360° Camera Alignment via Segmentation

Benjamin Davidson, Mohsan S. Alvi, and João F. Henriques

Disperse.io
{ben, mohsan, joao}@disperse.io

Abstract. Panoramic 360° images taken under unconstrained conditions present a significant challenge to current state-of-the-art recognition pipelines, since the assumption of a mostly upright camera is no longer valid. In this work, we investigate how to solve this problem by fusing purely geometric cues, such as apparent vanishing points, with learned semantic cues, such as the expectation that some visual elements (e.g. doors) have a natural upright position. We train a deep neural network to leverage these cues to segment the image-space endpoints of an imagined "vertical axis", which is orthogonal to the ground plane of a scene, thus levelling the camera. We show that our segmentation-based strategy significantly increases performance, *reducing errors by half*, compared to the current state-of-the-art on two datasets of 360° imagery. We also demonstrate the importance of 360° camera levelling by analysing its impact on downstream tasks, finding that incorrect levelling severely degrades the performance of real-world computer vision pipelines.

1 Introduction

The ability of 360° (or spherical) imaging to record an entire scene with a single capture makes them a powerful tool, both for machine perception and for rapidly documenting entire scenes. For example, 360° imaging has been used to record crime scenes where it is vital to image the entire scene for evidence [32], to easily create Virtual Reality (VR) videos with minimal cost [24], and is perhaps most widely recognized in its role in creating Google Street View [12]. Arrays of cameras that can be composed into a full 360° image or video are also important in mobile applications with critical safety requirements, such as self-driving cars [1]. With the availability of inexpensive 360° capture devices, and the growth of VR headsets, there is an increased demand for techniques to automatically analyse and process spherical images.

The recent successes of computer vision, with deep learning playing a key role in the state-of-the-art object detectors [22], segmentation [30], camera pose estimation [20] and many others, seem to indicate that the same techniques should be directly applicable to 360° images. However, there are specific difficulties associated with this modality that need to be addressed. One common problem for spherical images is a misalignment between the camera frames' ground plane and the world frames' ground plane (see fig. 2).

This misalignment makes automatically processing spherical images more challenging than it needs to be. For example, training a spherical object detector

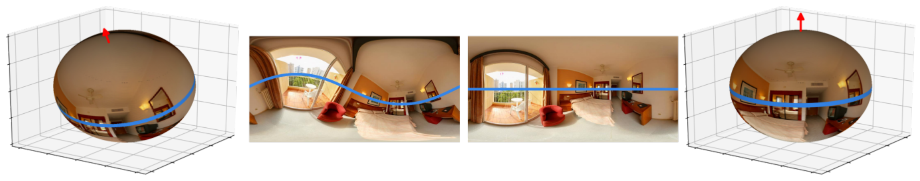

Fig. 1. Illustration of the problem of levelling spherical images. From left to right: a spherical image captured at a tilted angle relative to the sky (red arrow), with the horizon line shown in blue; its 2D representation (equirectangular image), with heavy distortions due to the rotation; the same image, undistorted by our system; the aligned spherical image in 3D.

on misaligned images would require the network to learn a representation which was invariant to rotations away from the vertical axis [15]. In contrast to this, if all images are level (upright), the representation could be sensitive to these rotations, simplifying the task to be learned [11].

The ground-plane alignment that we focus on estimates 2 degrees of freedom (DOF) (roll and pitch), and must be contrasted to general camera pose estimation, which estimates 6 DOF (translation and rotation in 3D) [20]. Ground-plane alignment can be performed *with a single image*, by using simple cues (e.g. vertical walls, ground or sky/ceiling positions). Differently, 6D camera pose estimation requires extra reference images [14, 20, 19], making it much less applicable.

Aligning spherical images to the ground is also an important pre-processing step for downstream tasks (we demonstrate this empirically in sec. 3). State-of-the-art object detectors, and segmentation networks are trained and evaluated on upright equirectangular images [37, 38, 8], and do not work under arbitrary rotations. Similarly, human visual recognition also degrades quickly with extreme rotations [33], and there are classification problems that are impossible to solve under arbitrary rotations (the canonical example being the distinction between the digits 9 and 6). Ground-plane alignment can also make pose estimation more robust, as estimating the pose of a levelled image requires two fewer DOF [28].

At a high level, our method estimates the axis orthogonal to the ground (vertical axis) by segmenting the unit sphere (where each point corresponds to a different direction) into likely candidates. We leverage a state-of-the-art segmentation network [30], by exploiting the fact that the unit sphere can be mapped to a 2D image via the equirectangular transform (sec. 2.1). The network is trained to segment the sphere into those directions likely to correspond to the vertical axis.

In addition to the novel segmentation formulation of this problem, we propose to combine the strengths of both geometrical methods and learning-based methods. Geometrical methods, such as those based on detecting and accumulating votes for locations of vanishing points (VP) [10, 21, 37], are very accurate, but brittle under noise and uncertainty. Learning-based methods are less accurate but very robust. We combine both, by incorporating a residual stream

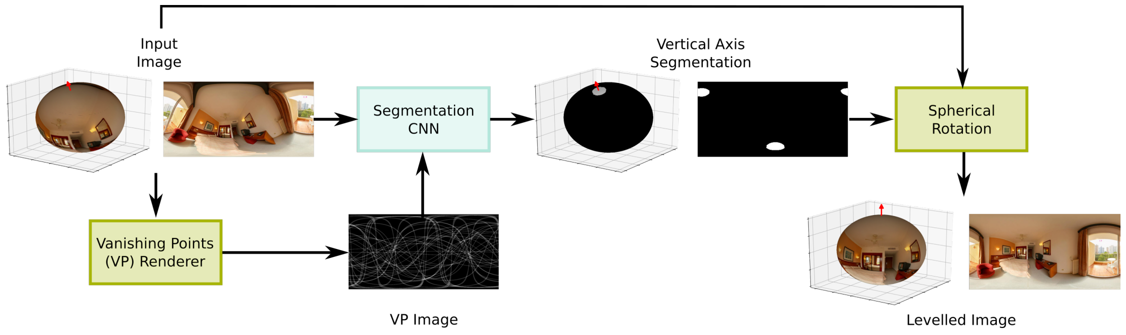

Fig. 2. Overview of the proposed method. We train a convolutional network to produce a segmentation of an equirectangular image, using vanishing point features as extra geometric information. The output segmentation encodes the endpoints of the vertical axis, which we use to orient the image upright.

that propagates information about VP likelihoods, and this way informs our segmentation network with precise geometrical information.

By leveraging the power of feature engineering with state-of-the-art segmentation techniques, our method is the most accurate to date. We compare our method with the two most recent automatic alignment methods Deep360Up [18] and Coarse2Fine [27]. We demonstrate improved performance on the Sun360 dataset [35], as well a new dataset of construction images that we collected.

1.1 Related work

Ground plane alignment is related to pose estimation [14, 20, 19], as described in sec. 1. Another related line of work is rotation invariant (or equivariant) networks [34, 15], which aims to make models more robust and predictable w.r.t. rotations, and is complementary to our method. We aim instead to predict and undo the effect of a single global rotation, with a semantically-defined reference (the ground plane).

The classical solution to ground plane alignment has been to extract the straight line segments from an image, and use these to estimate a vanishing point in the direction of the vertical axis [10, 21, 37]. These methods rely on what are known as the Manhattan or Atlanta world assumptions [4, 25], which assert that the scene that has been captured will contain some orthogonal structure, given the tendency in human construction to build at right angles. It must be remarked that this assumption does not always hold in practice. One typical way to extract this orthogonal structure is to determine the direction in which all straight lines in an image are pointing, and have each line vote on vanishing point directions [37], in a manner similar to the Hough transform [7] (c.f. sec. 2.3). The orthogonal directions of the scene can then be found by looking for the three orthogonal directions which together have the most votes. However, many scenes may not have this orthogonal structure, and we may not be able to extract many straight line segments from the image. Moreover, the maximal orthogonal set found by maximisation may not be the true orthogonal directions. Due to the many assumptions of this approach, it is very brittle in practice, despite the apparent strength of the vanishing point features it uses.

Deep learning solutions to ground plane alignment have shown to be more robust than the classical vanishing point methods. The existing methods are either a variation of a deep convolutional regression network [18, 17] or a classification network [27]. In the most recent regression network, referred to as Deep360Up, the vertical direction (pointing upwards) is output directly from a DenseNet [16], which is trained using the logarithm of the hyperbolic cosine between the estimated and ground truth vertical directions [18]. The most accurate and recent deep approach has been to use a *coarse to fine* classification network [27]. This approach, referred to as Coarse2Fine, classifies the pitch and roll of an image as belonging to a $10°$ bin (coarse), thus adjusting the image to be within $\pm 5°$, and then classifying the adjusted image to be within a $1°$ bin (fine). Another standard feature of such solutions is to generate training data from already levelled images (which we discuss in sec. 2.4). Though these methods have once again demonstrated the power of deep networks, we show in sec. 3 that the proposed segmentation approach is more accurate.

A related line of work is to propose network architectures that directly work with spherical images, for example for classification and detection [3], or for segmentation and depth prediction [31]. Our levelling method can alleviate any upright-world assumptions in these works, as well as standard networks, and is thus complementary.

2 Methods

Our approach can broadly be split into three stages: calculating the vanishing points, segmenting the image, and processing the segmentation into a single vertical direction. Before describing our method in detail, we provide some background on equirectangular images and some useful operations.

2.1 Background on equirectangular images

An equirectangular image is a planar representation of an image on the sphere, where height and width correspond respectively to latitude and longitude. The explicit transformation (denoted p) between pixel coordinates (x, y) and spherical coordinates (λ, ϕ) is straightforward:

$$p : \mathbb{R}^2 \to \mathbb{S}^2, \quad p(x, y) = \left(\frac{\pi y}{h}, 2\pi - \frac{\pi x}{w}\right) = (\lambda, \phi) \tag{1}$$

where w and h are the dimensions in pixels. Note that this is an invertible transformation and so we can move from the image to the sphere and vice-versa. Using p we will frequently refer to an equirectangular image as being on the sphere, by which we mean the projection of the image to the sphere. Furthermore we

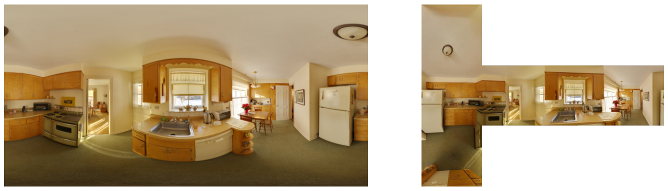

Fig. 3. An example equirectangular image, and its corresponding projection on the faces of a cube. Note the many curved lines in the equirectangular image, which become straight in the corresponding cube face.

can map spherical to Cartesian coordinates and vice versa using the spherical to cartesian transformation: $f(\lambda, \phi) = (\cos(\phi)\sin(\lambda), \sin(\phi)\sin(\lambda), \cos(\lambda))$. We can use these transformations to rotate an equirectangular image I_{src} to create another image I_{dst} of different orientation, by rotating the sphere. Starting from a point x_{dst} in the pixel space of I_{dst} we project x_{dst} to the sphere $p(x_{\text{dst}})$, and rotate the sphere with a rotation matrix $R \in SO(3)$. Note that R represents an arbitrary rotation in 3D space, with an axis of rotation not necessarily corresponding to latitude or longitude. Doing so gives the following relationship between coordinate systems:

$$y_{\text{src}} = Rf(p(x_{\text{dst}})) \tag{2}$$

After this transformation we project back to image space: $x_{\text{src}} = p^{-1}(f^{-1}(y_{\text{src}}))$. The transformation of image coordinates x_{dst} to x_{src} allows us to re-sample an image I_{src} to create I_{dst}, for example by bilinear interpolation [15]. As can be seen in eq. (2), we may rotate the image so that we have an equirectangular image of any orientation, which we will use to generate training data for our segmentation network.

Another subtle but important aspect about equirectangular images is how to extract straight line segments visible within the scene. Straight lines in the scene do not in general map to straight lines in an equirectangular image (see fig. 3). To recover straight lines from an equirectangular image, we need to convert it to one or more perspective images. We cover the full 360° view with perspective views, corresponding to 6 cube faces (see fig. 3). Each one is produced by rendering the sphere (with the mapped texture) from 6 different points-of-view, at right angles. This "cube mapping" is commonly used in computer graphics to render far-away scenes [26]. This allows using unmodified line segment detectors.

2.2 Segmentation framework

Our method is based on a convolutional neural network optimised for segmentation, with side-information about vanishing points as input to an attention module. The output of our network is a binary segmentation of the original equirectangular image, which by applying the pixel to spherical transform p may be thought of as a segmentation of the sphere into background and likely directions for the vertical axis. Specifically, we segment all points on the sphere which are within 5 degrees of the north or south pole, where the poles are taken relative to camera coordinates (see fig. 2 for an example segmentation). By embedding all useful inputs and outputs in a 2-dimensional space, we can leverage highly successful 2D segmentation networks, and allow predictions to be based on both geometric and semantic cues (i.e., vanishing points and poses of distinctive objects in images).

Network architecture. The base architecture that we use is the Gated-Shape CNN (GSCNN) [30]. The GSCNN is a fully convolutional network, designed to utilise side information about object boundaries to improve its performance on semantic segmentation tasks. It consists of a backbone feature extractor, in our case InceptionV3 [29], an ASPP (atrous spatial pyramid pooling) layer, and the shape stream. The shape stream in the original work accepts image gradients and intermediate backbone features as inputs, and outputs a single channel feature image. The output shape stream features are then combined with other backbone features in the ASPP layer to generate a dense feature map of the same resolution as the input image.

Our architecture modifies GSCNNs so that it would be more informative to call the shape stream the *vanishing point stream*, as we replace image gradients with the vanishing point image V (see sec. 2.3). The reasoning behind this is that V is a feature that is highly informative w.r.t. the vertical axis, and we would like to let the network exploit this source of information. Also, feeding V to the network in this manner allows us to use a pre-trained backbone network, which would not be possible by just concatenating V to the channels of the image. Using a GSCNN enabled us to introduce information relating to vanishing points, whilst also retaining the ability to use pre-trained backbones.

2.3 Vanishing point image

Vanishing points have proven to be a strong geometric cue for many computer vision tasks, including ground plane alignment [10]. In many scenes a horizon line is visible, or orthogonal structures such as the corners of buildings. These

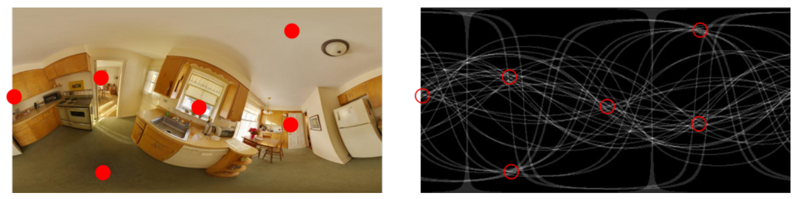

Fig. 4. An equirectangular image and the corresponding vanishing point image (sec. 2.3). The 6 regions highlighted in red are areas which have received a large number of votes. Note that each highly-voted region corresponds to one face of the approximately cuboid room (the four walls, floor and ceiling).

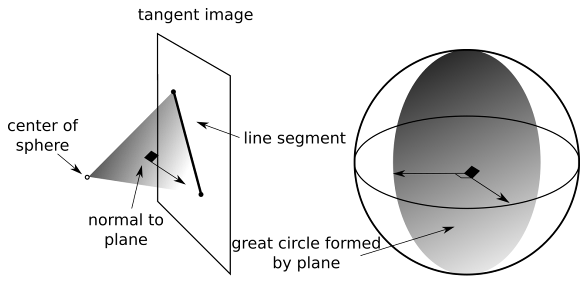

Fig. 5. Illustration of how to calculate vanishing point features from line segments in a cube face. We can see that every point on the great circle formed by intersecting the plane and the sphere will be orthogonal to the normal vector. Therefore, every point in this circle receives a vote. In practice the 2D surface of the sphere is discretised (sec. 2.1), and every bin within some threshold distance of this circle receives a vote.

structures are useful for determining the vertical axis, and can be emphasised by calculating vanishing points (see fig. 4). Moreover, these features can be computed directly from images, with no learning required. This makes them excellent features for our purpose.

To build the vanishing point image in fig. 4 we extract all of the straight lines in the scene and use each line to vote on vanishing directions. The first step of this process is to project the equirectangular image to the 6 cube faces (fig. 3, right) and extract line segments from each face. To extract the line segments we use Canny edge detection combined with a probabilistic Hough transform [2, 9]. We then convert each line segment to a plane, defined by the line endpoints, and the origin of the sphere. Let n be the normal vector to this plane. We use n to vote for vanishing point locations, by voting for all directions on the sphere which are orthogonal to n. Geometrically this means all points on the great circle defined by the intersection of the plane and sphere receive a vote. In practice we split the sphere into $h \times w$ bins by projecting each pixel in an equirectangular image I to the sphere and then voting via

$$V^n_{h_0,w_0} = \begin{cases} 1 & |n \cdot f(p(I_{h_0,w_0}))| < \lambda_{\text{vanishing}} \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

We calculate a normal vector n for every line segment and accumulate votes by summing $V = \sum_n V^n$. Finally we normalise V to be an intensity image with values in the range of $[0, 255]$. High values will correspond to probable vanishing points, which have many line segments pointing towards them, and will assist our network in finding the vertical axis (fig. 5).

2.4 Training method

To train our network, we use a weighted generalised dice loss [5] on uniformly distributed points on the sphere. This is in contrast to the original GSCNN work which utilises auxiliary and regularising losses [30], and which have no direct analogue in our setting. We chose to use the generalised dice loss as this has been shown to perform well in situations where there are large class imbalances between foreground and background classes [5]. This is a concern in our setting, since vanishing points are sparse.

We do not compute the loss directly on the 2D segmentation image, as this would over-sample the polar regions, thus disproportionately weighing vertical directions near them. Instead, we select points that are uniformly distributed around the sphere, and project these into the equirectangular segmentation, and ground truth. Finally, we interpolate the values of each projected point to construct y and $\hat{y}$.

Training data. The data fed to our network during training are equirectangular images (e.g. from the Sun360 dataset [36]), and ground truth equirectangular segmentations, which we generate from already levelled equirectangular images. The dataset we begin with consists only of levelled equirectangular images. For all of these images, we know that the vertical direction is $z = (0, 0, 1)$. By rotating a levelled image with a random rotation R and using eq. (2), we know that the resulting vertical direction of the rotated image will be $R^{-1}z$. From this we can generate training pairs of image and vertical direction. Now, given a vertical direction, it is simple to construct a binary equirectangular segmentation. Let u be the generic vertical direction for some image and I an equirectangular image. After applying $f \circ p$ to all pixel values in I, we can consider the i, j^{th} pixel as sitting at $x_{i,j}$ on the sphere in $\mathbb{R}^3$. Our segmentation $s_{i,j}$ is 1 where $|u \cdot x_{i,j}| > \lambda_{seg}$ and 0 otherwise, which means that we consider pixels that project near to the vertical axis as foreground (1) and all others as background (0). Rotating level images whilst keeping track of the vertical axis allows us to construct many pairs of image and segmentation from a single levelled image.

To actually generate our dataset we compute n_{rot} rotations which will place the vertical axis uniformly around the sphere, and then apply a small offset rotation. Performing these almost uniform rotations avoids using the same n_{rot} rotations for every image, whilst ensuring that the directions completely cover the sphere (c.f. Appendix A). Note there are infinitely many rotations placing the vertical direction at a specific point (by rolling around the vertical axis). We incorporate this rotation online during training, as the rotation can be represented via rolling the equirectangular image along its width axis.

Even when using high quality interpolation methods, we cannot avoid rotational artifacts appearing in the rotated images, which can adversely impact generalization performance. This relatively subtle issue will be discussed in sec. 3.

2.5 Test-time prediction

Once we have an equirectangular segmentation, we can extract a vertical direction by selecting the most probable connected component and taking its centroid

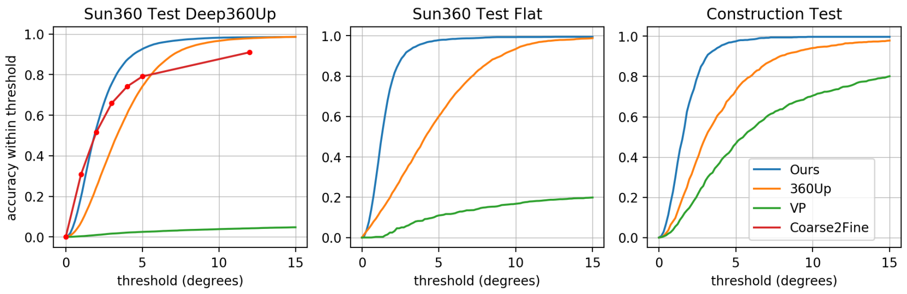

Fig. 6. Accuracy evaluation on the test splits of 3 datasets. From left to right: Sun360 with the same artificial rotations as used by Deep360up [18], Sun360 without rotation, and the construction dataset (sec. 3.3). We report the accuracy over different angular thresholds, for 4 methods: ours, vanishing points (VP, a purely geometric method), and 2 state-of-the-art deep learning methods (360Up and Coarse2Fine). Our method significantly outperforms others on images without artificial rotations (center and right).

as the vertical direction. Given such a centroid c and eq. (1) we recover the vertical direction via $p(c) \in \mathbb{S}^2$. While this is a simple computer vision operation, for completeness we describe it fully in Appendix A.

Test-time augmentation. The final stage of our approach is an optional test time augmentation, which may rotate an image and rerun the segmentation. Let u be a candidate vertical direction obtained after running a single forward pass of the network. If the image was already close to level, *i.e.* u is close to $z = (0, 0, 1)$, then we rotate the images pitch by 20° and rerun the entire inference and post-processing steps to get a new u'. The reason for this is that, if the image is already close to level, the vanishing point features for the vertical axis are close to the points of most distortion: $\pm z$. Following this, we rotate u' back 20° and take the resulting vector as the vertical direction.

Testing data. We collected a test set of unlevelled images, where the vertical direction has been calculated manually. To calculate the vertical direction manually two vertical lines, vertical in the world frame, are manually identified, which allows us to construct a plane parallel to the ground plane, by computing the normals as in sec. 2.3. This plane parallel to the ground trivially gives us the vertical axis, as the axis orthogonal to the plane. By ensuring we use unrotated test images, we avoid data leakage due to rotational artifacts present in the images (see sec. 3).

3 Experiments

We trained and tested our methods on three datasets: the Sun360 dataset, a synthetic dataset of noise, and a dataset of construction images. Training on synthetically rotated Sun360 images can lead to rotational artifacts (sec. 3.2) being learned by the network, a common problem with similar synthetic training regimes [23, 6]. As all images in the Sun360 dataset are level, we could only evaluate our networks' performance on levelled images without introducing rotational artifacts. To measure the extent to which the network relies on rotational artifacts, we created a dataset of rotated noise images. Lastly, to accurately estimate the networks' performance on unlevelled images, without the aid of artifacts, we collected a dataset consisting of images which were not level, and for which the vertical direction was known.

In the following experiments, we compare our method with that of Deep360Up [18] and a baseline vanishing point (VP) method based on [38]. We made use of the publicly available Deep360Up implementation. When possible we also report the performance of the Coarse2Fine [27] approach, by testing on the same test set. On the Sun360 dataset we also show the importance of the vanishing point stream, by removing it from the network and observing a reduced performance.

Finally, we demonstrate the importance of levelling images for downstream tasks by training a segmentation network on levelled and unlevelled images. We make use of our implementation of the original GSCNN work as the segmentation network, and the dataset from [13].

3.1 Sun360 dataset

This dataset consists of 30,000 levelled images, and we use a 80-10-10 split for training, validation and test. As all images in this dataset are already level, we cannot test on any images which do not contain rotational artifacts. To account for this, we evaluated the network on the original, level, images as well as rotated images. We report performance on 3 subsets of data: the unrotated level test set (referred to as `Test Flat`), the unrotated validation set, and a rotated validation set where all vertical directions are in the upper-hemisphere. To compare our method with both Deep360Up and Coarse2Fine we also report results on a synthetically rotated test set, referred to as `Test Deep360Up`, consisting of 17,825 images that were used to evaluate both methods in the original works.

The accuracy of our method as well as the brittleness of the classical vanishing point method is shown in table 1. Our approach is the most accurate on both the level and synthetically rotated test sets, when considering a threshold of at least $2°$. The Coarse2Fine approach does achieve a higher accuracy than our method when considering a $1°$ threshold, but then falls off to be the lowest out of all considered deep learning methods, at larger thresholds. A possible explanation for this dropoff is that the Coarse2Fine approach solves the problem in two stages: first adjusting the image to be within $10°$ of level, and then refining this adjusted image to be within $1°$. Therefore, if the initial estimation is incorrect, the network can never recover the true vertical direction. In contrast, our method is completely end-to-end, and so we do not depend on the output of a previous stage, giving a more robust approximation. Here we also demonstrate the importance of the vanishing point stream, as removing it significantly reduces performance. The poor performance of the vanishing point method is explained

Table 1. Performance for different subsets of the Sun360 dataset (see text for details). We report the percentage of images for which the vertical axis is correctly estimated within a threshold of $x°$.

		Percentage of estimated axes within $x°$							
Dataset	Method	1°	2°	3°	4°	5°	7.5°	10°	12°
Val Rotated	Ours	**25.1**	**60.5**	**80.5**	**90.3**	**94.7**	**97.5**	**98.1**	**98.4**
	Ours (no VP)	20.9	45.6	57.0	74.4	90.2	96.5	97.4	97.8
	Deep360Up	4.9	17.6	34.8	54.7	70.1	91.0	95.7	97.5
	VP	0.2	0.7	1.2	1.5	1.7	2.2	2.5	2.8
Val Flat	Ours	**34.7**	**78.8**	**92.4**	**96.4**	**97.9**	**99.0**	**99.3**	**99.4**
	Ours (no VP)	0.3	1.0	2.9	67.0	97.0	98.9	**99.3**	**99.4**
	Deep360Up	9.8	20.5	33.3	45.5	58.8	81.4	93.2	97.1
	VP	0.5	2.8	5.1	7.3	9.4	12.9	16.0	17.6
Test Deep360	Ours	19.7	**53.6**	**75.5**	**87.2**	**92.6**	**97.1**	**98.2**	**98.4**
	Ours (no VP)	7.5	23.5	40.3	55.9	68.2	87.1	94.8	97.4
	Deep360Up	7.1	24.5	43.9	60.7	74.2	91.9	96.6	97.9
	Coarse2Fine	**30.9**	51.7	65.9	74.1	79.1	NA	NA	91.0
	VP	0.3	0.9	1.6	2.1	2.5	3.3	3.8	4.2
Test Flat	Ours	**34.0**	**78.4**	**92.4**	**96.2**	**97.8**	**98.8**	**99.3**	**99.4**
	Ours (no VP)	0.4	1.3	3.1	63.9	96.5	98.6	99.0	99.2
	Deep360Up	10.2	22.5	35.3	48.2	60.1	82.3	93.4	97.3
	VP	0.3	2.5	6.1	8.8	11.0	14.1	16.7	18.5

by the nature of the Sun360 dataset, which consists of mostly natural scenes (eg. forests), and therefore does not satisfy the Manhattan world assumption.

3.2 Noise dataset

As we synthetically rotate images during training, it was crucial to ensure the network was not “cheating”, i.e. simply using visual artifacts induced by synthetic rotations to solve the problem, and not learning high-level cues that generalize to images with real rotations. Deep networks are very efficient at finding the simplest solution to a problem, and the existence of shortcuts is a prevalent problem in unsupervised learning, for example taking advantage of boundary effects [23, sec. 4.2] or chromatic aberrations of lenses [6, sec. 3.1].

We demonstrate empirically that, in fact, a network can invert a rotation on *pure noise* successfully. To do this, we generated images of random (white) noise, rotated them, and used them to train both our method and the Deep360Up method. We found that in both cases the network could learn to undo the transformation. This highlights the need for an unrotated test set to be sure of the network's performance at test time. Note that we generated a new random noise image at each training and validation step, meaning that this was not a result of over-fitting, as every image the network saw was different. For Deep360Up, we observed the average angular error in this case to be around 5 degrees, and for our method we saw the generalised dice loss fall to 0.04. Both indicate that the network was able to significantly beat chance using only rotational artifacts.

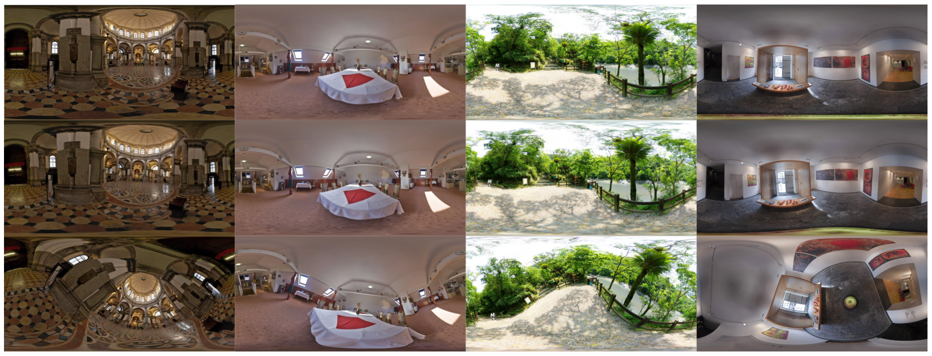

Fig. 7. Visualisation of automatically levelled validation images: proposed method (top), Deep360Up (middle), and the misaligned counterparts (bottom row).

3.3 Construction dataset

To ensure our network was actually solving the problem at hand, we collected a dataset of images from construction sites where we had the raw capture, and the vertical axis of the raw capture. This dataset consists of 10,054 images where we use a 90-10 split for training and validation, and 1006 images for testing. The imbalance in the number of images for training and validation compared to testing images is due to the nature of the data collection process: the training and validation images were already rotated to be level; in contrast, the testing data was gathered manually and consisted of the original capture, which in many cases was not level. This permitted us to test our approach on unlevelled images, that did not contain rotational artifacts. A total of 9365 distinct locations were captured from 16 construction sites, with no overlap in locations between the training and testing data. 48.7% of images were within 3° of level, and 90% were within 12° of level, see figure 8 for typical example scenes from this dataset. Again, our method was considerably more accurate than existing state-of-the-art techniques. Table 2 shows that our approach is the most accurate on all datasets, achieving 98% of estimates within 5 degrees for the test set.

Note that the performance of the vanishing point method on the construction data is significantly better than when applied to the Sun360 data. This can be explained due to the construction dataset consisting of rooms that satisfy the Manhattan world assumption, in contrast to the Sun360 dataset.

3.4 Downstream segmentation task

To illustrate the importance of levelling images for downstream tasks we trained several segmentation models using the dataset in [13], which consists of 666 images from the Sun360 dataset, for which the authors have added segmentation labels for 15 classes. As all images in the Sun360 dataset are already level, we created a rotated segmentation dataset by randomly rotating each image so

Table 2. Performance for different subsets of the construction dataset (see text for details). We report the percentage of images for which the vertical axis is correctly estimated within a threshold of x°.

		Percentage of estimated axes within x°							
Dataset	Method	1°	2°	3°	4°	5°	7.5°	10°	12°
Val. rotated	Ours	**23.1**	**59.0**	**79.2**	**88.2**	**93.2**	**96.4**	**97.2**	**97.5**
	Deep360Up	3.0	12.1	27.4	42.7	58.3	82.5	91.6	95.7
	VP	4.2	13.4	22.3	28.9	33.3	38.9	41.6	43.0
Val. flat	Ours	**25.3**	**66.3**	**87.4**	**93.5**	**96.0**	**97.9**	**98.3**	**98.7**
	Deep360Up	12.4	25.3	39.6	50.8	61.5	82.4	93.5	96.7
	VP	2.8	11.6	25.7	39.1	47.4	60.8	68.6	73.2
Test	Ours	**26.9**	**67.3**	**88.6**	**95.0**	**97.5**	**99.4**	**99.7**	**99.7**
	Deep360Up	9.0	29.3	49.2	62.1	73.0	88.5	94.2	96.2
	VP	4.9	15.1	27.9	38.1	46.7	62.4	70.5	74.9

Table 3. Downstream task performance (mean IOU, in percentages) on different subsets of data (sec. 3.4). For each model we highlight the *worst* performance in bold.

		Evaluation Dataset		
		Original	Levelled	Rotated
Training Dataset	Original	40.1	40.0	**26.7**
	Levelled	42.5	42.0	**31.4**
	Rotated	43.0	42.7	**39.1**

that its vertical direction was at most 45° away from level. We also constructed a levelled dataset by applying our method to the rotated images, using the estimated rotations to level the images and their annotations. In total we used these 3 datasets: original, rotated, and levelled, to train 3 segmentation models.

The segmentation models consist of our own implementation of GSCNNs. Our training regime followed the original work [30] except that we trained for 100 epochs. After training each model, we then evaluated the mean IOU on the original, rotated, and levelled validation sets, consisting of 100 images.

Table 3 shows that all models performed the worst on the rotated dataset, even the model trained specifically on rotated images. This drop in performance is particularly striking for models trained on levelled images, with drops of 13.3% and 10.6% for the model trained on the original dataset and levelled dataset respectively. This highlights a significant problem for many 360° processing methods, which have been trained and evaluated on levelled images. These methods may not generalise well at test time where images may not be level. Our method solves this problem as can be seen in table 3, where the automatically levelled images achieve close to the same performance as the original, levelled dataset.

4 Conclusion

In this paper, we presented the most accurate auto-alignment method to date, developed by combining state-of-the-art segmentation methods with classical

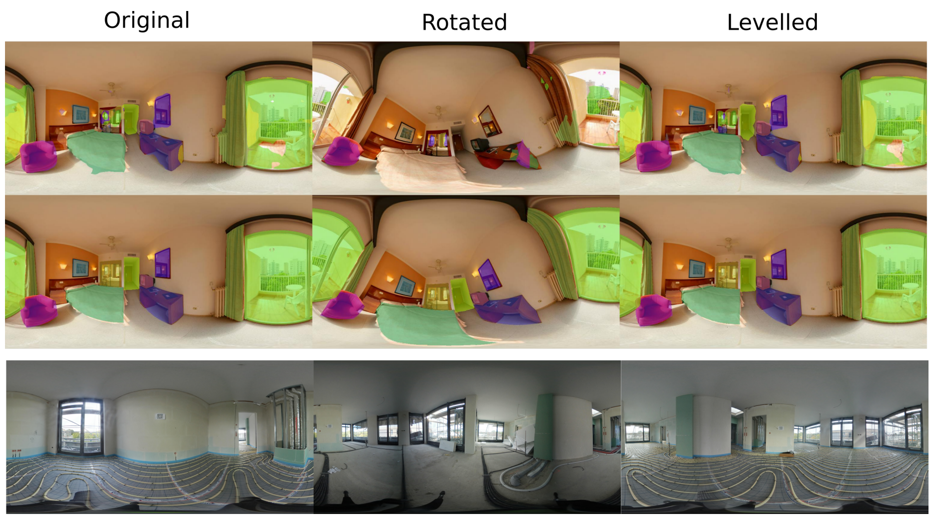

Fig. 8. Top row: Qualitative results for the downstream task of semantic segmentation (sec. 3.4). Middle row: Ground truth. The model performs well on the original images (left), but significantly worse on rotated images (center). Levelling with our method (right) recovers the performance. This highlights the importance of levelling for realistic downstream tasks. Bottom row: Example scenes from the construction dataset.

vanishing point features. We have demonstrated that care needs to be taken when generating training data to avoid data leakage. Moreover, we have demonstrated that casting the vertical axis estimation problem as a segmentation problem results in improved performance, whilst using standard segmentation techniques.

One issue with our approach is that we make the assumption that the vertical direction is already in the upper hemisphere. Though this is a reasonable assumption given how images are captured (where such misalignment is rarely an issue), and the availability of onboard sensors to roughly align an image, we could remedy this problem by instead segmenting the image into three classes: up, down and background. Doing so would allow us to calculate a vertical axis as before, but then use the up or down label to vote for the up direction. Future work could also try directly regressing the location of the vertical direction following the segmentation. We leave this for future work as it would require a considerable modification of the proposed framework.

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
2. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-8**(6), 679–698 (Nov 1986). https://doi.org/10.1109/TPAMI.1986.4767851
3. Coors, B., Paul Condurache, A., Geiger, A.: Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In: The European Conference on Computer Vision (ECCV) (September 2018)
4. Coughlan, J.M., Yuille, A.L.: The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems 13, pp. 845–851. MIT Press (2001), http://papers.nips.cc/paper/1804-the-manhattan-world-assumption-regularities-in-scene-statistics-which-enable-bayesian-inference.pdf
5. Davidson, B., Kalitzeos, A., Carroll, J., Dubra, A., Ourselin, S., Michaelides, M., Bergeles, C.: Automatic cone photoreceptor localisation in healthy and stargardt afflicted retinas using deep learning. Scientific Reports **8**(1), 7911 (2018). https://doi.org/10.1038/s41598-018-26350-3, https://doi.org/10.1038/s41598-018-26350-3
6. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: The IEEE International Conference on Computer Vision (ICCV) (December 2015)
7. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (Jan 1972). https://doi.org/10.1145/361237.361242, https://doi.org/10.1145/361237.361242
8. Fernandez-Labrador, C., Fácil, J.M., Pérez-Yus, A., Demonceaux, C., Guerrero, J.J.: Panoroom: From the sphere to the 3d layout. CoRR **abs/1808.09879** (2018), http://arxiv.org/abs/1808.09879
9. Galamhos, C., Matas, J., Kittler, J.: Progressive probabilistic hough transform for line detection. In: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149). vol. 1, pp. 554–560 Vol. 1 (June 1999). https://doi.org/10.1109/CVPR.1999.786993
10. Gallagher, A.C.: Using vanishing points to correct camera rotation in images. In: The 2nd Canadian Conference on Computer and Robot Vision (CRV'05). pp. 460–467 (May 2005). https://doi.org/10.1109/CRV.2005.84
11. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR 2018 (2018)
12. Google: Google Street View product page. https://www.google.com/streetview/ (2007), [Online, accessed March 2020]
13. Guerrero-Viu, J., Fernandez-Labrador, C., Demonceaux, C., Guerrero, J.J.: What's in my Room? Object Recognition on Indoor Panoramic Images. arXiv e-prints arXiv:1910.06138 (Oct 2019)
14. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, USA, 2 edn. (2003)
15. Henriques, J.F., Vedaldi, A.: Warped convolutions: Efficient invariance to spatial transformations. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1461–1469. JMLR. org (2017)

16. Huang, G., Liu, Z., v. d. Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269 (July 2017). https://doi.org/10.1109/CVPR.2017.243
17. Jeon, J., Jung, J., Lee, S.: Deep upright adjustment of 360 panoramas using multiple roll estimations. In: Jawahar, C., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision – ACCV 2018. pp. 199–214. Springer International Publishing, Cham (2019)
18. Jung, R., Lee, A.S.J., Ashtari, A., Bazin, J.: Deep360up: A deep learning-based approach for automatic vr image upright adjustment. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 1–8 (March 2019). https://doi.org/10.1109/VR.2019.8798326
19. Kendall, A., Grimes, M., Cipolla, R.: Posenet: A convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the IEEE international conference on computer vision. pp. 2938–2946 (2015)
20. Lee, M., Fowlkes, C.C.: Cemnet: Self-supervised learning for accurate continuous ego-motion estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
21. Lezama, J., v. Gioi, R.G., Randall, G., Morel, J.: Finding vanishing points via point alignments in image primal and dual domains. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 509–515 (June 2014). https://doi.org/10.1109/CVPR.2014.72
22. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
23. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision. pp. 69–84. Springer (2016)
24. O'Sullivan, B., Alam, F., Matava, C.: Creating low-cost 360-degree virtual reality videos for hospitals: A technical paper on the dos and don'ts. Journal of medical Internet research **20**(7), e239–e239 (Jul 2018). https://doi.org/10.2196/jmir.9596, https://www.ncbi.nlm.nih.gov/pubmed/30012545, 30012545[pmid]
25. Schindler, G., Dellaert, F.: Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. vol. 1, pp. I–I (June 2004). https://doi.org/10.1109/CVPR.2004.1315033
26. Sellers, G., Wright, R.S., Haemel, N.: OpenGL Superbible: Comprehensive Tutorial and Reference. Addison-Wesley Professional, 7th edn. (2015)
27. Shan, Y., Li, S.: Discrete spherical image representation for CNN-based inclination estimation. IEEE Access **8**, 2008–2022 (2020). https://doi.org/10.1109/ACCESS.2019.2962133
28. Sweeney, C., Flynn, J., Nuernberger, B., Turk, M., Höllerer, T.: Efficient computation of absolute pose for gravity-aware augmented reality. In: 2015 IEEE International Symposium on Mixed and Augmented Reality. pp. 19–24 (Sep 2015). https://doi.org/10.1109/ISMAR.2015.20
29. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (jun 2016). https://doi.org/10.1109/cvpr.2016.308,

30. Takikawa, T., Acuna, D., Jampani, V., Fidler, S.: Gated-SCNN: Gated shape CNNs for semantic segmentation. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
31. Tateno, K., Navab, N., Tombari, F.: Distortion-aware convolutional filters for dense prediction in panoramic images. In: The European Conference on Computer Vision (ECCV) (September 2018)
32. Trombka, J.I., Schweitzer, J., Selavka, C., Dale, M., Gahn, N., Floyd, S., Marie, J., Hobson, M., Zeosky, J., Martin, K., McClannahan, T., Solomon, P., Gottschang, E.: Crime scene investigations using portable, non-destructive space exploration technology. Forensic Science International **129**(1), 1 – 9 (2002). https://doi.org/https://doi.org/10.1016/S0379-0738(02)00079-8, http://www.sciencedirect.com/science/article/pii/S0379073802000798
33. Wallraven, C., Schwaninger, A., Schuhmacher, S., Bülthoff, H.: View-based recognition of faces in man and machine: Re-visiting inter-extra-ortho. vol. 2525, pp. 651–660 (11 2002). https://doi.org/10.1007/3-540-36181-2_65
34. Weiler, M., Hamprecht, F.A., Storath, M.: Learning steerable filters for rotation equivariant CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 849–858 (2018)
35. Xiao, J., Ehinger, K.A., Oliva, A., Torralba, A.: Recognizing scene viewpoint using panoramic place representation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2695–2702 (June 2012). https://doi.org/10.1109/CVPR.2012.6247991
36. Xiao, J., Ehinger, K., Oliva, A., Antonio, T.: Recognizing scene viewpoint using panoramic place representation. In: Proceedings of 25th IEEE Conference on Computer Vision and Pattern Recognition. pp. 0–0 (2012)
37. Xu, J., Stenger, B., Kerola, T., Tung, T.: Pano2cad: Room layout from a single panorama image. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 354–362 (March 2017). https://doi.org/10.1109/WACV.2017.46
38. Zhang, Y., Song, S., Tan, P., Xiao, J.: Panocontext: A whole-room 3d context model for panoramic scene understanding. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 668–686. Springer International Publishing, Cham (2014)