Corners for Layout: End-to-End Layout Recovery from 360 Images –Supplementary Material–

Clara Fernandez-Labrador*[1,2], Jose M. Facil*[1], Alejandro Perez-Yus[1], Cédric Demonceaux[2], Javier Civera[1] and Jose J. Guerrero[1]

***Index Terms*—Omnidirectional Vision, Semantic Scene Understanding**

IN this **supplementary document** we give more details about the network architecture of the proposed model. We also provide more insights about equirectangular convolutions. An extensive explanation about how to create the 3D layout from the corners maps is presented as well. We dedicate also one section to explain how we create the synthetic translations for the proposed robustness analysis. Furthermore, we present an oracle evaluation of our method. We also provide a video to present qualitative differences between using standard convolutions or equirectangular convolutions in our model. Finally, we show additional qualitative results in the SUN360 and Stanford 2D-3D datasets.

I. Implementation

Tables I and II show more details about the network architecture and the differences between the two variants, the first one with standard convolutions, StdConvs, and the second one with Equirectangular Convolutions, EquiConvs. Notice that the decoder uses ReLU as standard activation function except for the prediction layers that use Sigmoid to obtain the edge and corner maps. Prediction layers include those that generate intermediate predictions, **IP-** in Tables I and II, and the final **output** layer.

The blocks for the two models we present are:

CFL StdConvs: Uses the *encoder* and *decoder convs*. In this case *encoder* uses the *std-conv-block* and *std-id-block*, see Tables I and II.

CFL EquiConvs: Uses the *encoder* and *decoder EquiConvs*. In this case *encoder* uses the *equi-conv-block* and *equi-id-block*, see Tables I and II.

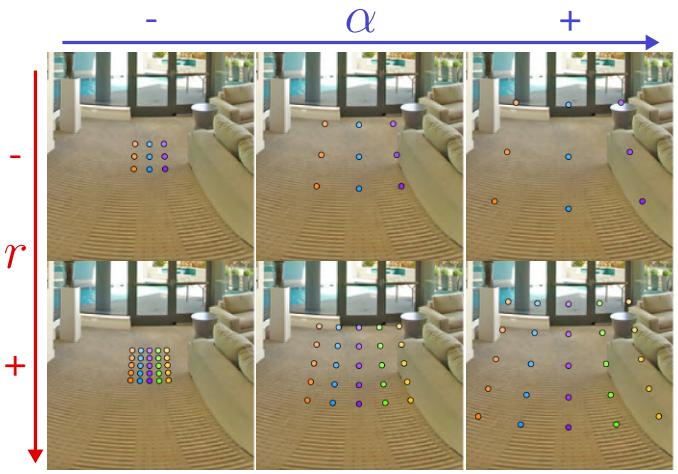

Fig. 1: **Effect of changing field of view α (rad) and resolution r in EquiConvs**. 1st column shows a narrow field of view $\alpha = 0.2$. 2nd column shows a wider kernel keeping its resolution (atrous-like), $\alpha = 0.5$. 3rd column shows an even larger field of view for the kernel, $\alpha = 0.8$. Notice how the kernel adapts to the equirectangular distortion. Rows are resolutions $r = 3$ and $r = 5$.

II. Atrous Equirectangular Convolution

As presented in the paper we represent the kernel of EquiConvs using two parameters, resolution (r) and field of view (α). Although we use by default the same resolution and field of view from the image in our model it can be different. As we increase the resolution of the kernel, the angular distance between the elements decreases, with the intuitive upper limit of not giving more resolution to the kernel than the image itself. In other words, the kernel is defined in a sphere, being its radius less or equal to the image sphere radius. EquiConvs can also be seen as a general model for spherical Atrous Convolutions [2], [3] where the kernel size is what we call resolution, and the **rate** is the field of view of the kernel divided by the resolution. An example of the differences of EquiConvs by modifiying α and r can be seen in Figure 1.

Manuscript received: September, 10, 2019; Revised November, 12, 2019; Accepted January, 9, 2020.

This paper was recommended for publication by Editor Eric Marchand upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Spanish government (project RTI2018-096903-B-I00 and project PGC2018-096367-B-I00), the Aragn regional government (Grupo DGA-T45 17R/FSE) and the Regional Council of Bourgogne-Franche-Comt (2017-9201AAO048S01342).

* Equal contribution

[1] C. Fernandez-Labrador, J.M. Facil, A. Perez-Yus, J. Civera and J.J. Guerrero are with Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain {cfernandez, jmfacil, alperez,jcivera,josechu.guerrero}@unizar.es

[2] C. Fernandez-Labrador and Cédric Demonceaux are with VIBOT ERL CNRS 6000, ImViA, Université Bourgogne Franche-Comté, France. cedric.demonceaux@u-bourgogne.fr

Digital Object Identifier (DOI): see top of this page.

INPUT	LAYER	K	S	FUN	CH	OUTPUT
encoder						
image	conv+BN	7	2	ReLU	64	conv1
conv1	maxpool	3	2	-	64	pool
pool	conv-block	3	1	ReLU	256	res2a
res2a	id-block	3	-	ReLU	256	res2b
res2b	id-block	3	-	ReLU	256	res2c
res2c	conv-block	3	2	ReLU	512	res3a
res3a	id-block	3	-	ReLU	512	res3b
res3b	id-block	3	-	ReLU	512	res3c
res3c	id-block	3	-	ReLU	512	res3d
res3d	conv-block	3	2	ReLU	1024	res4a
res4a	id-block	3	-	ReLU	1024	res4b
res4b	id-block	3	-	ReLU	1024	res4c
res4c	id-block	3	-	ReLU	1024	res4d
res4d	id-block	3	-	ReLU	1024	res4e
res4e	id-block	3	-	ReLU	1024	res4f
res4f	conv-block	3	2	ReLU	2048	res5a
res4a	id-block	3	-	ReLU	2048	res5b
res4b	id-block	3	-	ReLU	2048	res5c
decoder convs						
res5c	upconv	5	2	ReLU	512	upconv4
res4f,upconv4	concat	-	-	-	-	in3
in3	upconv	5	2	ReLU	256	upconv3
upconv3	upconv	3	1	Sigmoid	2	**IP-1**
res3d,upconv3,IP-1	concat	-	-	-	-	in2
in2	upconv	5	2	ReLU	128	upconv2
upconv2	upconv	3	1	Sigmoid	2	**IP-2**
res2c,upconv2,IP-2	concat	-	-	-	-	in1
in1	upconv	5	2	ReLU	64	upconv1
upconv1	upconv	3	1	Sigmoid	2	**IP-3**
conv1,upconv1,IP-3	concat	-	-	-	-	in
in	upconv	3	1	ReLU	64	upconv
upconv	upconv	3	1	Sigmoid	2	**output**
decoder Equiconvs						
res5c	EquiConv	3	1	ReLU	512	equiconv4
equiconv4	unpool	2	2	-	-	unpool4
res4f,unpool4	concat	-	-	-	-	in3
in3	EquiConv	3	1	ReLU	256	equiconv3
equiconv3	unpool	2	2	-	-	unpool3
unpool3	EquiConv	3	1	Sigmoid	2	**IP-1**
res3d,unpool3,IP-1	concat	-	-	-	-	in2
in2	EquiConv	3	1	ReLU	128	equiconv2
equiconv2	unpool	2	2	-	-	unpool2
unpool2	EquiConv	3	1	Sigmoid	2	**IP-2**
res2c,unpool2,IP-2	concat	-	-	-	-	in1
in1	EquiConv	5	1	ReLU	64	equiconv1
equiconv1	unpool	2	2	-	-	unpool1
unpool1	EquiConv	3	1	Sigmoid	2	**IP-3**
conv1,unpool1,IP-3	concat	-	-	-	-	in
in	EquiConv	5	1	ReLU	64	equiconv
equiconv	EquiConv	3	2	Sigmoid	2	**output**

TABLE I: Network Architecture Details. Note that *id-block* and *conv-block* can be either *equi-* or *std-* depending on the version of the network (see Table II).

INPUT	LAYER	K	S	BN	FUN	CH	OUTPUT
std-conv-block							
I	**std-conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	StdConv	1	s		f	N/4	O2a
O2a	StdConv	k	1		f	N/4	O2b
O2b	StdConv	1	1		-	N	O2c
I	conv	I	s		-	N	O-I
O-I,O2c	add	1	1	-	f	N	O
std-id-block							
I	**std-id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	StdConv	1	1		f	N/4	O2a
O2a	StdConv	k	1		f	N/4	O2b
O2b	StdConv	1	1		-	N	O2c
I,O2c	add	I	1	-	f	N	O
equi-conv-block							
I	**equi-conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	EquiConv	1	s		f	N/4	O2a
O2a	EquiConv	k	1		f	N/4	O2b
O2b	EquiConv	1	1		-	N	O2c
I	EquiConv	I	s		-	N	O-I
O-I,O2c	add	1	1	-	f	N	O
equi–id-block							
I	**equi-id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	EquiConv	1	1		f	N/4	O2a
O2a	EquiConv	k	1		f	N/4	O2b
O2b	EquiConv	1	1		-	N	O2c
I,O2c	add	I	1	-	f	N	O

TABLE II: Blocks of the Network

III. From corner maps to 3D layout

Here we provide a further explanation of how the process to go from 2D to 3D works. From the predicted 2D corner positions, we can directly recover the 3D layout by doing the following assumptions:

a) **Soft Manhattan or Atlanta world.** This is a relaxation of the Manhattan World assumption whereby horizontal directions are not necessarily orthogonal to each other. That is, walls can intersect with each other in any direction.

b) **Ceiling-floor parallelism.** Corners can be classified depending on their position along the vertical direction (above or below the horizon line, which in central panoramas is at the middle row) between ceiling and floor corners respectively. Floor corners are on the same floor plane and ceiling corners are directly above the floor ones. The vertical direction is the normal direction of both floor and ceiling planes.

c) **Unitary camera height.** This is trivial as results are up to scale but needed to predict the total height of the room.

Taking all of this into account, we can define a plane as the set of all points $P = (x, y, z)$ such that $P \cdot N + d = 0$, where the normal $N = (n_x, n_y, n_z)$ is a normalized vector perpendicular to its surface and d is the distance that separates it from the origin of coordinates in the direction of the normal. Due to assumptions b) and c), N of both the floor and ceiling planes is equal and corresponds to the vertical direction, and the distance d from the floor to the camera is known. The distance to the ceiling is yet unknown.

Additionally, thanks to the nature of spherical images, we can easily obtain the 3D ray $R(t) = O + \vec{V} \cdot t$ (parametric representation) going from the center of the sphere $O = (o_x, o_y, o_z)$ through the corner position, with normalized direction vector $\vec{V} = (v_x, v_y, v_z)$. To obtain the normalized direction vector $\vec{V}$, we need the corner position in the sphere, thus we transform the image coordinates of the corners (u, v) into spherical coordinates and then to the Euclidean 3D space. Equations for this can be found in Section 4.1. of the paper. In the first place, Eq (1) (6, in the paper) give us the angles that define the point (u, v) in the sphere.

$$\phi = (u - \frac{W}{2})\frac{2\pi}{W} \quad ; \quad \theta = -(v - \frac{H}{2})\frac{\pi}{H} \tag{1}$$

Where W and H are the width and height of the equirectangular image. Second, once these rotations are known we can compute the direction of the ray. Therefore, using Eq (2) (5,

in the paper) we can calculate $\vec{V}$.

$$\vec{V} = \begin{bmatrix} -\cos(\theta)\sin(\phi) \\ \sin(\theta) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \quad (2)$$

The intersection between the corner ray and the corresponding floor or ceiling plane will give us the actual 3D corner point $P = (x, y, z)$ (up to scale), *ie.* the intersection represents that point P on the surface of the plane that verifies the ray equation: $(o_x+v_x \cdot t)n_x+(o_y+v_y \cdot t)n_y+(o_z+v_z \cdot t)n_z+d = 0$. The point P of intersection would simply be the result of evaluating the calculated t, Eq (3), in the ray equation $R(t)$.

$$t = -\frac{o_x n_x + o_y n_y + o_z n_z + d}{v_x n_x + v_y n_y + v_z n_z} \quad (3)$$

Let's consider we have performed the operations to compute one corner point on the floor plane, $P^F = (x^F, y^F, z^F)$. The corresponding point on the ceiling plane (P^C) will be on top of it (*ie.* $x^F = x^C$ and $y^F = y^C$). Therefore, we can use this to compute t^C, Eq (4), and thus the ceiling point:

$$t^C = \frac{(x^F - o_x)}{v_x^C} \quad (4)$$

where $\vec{V}^C = (v_x^C, v_y^C, v_z^C)$ is computed as in (2) with the corresponding ceiling point in the image. Notice that with P^C we have the information we were missing to recover the ceiling plane.

IV. Synthetic translations

To generate the synthetic translations we use the 3D layout reconstruction from the ground truth. The idea is to obtain the color values of each pixel in the new synthetic image, with similar reasoning about ray to plane intersection that can be found in Section III. For each pixel of this image we can recover its spherical coordinates (see Eq. (1)) and the direction of the ray emanating from the reference frame (see Eq. (2)). In Fig. 2 we represent the point in the unitary sphere x' and the corresponding ray R' on the translated reference frame. We compute the intersection of the ray R' and the ground truth layout as a ray to plane intersection as explained in Section III, specifically Eq. (3). To simulate the translation, the origin of the sphere O' will be set accordingly as $O + t$. Since the layout has several planes and the ray could intersect more than one, the closest point is always selected. Once we have the 3D point X we can change its reference frame (*ie.* subtract t) and project it back to the sphere before the translation to recover its point x whose color value can be recovered by going back to the original equirectangular image. We go through all the pixels in the new synthetic image until it is completely filled with color values. Since we only have the ground truth 3D layout but not the complete 3D reconstruction, the objects in the scene could appear deformed because of the change in perspective if translations are too large. The effect of this change in perspective was not noticeable for the translations we applied, and it did not affect the results.

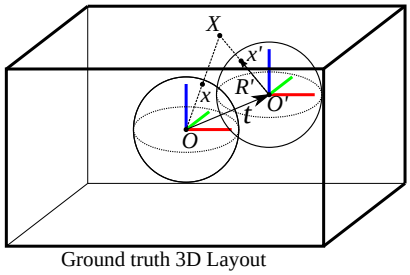

Fig. 2: Generation of synthetic translated image from the ground truth layout. Each point in the sphere in the new reference (x') takes the color from the projection of the 3D intersection point X to the sphere in the old reference (x).

Test	Method	$3DIoU$ %	CE %	PE^{SS} %	PE^{CS} %
SUN360	CFL StdConvs	78.79	0.79	2.49	3.33
	CFL EquiConvs	78.87	0.75	2.6	3.03
	CFL Oracle	97.24	0.07	0.5	0.55
Std.2D3D	CFL StdConvs	65.13	1.44	4.75	6.05
	CFL EquiConvs	62.83	1.33	4.89	6.07
	CFL Oracle	97.39	0.07	0.48	0.54
			smallest the best		

TABLE III: **Layout results** on both datasets, training on SUN360 data. *SS*: Simple Segmentation (3 categories): ceiling, floor and walls [6]. *CS*: Complete Segmentation: ceiling, floor, wall$_1$,..., wall$_n$ [4]. CFL -Convs is our method evaluated using the output of our network with one type of convolution, CFL Oracle represents our method when having a perfect corner map from the neural network.

V. Oracle Corner Maps

In order to better understand our pipeline performance, we have decouple it into two parts. The CNN prediction and the Layout estimation from corner maps. We have evaluated and compared our pipeline with GT corner maps, *i.e.* we remove the CNN and evaluate our layout recovery as if we would have get perfect corner maps from the CNN. This gives us an upper bound for the CNN prediction and tell us where are we now w.r.t. this upper bound. In Table III we show these numbers in two public datasets. We can see two things in this table. 1) The second part of our pipeline, when having good quality corner maps, *CFL Oracle* in the table, performs very accurate. This means it is a good approach if we assume we will have good corner predictions. 2) Making our networks more accurate could improve around 20% our accuracy.

VI. CFL Video

The aim of the supplementary video is to show, for each tested panorama, the predictions obtained through our two CFL models, StdConvs and EquiConvs, when each panorama is rotated from 0° to 360° horizontally.

If we pay attention to both ends of the predictions (left and right borders) we can see a meaningful difference between both models, specifically when corners get closer to the borders of the image.

With EquiConvs, we do not use padding when the kernel reaches the border of the image since offsets take the points to

their correct position on the other side of the 360 image. This allows the model to understand the continuity of the scene. StdConvs, instead, by using zero-padding, do not consider relationship between borders of equirectangular images.

This means that EquiConvs, working in a continuous spherical space, produce much more consistent predictions along all the performed rotations, whereas StdConvs lose consistency between left and right sides of the panoramas.

As a consequence of that, in most cases when corners approach the borders, StdConvs predict these corners twice, at both ends, and thus layout estimations are wrong. This effect is highlighted in the video with red bounding boxes.

The supplementary video can be found here: https://youtu.be/dK$_v sVYiPaA$

VII. Qualitative Results

Here we show additional qualitative results of our recovered layouts in SUN360 and Stanford 2D-3D datasets.

Figures 3 and 4 collect examples in SUN360 dataset and show indoor scenes with different geometries, not only cuboid shapes. Figure 5 shows examples in Stanford 2D-3D dataset. Panoramas in this dataset do not cover full view vertically and the indoor scenes represent more challenging scenarios like cluttered laboratories or corridors.

References

[1] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv*, Feb. 2017. 7

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 1

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1

[4] C. Fernandez-Labrador, A. Perez-Yus, G. Lopez-Nicolas, and J. J. Guerrero. Layouts from panoramic images with geometry and deep learning. *arXiv preprint arXiv:1806.08294*, 2018. 3

[5] J. Xiao, K. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2695–2702, 2012. 5, 6

[6] C. Zou, A. Colburn, Q. Shan, and D. Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018. 3

Fig. 3: Layout predictions (light magenta) and ground truth (dark magenta) on the SUN360 annotation dataset [5]. Best viewed in color.

Fig. 4: Layout predictions (light magenta) and ground truth (dark magenta) for **complex room geometries** on the SUN360 annotation dataset [5]. Best viewed in color.

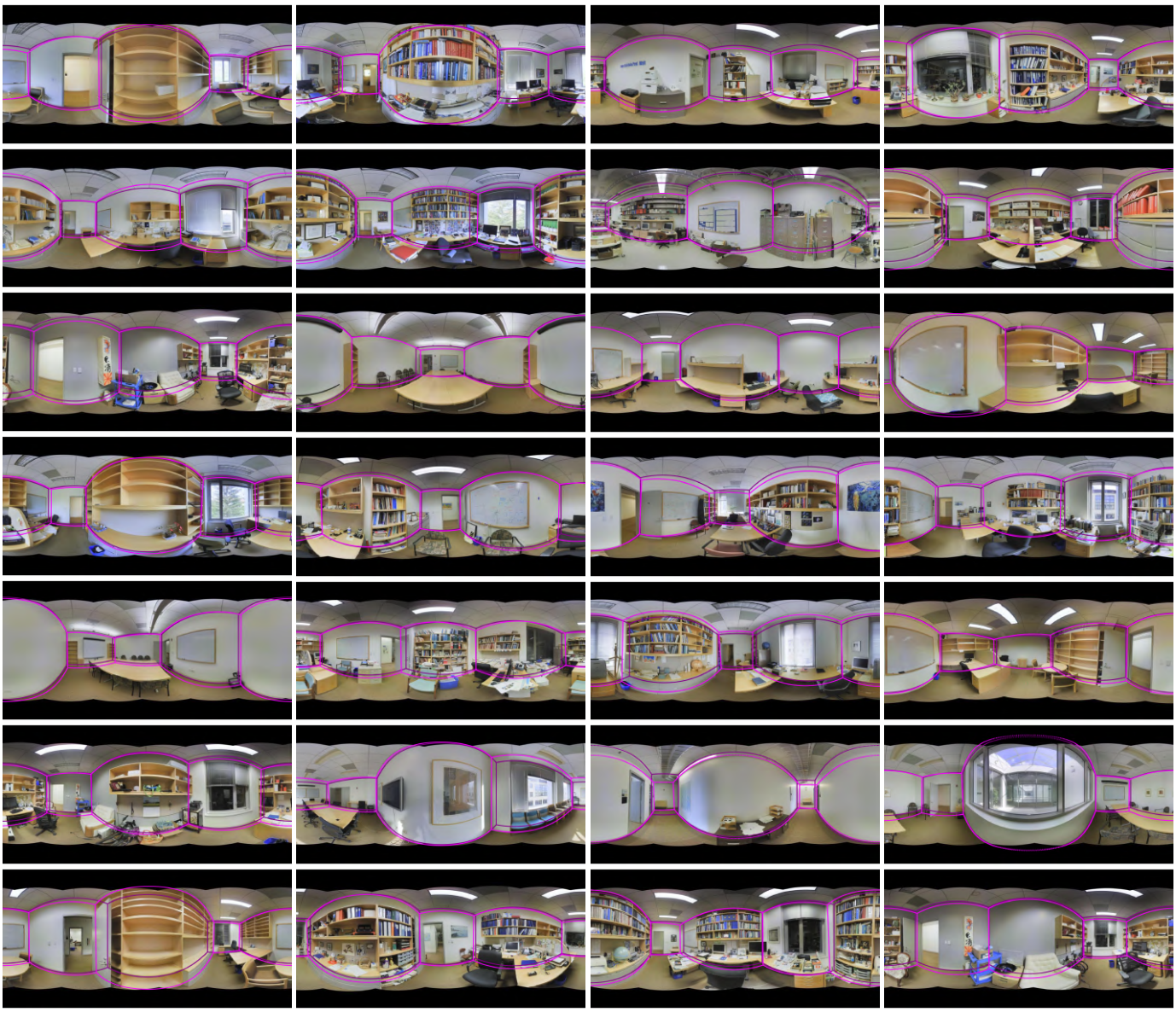

Fig. 5: Layout predictions (light magenta) and ground truth (dark magenta) on the Stanford 2D-3D annotation dataset [1]. Best viewed in color.