for each pixel i , and similarly vector $\mathbf{b} \in \mathbb{R}^{3N}$ can be formed by stacking the vectors $\mathbf{f}_z^g(i)$.

If there were no unit-norm constraint, this problem would be a standard least squares problem. Similarly, if $\mathbf{b} = \mathbf{0}$, the problem becomes a homogeneous least squares problem that can be solved in closed form using SVD [17]. In our case, \mathbf{b} is not necessarily a zero vector, preventing us from using such standard approaches. However, we show that Eq. 4 can be solved analytically, allowing us to use it to compute a loss in an end-to-end training pipeline.

In particular, we can write the Lagrangian of Eq. 4 as

$$L = (\mathbf{A}\mathbf{u} - \mathbf{b})^T(\mathbf{A}\mathbf{u} - \mathbf{b}) - \lambda(\mathbf{u}^T \mathbf{u} - 1) \quad (5)$$

where λ is a Lagrange multiplier. The KarushKuhnTucker condition of Eq. 5 leads to the following equations:

$$(\mathbf{A}^T \mathbf{A} - \lambda \mathbf{I})\mathbf{u} = \mathbf{A}^T \mathbf{b}, \quad \mathbf{u}^T \mathbf{u} = 1 \quad (6)$$

To solve for λ and \mathbf{u} from Eq. 6 analytically, we use the techniques proposed in [14]. Specifically, we have the following theorem [14]:

Theorem 1 *Eq. 6 can be reduced to a quadratic eigenvalue problem (QEP):*

$$\mathbf{I}\lambda^2 - 2\mathbf{H}\lambda + \mathbf{H}^2 - \mathbf{g}\mathbf{g}^T = 0 \quad (7)$$

where $\mathbf{H} = \mathbf{A}^T \mathbf{A}$, $\mathbf{g} = \mathbf{A}^T \mathbf{b}$, and Eq. 7 has a solution for λ . Further, the solution λ and $\mathbf{u} = (\mathbf{H} - \lambda \mathbf{I})^{-1} \mathbf{g}$ satisfies $(\mathbf{H} - \lambda \mathbf{I})\mathbf{u} = \mathbf{g}$ and $\mathbf{u}^T \mathbf{u} = 1$.

We refer readers to the supplementary material and to [14] for the proof. Fortunately, to solve this QEP, we can reduce it to an ordinary eigenvalue problem [14]:

$$\begin{bmatrix} \mathbf{H} & -\mathbf{I} \\ -\mathbf{g}\mathbf{g}^T & \mathbf{H} \end{bmatrix} \begin{bmatrix} \gamma \\ \mu \end{bmatrix} = \lambda \begin{bmatrix} \gamma \\ \mu \end{bmatrix} \quad (8)$$

where $\gamma = (\mathbf{H} - \lambda \mathbf{I})^{-2} \mathbf{g}$ and $\mu = (\mathbf{H} - \lambda \mathbf{I})\gamma$. Since the block matrix on the left hand side of Eq. 8 is not necessarily symmetric, the optimal λ corresponds to its minimum real eigenvalue. The derivative of this eigenvalue can be found in closed form [45], and so the solver is fully differentiable.

Weighted least squares. To improve the robustness of the least squares solver, we weight each correspondence in Eq. 3:

$$\begin{aligned} \min_{\mathbf{u}} \sum_{i=1}^N \left\| \mathbf{W}(i) (\mathbf{u}^T \mathbf{F}^c(i) - \mathbf{f}_z^g(i))^T \right\|_2^2 \\ \text{subject to } \|\mathbf{u}\|_2 = 1 \end{aligned} \quad (9)$$

and corresponding Lagrangian can be similarly modified as

$$L' = (\mathbf{A}\mathbf{u} - \mathbf{b})^T \mathbf{W}^T \mathbf{W}(\mathbf{A}\mathbf{u} - \mathbf{b}) - \lambda(\mathbf{u}^T \mathbf{u} - 1) \quad (10)$$

where $\mathbf{W} \in \mathbb{R}^{3N \times 3N}$ is a diagonal matrix, and each 3×3 block, denoted as $\mathbf{W}(i)$, is $\text{diag}([w_n(i) \ w_t(i) \ w_b(i)])$. Hence, we can use the technique described above to solve for λ and \mathbf{u} . In our experiments, we show that the predicted weights not only help to reduce the overall estimation error in the presence of noisy predictions, but also focus on supporting structures, as shown in Figure 4 and Figure 5.

3.2. Loss functions

UprightNet jointly optimizes for surface frames, weights, and camera orientation in an end-to-end fashion. Our overall loss function is the weighted sum of terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_o + \alpha_F \mathcal{L}_F + \alpha_{\nabla} \mathcal{L}_{\nabla} \quad (11)$$

In contrast to prior approaches that directly perform regression or classification on the ground-truth camera orientation, our method explicitly makes use of geometric reasoning over the entire scene, and we can train a network end-to-end with two primary objectives:

- A **camera orientation loss** that measures the error between recovered up vector and the ground-truth.
- A **surface geometry loss** that measures errors between predicted surface frames and ground-truth surface frames in both local camera and global upright coordinate systems.

Camera orientation loss \mathcal{L}_o . The camera orientation loss is applied to the up vector estimated by the surface frame correspondences and weights using our proposed constrained

weighted least squares solver. Specifically, the loss is defined as the angular distance between the estimated up vector $\hat{\mathbf{u}}$ and the ground-truth one \mathbf{u} :

$$\mathcal{L}_o = \arccos(\hat{\mathbf{u}} \cdot \mathbf{u}) \quad (12)$$

Note that both $\hat{\mathbf{u}}$ and \mathbf{u} are unit vectors. We can backpropagate through our differentiable constrained weighted least squares solver to minimize this loss directly.

A numerical difficulty is that the gradient of $\arccos(x)$ reaches infinity when $x = 1$. To avoid exploding gradients, our loss automatically switches to $1 - \hat{\mathbf{u}}^T \mathbf{u}$ when $\hat{\mathbf{u}} \cdot \mathbf{u}$ is greater than $1 - \epsilon$. In our experiments, we set $\epsilon = 10^{-6}$ and find that this strategy leads to faster training convergence and better performance compared to alternatives.

Surface frames loss \mathcal{L}_F . We also introduce a supervised loss \mathcal{L}_F over predicted surface frames in both coordinate systems to encourage the network to learn a consistent surface geometry representation. In particular, we compute the cosine similarity between each column of $\hat{\mathbf{F}}^c$ and the corresponding column of the ground-truth \mathbf{F}^c . We also compute the cosine similarity between $\hat{\mathbf{f}}_z^g$ and the ground-truth \mathbf{f}_z^g , yielding the following loss:

$$\mathcal{L}_F = 2 - \frac{1}{3N} \sum_{i=1}^N \sum_{\mathbf{f} \in \{\mathbf{n}, \mathbf{t}, \mathbf{b}\}} \hat{\mathbf{f}}^c(i) \cdot \mathbf{f}^c(i) - \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{f}}_z^g(i) \cdot \mathbf{f}_z^g(i) \quad (13)$$

Gradient consistency loss \mathcal{L}_∇ . Finally, to encourage piecewise constant predictions on flat surfaces and sharp discontinuities, we include a gradient consistency loss across multiple scales, similar to prior work [30, 31, 32]. The gradient consistency loss \mathcal{L}_∇ measures the ℓ_1 error between the gradients of the prediction and the corresponding ground truth:

$$\begin{aligned} \mathcal{L}_\nabla = & \sum_{s=1}^S \frac{1}{3N_s} \sum_{i=1}^{N_s} \sum_{\mathbf{f} \in \{\mathbf{n}, \mathbf{t}, \mathbf{b}\}} \|\nabla \hat{\mathbf{f}}^c(i) - \nabla \mathbf{f}^c(i)\|_1 \\ & + \sum_{s=1}^S \frac{1}{N_s} \sum_{i=1}^{N_s} \|\nabla \hat{\mathbf{f}}_z^g(i) - \nabla \mathbf{f}_z^g(i)\|_1 \end{aligned} \quad (14)$$

where S is the number of scales, and N_s is the number of pixels in each scale. In our experiments, we set $S = 4$ and use nearest neighborhood downsampling to create image pyramids for both the prediction and ground-truth.

3.3. Network architecture

We adopt a U-Net-style network architecture [16, 37] for UprightNet. Our network consists of one encoder and three separate decoders for $\hat{\mathbf{F}}^c$ (9 channels), $\hat{\mathbf{f}}_z^g$ (3 channels) and weight maps (3 channels), respectively. We adopt an ImageNet [38] pretrained ResNet-50 [18] as the backbone encoder. Each decoder layer is composed of a 3×3 convolutional layer followed by bilinear upsampling, and skip

connections are also applied. We normalize each column of $\hat{\mathbf{F}}^c$ and $\hat{\mathbf{f}}_z^g$ to unit length. For the weight maps, we add a sigmoid function at the end of the weight stream and normalize predicted weight maps by dividing them by their mean.

4. Experiments

To validate the effectiveness of UprightNet, we train and test on synthetic images from the InteriorNet dataset [28] and real data from ScanNet [10], and compare with several prior single-image orientation estimation methods. Furthermore, to test generalization ability, we directly apply all methods trained on ScanNet to images drawn from the SUN360 [49] dataset without fine-tuning. For all datasets, we show both qualitative and quantitative results, as well as comparisons to other baselines.

4.1. Datasets

InteriorNet [28] is a large, photo-realistic indoor scene dataset of high-quality rendered images with associated ground-truth surface normals and camera poses. We use a pre-release subset of around 34k images. Each scene includes 3 images randomly sampled from a rendered videos. Compared to other synthetic dataset such as SUNCG [41], InteriorNet has a much larger variation in camera pitch and roll. In our experiments, we randomly split InteriorNet into training, validation, and test sets using ratios of 77%, 3%, and 20% based on different scene IDs. During training, we generate randomly cropped images with a vertical field of view (FoV) varying between 35 and 45 degrees.

ScanNet [10] is an RGB-D video dataset containing indoor scenes with 3D camera poses and dense surface reconstructions. We use around 50K image frames, sampled approximately every second, for training and evaluation. In addition, during training we use rendered surface normals produced by Zhang and Funkhouser [53] for ground truth supervision, and we use the official train/val/test split based on scene id. We also use the same technique we use with InteriorNet to randomly crop images.

SUN360. We also construct a test set of rectilinear crops from the **SUN360** panorama dataset [49] for use as a cross-dataset test set. Specifically, we extract six rectified images from each indoor panorama, with output camera parameters uniformly sampled from the ranges present in the training set of ScanNet. For all datasets and methods, we resize images to 384x288 before feeding them to the networks.

4.2. Training details

We implement UprightNet in PyTorch [1]. For all experiments, we train using Adam [25], starting from an ImageNet-pretrained encoder with initial learning rate 0.0004 and mini-batch size of 8. During training, we halve the learning rate

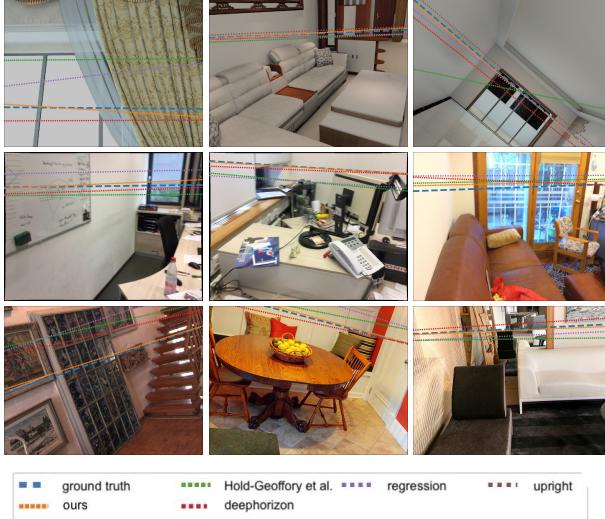


Figure 3: **Qualitative comparison of horizon line predictions.** From top row to bottom row: InteriorNet, ScanNet and SUN360. Our trained model outperforms other baselines in terms of accuracy on all three datasets.

every 5 epochs. More details on hyperparameter settings are included in the supplemental material.

4.3. Comparisons to baseline methods

We compare UprightNet with four baseline methods:

- A regression baseline: a CNN that predicts roll and pitch angles, trained with an ℓ_1 loss.
- DeepHorizon [48], a CNN classification approach to predicting horizon offsets and slopes.
- Hold-Geoffroy *et al.* [19]: a CNN classification method for predicting horizon lines and fields of view.
- Lee *et al.* [26]: a state-of-the-art classical geometry-based method.

To facilitate fair comparison, we re-implemented [48] and [19] based on their published descriptions. For all learning-based methods, we adopt the same network architecture and pretrained weights as our method, and train and evaluate them on the same datasets using the same training strategy.

To evaluate estimated camera orientations, we compute the mean and median angular error between the predicted and ground truth up vectors, as in Equation 12, as well as absolute errors in both pitch and roll angles.

Table 1 shows quantitative results on InteriorNet, and the top row of Figure 3 shows a visualization of estimated horizon lines superimposed on the input images for qualitative comparisons. Our proposed method achieve a significant improvement (30%) compared to prior CNN-based and geometry-based methods for all error metrics.

Method	angular error (°)		pitch error (°)		roll error (°)	
	avg.	med.	avg.	med.	avg.	med.
CNN Regression	2.29	1.20	1.88	1.02	0.93	0.34
Upright [26]	3.95	1.76	3.59	1.70	1.15	0.33
DeepHorizon [48]	2.12	1.13	1.65	0.85	0.93	0.40
Hold-Geoffroy <i>et al.</i> [19]	1.76	0.81	1.35	0.58	0.81	0.26
Ours	1.17	0.52	0.99	0.47	0.44	0.11

Table 1: **Quantitative comparisons on the InteriorNet test set.** Our approach has the best performance in terms of all the metrics. All errors are in degrees and lower is better.

Method	angular error (°)		pitch error (°)		roll error (°)	
	avg.	med.	avg.	med.	avg.	med.
CNN Regression	4.88	3.63	3.85	2.59	2.24	1.57
Upright [26]	14.51	5.71	4.65	2.13	12.99	4.32
DeepHorizon [48]	5.05	3.80	3.81	2.56	2.51	1.82
Hold-Geoffroy <i>et al.</i> [19]	4.55	3.33	3.53	2.33	2.15	1.50
Ours	3.74	2.94	2.88	2.04	2.12	1.48

Table 2: **Quantitative comparisons on the ScanNet test-set.** See Table 1 for descriptions.

Table 2 shows quantitative results on ScanNet, and the second row of Figure 3 shows predicted horizon lines on that dataset. ScanNet is more challenging for image calibration compared to InteriorNet due to motion blur, extreme pitch angles, and more complex scene structures. Nevertheless, our method still improves calibration accuracy by a relative 20% in angular and pitch error, and is slightly better than the best baseline methods in terms of roll angle.

Figure 4 shows visualizations of our predicted \mathbf{n}^c , \mathbf{f}_z^g , and weights for InteriorNet and ScanNet test images. For visualization, in Figure 4(d) we combine the three weights maps by adding the normalizing them, and overlay them on the original input images. Interestingly, the network learns weight maps that tend to focus on the vicinity of vertical/horizontal surface line junctions. Such lines are very strong cues for vanishing points. For instance, \mathbf{t}^c in the vicinity of vertical junctions and \mathbf{b}^c in the vicinity of horizontal junctions both represent 3D directions of their respective junction lines, and also point towards 2D vanishing points [17]. It is interesting that the network “discovers” such features automatically

4.4. Ablation analysis

We now explore how the weight maps and different configurations of surface geometry supervision affect the performance of UprightNet through validation on InteriorNet.

Impact of weights. We wanted to see if the network learns to correctly assign large weights to regions with smaller alignment error. We compute an alignment score map by using the ground-truth up vector \mathbf{u} to align each column of $\hat{\mathbf{F}}^c$ with its corresponding scalar in $\hat{\mathbf{f}}_z^c$. We define alignment scores for surface normals as $S_{\mathbf{n}} = \exp(-10|\mathbf{u}^T \hat{\mathbf{n}}^c - \hat{\mathbf{n}}_z^g|)$

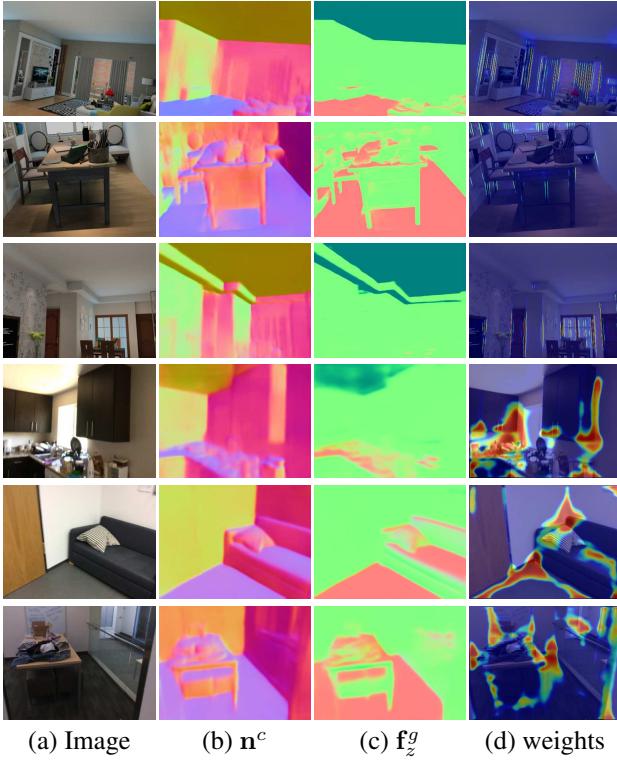


Figure 4: **Visualizations of predictions** in InteriorNet (top 3 rows) and ScanNet (bottom 3 rows). In (d), we overlay weight maps (combined weights for \mathbf{n} , \mathbf{t} and \mathbf{b}) over input images. Blue=small weights, red=large weights.

and similarly for the two tangent vectors.

Figure 5 visualizes these alignment scores, and compares them with the predicted weights. The network indeed tends to predict large weights where the alignment error is small. This suggests that, while our surface geometry predictions are not always accurate, the weights can capture which image regions are more likely to have correct predictions, thus leading to better estimation of camera orientation.

Impact of camera orientation loss. As shown in Table 3, we evaluate our models using different configurations to analyze their influence on the performance. Comparing with direct estimation of camera pose from the predicted surface frame correspondences, end-to-end optimization with camera orientation loss boosts performance significantly. We observe an additional increase in performance by incorporating predicted weights into orientation estimation. Since the supervision of surface frames in both coordinate systems is not required during training, we can also train a model using supervision only from local camera frames, or only from global upright frames. The ablations shown in Table 3 suggest that using both local camera and global upright surface geometry as supervision leads to the best performance.

Impact of surface geometry representation. We also ex-

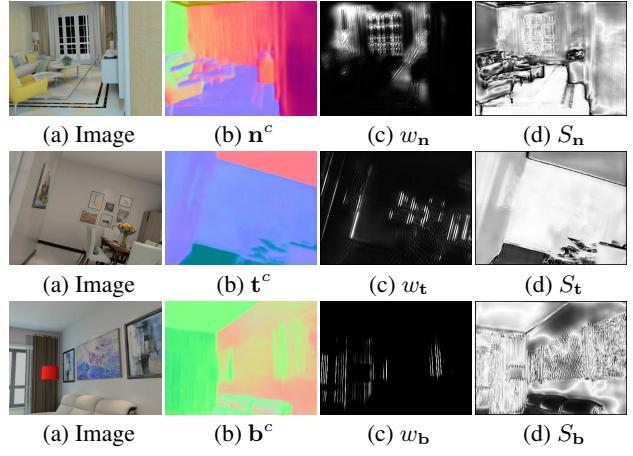


Figure 5: **Comparisons between predicted weights and alignment scores.** From left to right (a) input image, (b) predict camera surface frames, (c) predicted weights, (d) alignment scores S . In (c) and (d), white indicates large weights and high alignment score (i.e. low alignment error).

Method	angular error ($^{\circ}$)		pitch error ($^{\circ}$)		roll error ($^{\circ}$)	
	avg.	med.	avg.	med.	avg.	med.
Ours (w/o weight)	1.83	1.15	1.22	0.81	0.90	0.57
Ours (w/o \mathcal{L}_o)	2.71	1.74	1.21	2.83	1.37	0.93
Ours (w/o F^c loss)	1.70	1.11	1.31	0.99	0.76	0.57
Ours (w/o F^u loss)	1.82	1.18	1.32	0.98	0.79	0.55
Ours (full)	1.17	0.52	0.99	0.47	0.44	0.11

Table 3: **Ablation studies for different configurations on the InteriorNet test set.** Ours (full) indicate our proposed full method, which achieves the best performance among all tested configurations.

plore the influence of different surface geometry representations on camera orientation estimation. In particular, we compare our full surface geometry representation to (1) a single vector representation, i.e., just one of (\mathbf{n} , \mathbf{t} , \mathbf{b}), and (2) a combination of any two of them. As shown in Table 4, our proposed full surface frame representation achieves the best performance. This suggests that each basis vector of our surface frames captures complementary geometry cues in indoor scenes. In particular, \mathbf{n}^c of grounds/ceilings and \mathbf{t}^c of vertical lines directly represent the scene up vectors we seek, while \mathbf{b}^c of horizontal lines on supporting surfaces could correspond to major vanishing points in the scene.

4.5. Generalization to SUN360

We explore the ability of different learning-based calibration methods to generalize across datasets by taking models trained on ScanNet and testing them on crops from the SUN360 indoor panorama dataset. We summarize the results in Table 5, and visualize surface normal and weight

Method	angular error (°)		pitch error (°)		roll error (°)	
	avg.	med.	avg.	med.	avg.	med.
n	1.81	1.13	1.35	0.85	0.90	0.45
t	2.11	1.28	1.60	0.96	1.06	0.47
b	1.82	1.09	1.39	0.84	0.81	0.47
nt	1.52	0.97	1.27	0.83	0.58	0.29
nb	1.17	0.89	1.33	0.77	0.78	0.29
tb	1.56	0.66	1.29	0.62	0.66	0.20
ntb	1.17	0.52	0.99	0.47	0.44	0.11

Table 4: **Ablation studies for different surface geometry representations on the InteriorNet test set.** ntb indicates our full surface geometry representation.

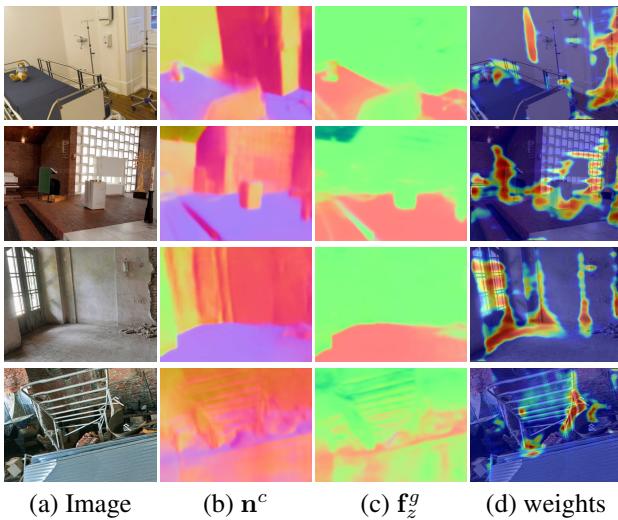


Figure 6: **Visualizations of predictions on the SUN360 testset.** The last row shows a failure case.

predictions of our model in Figure 6. While the errors are naturally higher on this unseen dataset, UprightNet still outperforms prior methods by large margins in all error metrics. This suggests that, compared with implicit features learned from direct regression or classification, using an intermediate geometric representation can help a model attain better generalization to new distributions of indoor imagery.

We also demonstrate the use of UprightNet for an application in virtual object insertion in Figure 7. We orient a 3D object with the camera’s pitch and roll estimated by UprightNet. The 2D translation of the object along the ground plane is manually chosen by the artist. Finally, a 2D render of the object is composited on top of the image to produce a final results, shown in the bottom row of Figure 7.

5. Limitations and future work

The last row of Figure 6 demonstrate a common failure mode of our approach, where the image lacks sufficient

Method	angular error (°)		pitch error (°)		roll error (°)	
	avg.	med.	avg.	med.	avg.	med.
CNN Regression	10.43	6.99	9.57	6.10	3.10	2.17
DeepHorizon [48]	9.53	6.28	8.68	5.50	2.98	1.89
Hold-Geoffroy <i>et al.</i> [19]	10.41	6.92	9.57	6.09	3.11	2.20
Ours	7.81	5.53	7.59	4.94	2.30	1.53

Table 5: **Generalization performance on the SUN360 test set.** All methods are trained/validated on ScanNet. Our method achieves the best performance on all metrics.



Figure 7: **Application.** Virtual insertion of 3D objects in images from SUN360 dataset using the camera orientation estimated by UprightNet.

supporting structure for the network to reason about geometry, resulting in inaccurate surface geometry and camera orientation predictions. In future work, other explicit 2D geometric priors, such as vanishing points, or camera intrinsics (if known) can also be integrated to our framework.

6. Conclusion

We introduced UprightNet, a new method for predicting 2DoF camera orientation from a single indoor image. Our key insight is to leverage surface geometry information from both the camera and upright coordinate systems, and pose camera orientation prediction as an alignment problem between these two frames. In particular, we showed how a network can be trained to predict camera-centric and global surface frames, and combine them with weights to estimate the camera orientation. Our evaluations demonstrated not only more accurate orientation estimation, but also better generalization to unseen datasets, compared with prior state-of-the-art methods.

Acknowledgments. We thank Geoffrey Oxholm, Qianqian Wang, and Kai Zhang for helpful discussion and comments. This work was funded in part by the National Science Foundation (grant IIS-1149393), and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

- [1] Pytorch, 2016. <http://pytorch.org>.
- [2] Touqeer Ahmad, George Bebis, Emma E Regentova, and Ara Nefian. A machine learning approach to horizon line detection using local features. In *International Symposium on Visual Computing*, pages 181–193. Springer, 2013.
- [3] Michel Antunes and Joao P Barreto. A global approach for the detection of vanishing points and mutually orthogonal vanishing directions. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1336–1343, 2013.
- [4] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5965–5974, 2016.
- [5] Jean-Charles Bazin and Marc Pollefeys. 3-line ransac for orthogonal vanishing point detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4282–4287. IEEE, 2012.
- [6] Jean-Charles Bazin, Yongduek Seo, Cédric Demonceaux, Pascal Vasseur, Katsushi Ikeuchi, Inso Kweon, and Marc Pollefeys. Globally optimal line clustering and vanishing point estimation in manhattan world. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 638–645. IEEE, 2012.
- [7] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 6684–6692, 2017.
- [8] Eric Brachmann and Carsten Rother. Learning less is more: 6d camera localization via 3d surface regression. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4654–4662, 2018.
- [9] Qian Chen, Haiyuan Wu, and Toshikazu Wada. Camera calibration with two arbitrary coplanar circles. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 521–532. Springer, 2004.
- [10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017.
- [11] Jonathan Deutscher, Michael Isard, and John MacCormick. Automatic camera calibration from a single manhattan image. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 175–188. Springer, 2002.
- [12] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 2650–2658, 2015.
- [13] Sergiy Fefilatyev, Volha Smarodzinava, Lawrence O Hall, and Dmitry B Goldgof. Horizon detection using machine learning techniques. In *International Conference on Machine Learning and Applications*, pages 17–21. IEEE, 2006.
- [14] Walter Gander, Gene H Golub, and Urs von Matt. A constrained eigenvalue problem. *Linear Algebra and its applications*, 114:815–839, 1989.
- [15] Bernard Ghanem, Ali Thabet, Juan Carlos Niebles, and Fabian Caba Heilbron. Robust manhattan frame estimation from a single rgb-d image. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 3772–3780, 2015.
- [16] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.
- [17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [19] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Jonathan Eisenmann, Matthew Fisher, Emiliano Gambaretto, Sunil Hadap, and Jean-François Lalonde. A perceptual measure for deep single image camera calibration. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2354–2363, 2018.
- [20] Jingwei Huang, Haotian Zhang, Li Yi, Thomas Funkhouser, Matthias Nießner, and Leonidas J Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4440–4449, 2019.
- [21] Jingwei Huang, Yichao Zhou, Thomas Funkhouser, and Leonidas Guibas. Framenet: Learning local canonical frames of 3d surfaces from a single rgb image. *arXiv preprint arXiv:1903.12305*, 2019.
- [22] Kyungdon Joo, Tae-Hyun Oh, Junsik Kim, and In So Kweon. Globally optimal manhattan frame estimation in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1763–1771, 2016.
- [23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 2938–2946, 2015.
- [24] Pyojin Kim, Brian Coltin, and H Jin Kim. Indoor rgb-d compass from a single line and plane. In *CVPR*, 2018.
- [25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [26] Hyunjoon Lee, Eli Shechtman, Jue Wang, and Seungyong Lee. Automatic upright adjustment of photographs with robust camera calibration. *Trans. Pattern Analysis and Machine Intelligence*, 36(5):833–844, 2014.
- [27] José Lezama, Rafael Grompone von Gioi, Gregory Randall, and Jean-Michel Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 509–515, 2014.
- [28] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tsoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *Proc. British Machine Vision Conf. (BMVC)*, 2018.
- [29] Yunpeng Li, Noah Snavely, and Daniel P. Huttenlocher. Location recognition using prioritized feature matching. In *Proc. European Conf. on Computer Vision (ECCV)*, 2010.

[30] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4521–4530, 2019.

[31] Zhengqi Li and Noah Snavely. CGIntrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 371–387, 2018.

[32] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2041–2050, 2018.

[33] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2579–2588, 2018.

[34] Yunze Man, Xinshuo Weng, and Kris Kitani. Groundnet: Segmentation-aware monocular ground plane estimation with geometric consistency. *arXiv preprint arXiv:1811.07222*, 2018.

[35] Faraz M Mirzaei and Stergios I Roumeliotis. Optimal estimation of vanishing points in a manhattan world. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 2454–2461. IEEE, 2011.

[36] Isao Miyagawa, Hiroyuki Arai, and Hideki Koike. Simple camera calibration from a single image using five points on two orthogonal 1-d objects. *IEEE Transactions on Image Processing*, 19(6):1528–1538, 2010.

[37] 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*, pages 234–241. Springer, 2015.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. of Computer Vision*, 115(3):211–252, 2015.

[39] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2011.

[40] Frederik Schaffalitzky and Andrew Zisserman. Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18(9):647–658, 2000.

[41] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2017.

[42] Julian Straub, Nishchal Bhandari, John J Leonard, and John W Fisher. Real-time manhattan world rotation estimation in 3d. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1913–1920. IEEE, 2015.

[43] Julian Straub, Oren Freifeld, Guy Rosman, John J Leonard, and John W Fisher. The manhattan frame modelmanhattan world inference in the space of surface normals. *Trans. Pattern Analysis and Machine Intelligence*, 40(1):235–249, 2018.

[44] Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Neural Information Processing Systems*, pages 2063–2074, 2018.

[45] Nico van der Aa. Perturbation theory for eigenvalue problems. 2005.

[46] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 539–547, 2015.

[47] Horst Wildenauer and Allan Hanbury. Robust camera self-calibration from monocular images of manhattan worlds. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2831–2838. IEEE, 2012.

[48] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon lines in the wild. In *Proc. British Machine Vision Conf. (BMVC)*, 2016.

[49] Jianxiong Xiao, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2695–2702, 2012.

[50] Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch. Pose estimation from line correspondences: A complete analysis and a series of solutions. *Trans. Pattern Analysis and Machine Intelligence*, 39(6):1209–1222, 2017.

[51] Yiliang Xu, Sangmin Oh, and Anthony Hoogs. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1376–1383, 2013.

[52] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattan world. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5657–5665, 2016.

[53] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 175–185, 2018.