CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth – Supplementary Material –

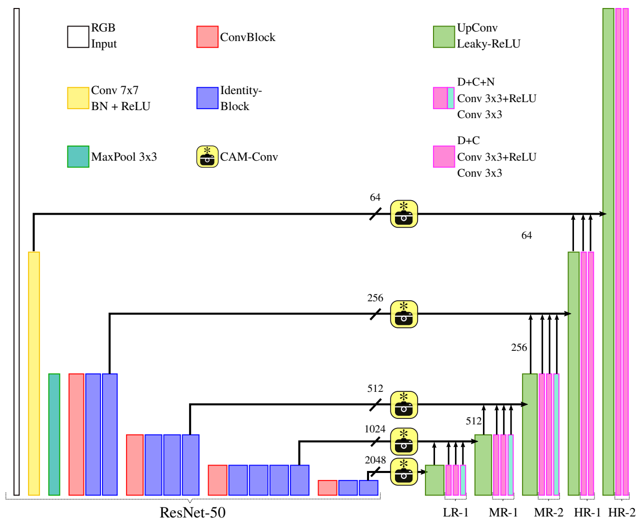

Figure 1. Figure taken from the main paper. Our network architecture, inspired by DispNet [10], to which we added CAM-Convs connecting the encoder and decoder. We predict depth, confidence and normals (D+C+N) in the first three intermediate resolution levels (LR-1,MR-1 and MR-2) and only depth and confidence (D+C) in the last two resolution levels (HR-1 and HR-2).

A. Model and Training

A.1. Network Architecture

As it is explained in the main paper, the network we use in this work has an encoder-decoder architecture inspired by DispNet [10]. We add skip-connections from the low-level feature maps of the encoder to the feature maps of the same size in the decoder, and concatenate them [11]. We do predictions at intermediate resolutions, which helps the training to converge faster and ensures that the internal features are more aimed for the task. As it is common in the literature [8, 7], our network's backbone is ResNet-50, pre-trained on the ImageNet Classification Dataset [5]. As suggested in the literature [4] and as our experiments confirm, pretraining the encoder on general image recognition tasks, as ImageNet, helps both in accuracy and convergence. A scheme of our network architecture can be seen in Figure 1. A more detailed view of the network implementation can be seen in Table 1 (some of the blocks used in this table are further detailed in Table 2). Bold output columns represent different pyramid-resolution predictions of the network.

The network predictions are:

ξ**: Inverse depth** $\xi = \frac{1}{d}$**.** We chose inverse depth for its linear relationship with the image space.

c**: Depth confidence.** As [14], we enforce the network to predict a confidence map for every depth prediction.

$\mathbf{n}$**: Surface normals.** The normals are predicted only for small resolutions (all except the last two), as the ground-truth normals are noisy at full resolution.

Predictions LR-1, MR-1, and MR-2 are composed of inverse depth, depth confidence and surface normals (a total of 5 channels see Table 1). HR-1 and HR-2 are composed only of inverse depth and depth confidence (a total of 2 channels see Table 1).

A.2. Training Schedule

We train all our networks using the TensorFlow framework [1]. We start from ResNet-50 (pre-trained) and a randomly initialized decoder. For optimization we use Adam [6] with a momentum of 0.9. The complete training of the network is composed by three different stages, each adding more layers and predictions to the decoder (SR,MR and HR respectively in Table 1).

1st stage. We train until the first two resolutions of the decoder (LR-1 in Table 1). This stage is the shortest one, trained only for $10k$ iterations with batch size 16. We only train the encoder layers and the decoder until the smallest prediction. We do not apply the scale-invariant loss for this resolution.

2nd stage. We train until the next two predictions (LR-1, MR-1 and MR-2 in Table 1). As in the previous stage we train only the layers that affect the outputs. This stage is trained for $50k$ iterations with batch size 16. We apply a scale-invariant loss to prediction MR-2 after $25k$ iterations.

3rd stage. We train the whole network, for $200k$ iterations with batch size 16. We apply a scale-invariant loss to predictions MR-2, HR-1 and HR-2 after $25k$ iterations.

Learning rate. The learning rate policy for the three stages is shown in Figure 2. The base learning rates for the three stages are $1e-3$, $5e-4$ and $1e-4$. The learning rate drops along the iterations, never being less than a minimum of

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
			skip-connections			
conv1	CAM-Conv	3	1	L-ReLU	64	conv1
res2c	CAM-Conv	3	1	L-ReLU	256	res2c
res3d	CAM-Conv	3	1	L-ReLU	512	res3d
res4f	CAM-Conv	3	1	L-ReLU	1024	res4f
res5c	CAM-Conv	3	1	L-ReLU	2048	res5c
			decoder			
res5c	upconv	4	2	L-ReLU	1024	upconv4
upconv4	conv	3	1	L-ReLU	24	inconv4
inconv4	conv	3	1	-	5	**LR-1**
LR-1,upconv4	concat	-	-	-	-	up-pre4
res4f,up-pre4	concat	-	-	-	-	in3
in3	upconv	4	2	L-ReLU	512	upconv3
upconv3	conv	3	1	L-ReLU	24	inconv3
inconv3	conv	3	1	-	5	**MR-1**
MR-1,upconv3	concat	-	-	-	-	up-pre3
res3d,up-pre3	concat	-	-	-	-	in2
in2	upconv	4	2	L-ReLU	256	upconv2
upconv2	conv	3	1	L-ReLU	24	inconv2
inconv2	conv	3	1	-	5	**MR-2**
MR-2,upconv2	concat	-	-	-	-	up-pre2
res2c,up-pre2	concat	-	-	-	-	in1
in1	upconv	4	2	L-ReLU	64	upconv1
upconv1	conv	3	1	L-ReLU	24	inconv1
inconv1	conv	3	1	-	2	**HR-1**
HR-1,upconv2	concat	-	-	-	-	up-pre1
conv1,up-pre1	concat	-	-	-	-	in
in	upconv	4	2	L-ReLU	32	upconv
upconv	conv	3	1	L-ReLU	24	inconv
inconv	conv	3	1	-	2	**HR-2**

Table 1. Network Architecture Details. We present three different blocks here. **1st**: *encoder* taken directly from ResNet-50, pretrained on the ImageNet Classification Dataset [5], sub-blocks of the *encoder* are more detailed in Table 2. **2nd**: *skip-connections* include also a CAM-Conv block (Table 2). **3rd**: *decoder* uses as inputs all the *skip-connections* and produce 5 different pyramid-resolution predictions (**bold** in the OUTPUT column).

$1e{-}6$. As we train multiple models we use a fixed automatic learning rate decay.

Weighting losses. For all stages we minimize the losses for several resolutions. We scale the losses according to the resolution level. Specifically, we multiply the losses by a factor $\frac{1}{k}$, where k denotes the resolution level. Starting with the finest resolution of the active stage. *I.e* in the 1st stage LR-1 prediction loss would be multiplied by 1. While in the 3rd stage LR-1 would be multiplied by $\frac{1}{5}$, MR-1 would be multiplied by $\frac{1}{4}$ and so on until HR-2 that would be multiplied by 1.

INPUT	LAYER	K	S	BN	FUN	CH	OUTPUT
			conv-block				
I	**conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	conv	1	s	✓	f	N/4	O2a
O2a	conv	k	1	✓	f	N/4	O2b
O2b	conv	1	1	✓	-	N	O2c
I	conv	1	s	✓	-	N	O-1
O-1,O2c	add	1	1	-	f	N	O
			id-block				
I	**id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	conv	1	1	✓	f	N/4	O2a
O2a	conv	k	1	✓	f	N/4	O2b
O2b	conv	1	1	✓	-	N	O2c
I,O2c	add	1	1	-	f	N	O
			CAM-Conv-block				
I	**CAM-Conv**	**k**	**s**	**-**	**f**	**N**	**O**
fov,cc,nc	resize	-	-	-	-	-	fov,cc,nc
I,fov,cc,nc	concat	-	s	-	-	-	intern
intern	conv	k	s	-	f	N	O

Table 2. Sub-Blocks for the Network. The first two are the standard conv and id block from ResNet, commonly used in the literature. The **CAM-Conv-block** is the one we introduce. It takes as input 3 different maps and feeds a convolution with them, together with the input features.

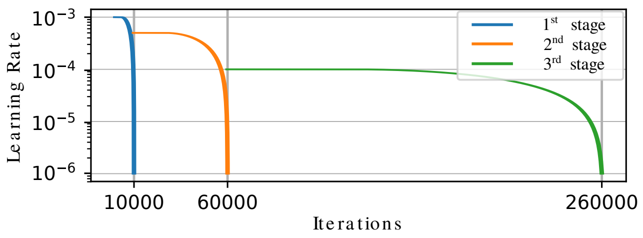

Figure 2. Learning rate policy for the three-stages training.

Area #	Eq. Images
	(# images)
1	190
2	299
3	85
4	258
5	373
6	208
Total	**1413**

Table 3. Statistics of images in the 2D-3D Semantics Dataset [2]. Number of images per area.

B. Experiments on Stanford Dataset

B.1. 2D-3D Semantics Stanford Dataset

In our experiments we used the 2D-3D Semantics Dataset [2], that contains RGB-D equirectangular images. With these images we are able to generate synthetic images with different camera intrinsics. This is essential to evaluate the influence of such intrinsic camera parameters minimiz-

Fold #	**Training**	**Testing**
	(Area #)	(Area #)
1	1,2,3,4,6	6
2	1,3,5,6	2,4
3	2,4,5	1,3,6

Table 4. Area assignation for the 3-fold cross validation scheme presented in [2] for the 2D-3D Semantics Dataset.

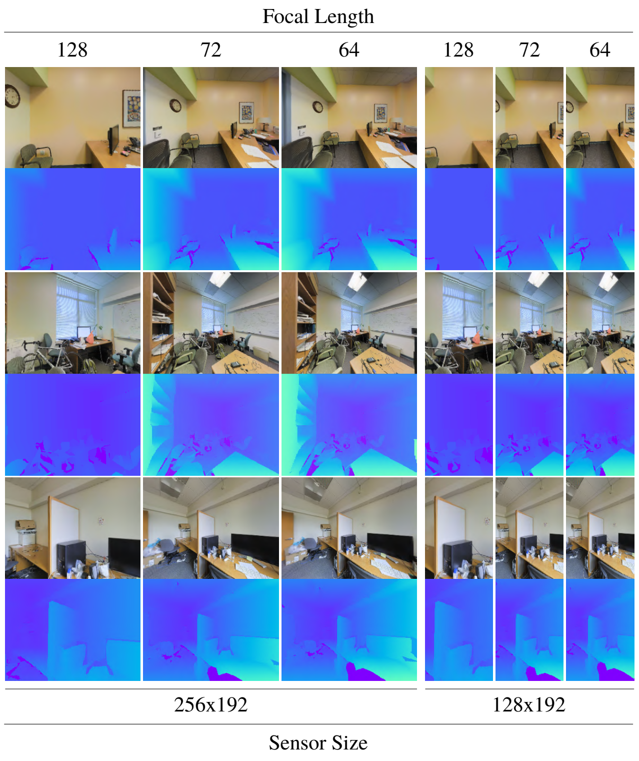

Figure 3. Examples of 3 different scenes (in rows) for which we generated several images with different intrinsic parameters.

Camera Params	**Range**
	(uniform random sample)
yaw	$(-180°, 180°)$
$pitch$	$(-6°, 6°)$
$roll$	$(-6°, 6°)$
x, y, z	*fixed*[*]
f (focal length)	*depending experiment*
$w \times h$	*depending experiment*

[*] the position of the camera provided for the equirectangular images remains unchanged, we do not use this parameter to generate more images.

Table 5. Statistics of images in the 2D-3D Semantics Dataset [2]. Number of images per area.[*] The position of the camera provided for the equi-rectangular images remains unchanged, we do not use this parameter to generate more images.

ing the effect of the dataset bias.

The dataset is divided into 6 different areas, see Table 3. The areas in the dataset represent parts of buildings with similarities in their appearance. We used the official train and test splits as suggested by the authors in [2]. The suggested 3-fold cross-validation scheme is shown in Table 4. All the experiments in this supplementary material, unless explicitly stated, are performed on the 2D-3D Semantics Dataset.

When training on this dataset, we generate images by randomizing camera parameters. Some parameters remain fixed depending on the experiment (*e.g.* sensor size is fixed during training). Different values of these parameters can be found in Table 5. Figure 3 shows several examples of images generated with different camera intrinsics.

Name	s_1	s_2	s_3
Sensor	256×192	192×256	224×224

Name	s_4	s_5	s_S	s_K
Sensor	128×96	320×320	256×192	384×128

Name	f_{72}	f_{128}	f_{64}	f_n
Focal	72	128	64	100

Table 6. Table from the main paper. Notation for different sensor sizes and focal lengths.

B.2. Notation

The notation for sensor sizes and focal lengths used during the evaluation is in Table 6. As an example, if a network has been trained with sensor sizes 192×256 and 224×224, and focal length 72, we will denote this model as $s_2 s_3 f_{72}$. In some experiments we use a random distribution for the focal length. As an example, if the synthesized focal lengths are uniformly distributed between 72 and 128, the model will be denoted as $\mathcal{U} f_{72} f_{128}$.

B.3. Influence of Context in Single View Depth

In this section we present the complete results for the experiment analyzing the influence of context. Single-view depth prediction is a task heavily related to context, and that benefits significantly from having more image content.

Context by reducing focal length: Table 7 shows a detailed version of the context influence experiment. In this experiment we train our network without CAM-Convs with images of different focal lengths (f_{64}, f_{72} and f_{128}) and a fixed sensor size s_1. We report the results for three folds separately, and the median error.

Context by augmenting sensor size: Table 8 shows a detailed version of the experiment of different versions of the network without CAM-Convs trained on different sensor sizes (s_1, s_4) with a fixed focal length f_{64}. The results are shown for the three folds separately and the

Focal Length	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
pixels		$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
	1	0.17	0.188	0.385	0.0286	0.0428	70.1	96.1	99.3
f_{64}	2	0.198	0.218	0.432	0.05	0.0691	65.1	90.7	98.3
	3	0.155	0.151	0.326	0.0305	0.0451	78.9	97.1	99.6
	1	0.168	0.169	0.411	0.0292	0.0427	70.8	96.2	99.3
f_{72}	2	0.206	0.203	0.474	0.052	0.074	62.8	90.4	97.9
	3	0.147	0.143	0.31	0.0302	0.0403	80.3	97.2	99.6
	1	0.197	0.146	0.547	0.0342	0.0553	61.0	93.6	99.0
f_{128}	2	0.23	0.156	0.586	0.0525	0.0837	56.1	88.2	97.7
	3	0.166	0.117	0.409	0.0329	0.0483	74.5	96.4	99.5
f_{64}	M_e	**0.17**	0.184	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
f_{72}	M_e	**0.17**	0.17	0.4	0.0354	0.0483	71.9	95.3	**99.2**
f_{128}	M_e	0.195	**0.141**	0.51	0.0387	0.0606	64.4	93.3	99.0
		smallest the best					*biggest the best*		

Table 7. Experiment comparing performance on single view depth prediction by changing focal lengths. We evaluate three different focal lengths (f_{64},f_{72} and f_{128}) fixing the sensor size to s_1. The upper part of the table shows the results for each fold. The bottom part shows the median M_e of the three folds. As expected having more context improves the depth prediction. In this experiment, focal length f_{64} has the best performance.

Sensor Size	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
pixels		$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
	1	0.17	0.188	0.385	0.0286	0.0428	70.1	96.1	99.3
s_1	2	0.198	0.218	0.432	0.05	0.0691	65.1	90.7	98.3
	3	0.155	0.151	0.326	0.0305	0.0451	78.9	97.1	99.6
	1	0.208	0.155	0.598	0.0348	0.0607	55.1	92.4	98.9
s_4	2	0.24	0.162	0.606	0.051	0.0901	54.2	87.4	97.8
	3	0.171	0.121	0.425	0.0319	0.0492	73.0	96.5	99.5
s_1	M_e	**0.17**	0.184	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
s_4	M_e	0.204	**0.146**	0.54	0.0384	0.0637	61.3	93.0	99.0
		smallest the best					*biggest the best*		

Table 8. Experiment comparing performance on single view depth prediction by changing sensor sizes. We evaluate two different sensor sizes (s_4 ands_1) fixing the focal length to f_{64}. The upper part of the table shows the results for each fold. The bottom part shows the median M_e of the three folds. As expected having more context improves the depth prediction. In this experiment, sensor size s_1 has the best performance.

median error.

As expected, in both experiments, the results are better for those images with a wider field of view, as the camera captures a larger part of the scene. In the case of fixed sensor size, this corresponds to the camera with the smallest focal length; in the case of fixed focal length, it is the camera with the biggest sensor size. As a curiosity, we found that the L_1 norm on inverse depth (l1.inv in the table) obtains a better score with smaller context in both experiments. We attribute this to the smaller weight that this metric gives to large errors, suggesting that context does not improve the errors uniformly but reduces by a bigger amount large prediction errors.

B.4. Focal Length Overfitting

In this section we show the complete results for the focal length overfitting experiments. Table 9 shows the results by fold and Table 10 shows the median for all the images on every fold. Unless it is specified otherwise, all the networks have been trained with focal length normalization. We distinguish between three test sets, each one of them generated with one focal length: f_{64}, f_{72} and f_{128}. We train networks on each one of these focal lengths and also train on multiple focal lengths with ($f_{72}f_{128}$ and $\mathcal{U}f_{72}f_{128}$) and without focal length normalization ($f_{72}f_{128}{}^*$).

Notice how networks that have been trained and tested with the same focal length obtain the best results. Also observe how networks that have been trained only in one focal length, generalize poorly when they are tested on different focal length. We also evaluated training on multiple focal lengths without focal length normalization ($f_{72}f_{128}{}^*$ in the table). This does not only perform worse than other approaches, but its convergence was difficult. This is expected, as not normalizing leads to inconsistent target depths.

We found that networks trained on smaller focal lengths tend, in general, to generalize better to bigger focals than the opposite case. See for example the test set f_{128}.

B.5. Sensor Size Overfitting

In this section we show the complete results for the two sensor size overfitting experiments. We compare the network version without CAM-Convs trained and tested on

Test f	**Train f**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
$pixels$	$pixels$		$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
f_{64}	f_{64}	1	0.173	0.187	0.391	0.0286	0.0431	69.5	96.0	99.3
		2	0.2	0.212	0.423	0.0492	0.0708	65.2	91.1	98.3
		3	0.154	0.16	0.359	0.0347	0.0422	78.1	96.7	99.4
	f_{72}	1	0.185	0.203	0.424	0.0293	0.0481	64.5	95.2	99.2
		2	0.205	0.223	0.457	0.0519	0.0706	63.2	90.3	98.0
		3	0.145	0.158	0.305	0.0301	0.0376	80.9	97.3	99.5
	f_{128}	1	0.288	0.377	0.647	0.0424	0.101	25.7	75.5	94.6
		2	0.246	0.314	0.604	0.0662	0.0885	46.8	81.9	95.2
		3	0.206	0.259	0.45	0.0415	0.0604	56.3	91.0	98.3
	$f_{72}f_{128}$*	1	0.427	0.685	0.835	0.0405	0.196	02.2	23.6	73.7
		2	0.359	0.526	0.765	0.0635	0.149	11.8	54.6	86.6
		3	0.428	0.726	1.0	0.0798	0.227	06.8	30.6	67.1
	$f_{72}f_{128}$	1	0.183	0.199	0.414	0.0311	0.048	66.0	95.0	99.2
		2	0.205	0.217	0.433	0.0514	0.074	64.3	90.6	98.1
		3	0.148	0.154	0.313	0.0307	0.0403	80.0	97.2	99.5
	$\mathcal{U}f_{72}f_{128}$	1	0.186	0.204	0.429	0.0314	0.0493	64.0	94.5	99.1
		2	0.213	0.224	0.457	0.055	0.08	62.2	89.6	97.7
		3	0.155	0.159	0.318	0.0321	0.0442	78.3	96.9	99.5
f_{72}	f_{72}	1	0.18	0.178	0.43	0.0306	0.0468	67.3	95.5	99.2
		2	0.208	0.203	0.474	0.0524	0.0747	62.5	89.8	97.8
		3	0.148	0.146	0.315	0.0302	0.0393	80.3	97.2	99.5
	f_{128}	1	0.272	0.32	0.643	0.0407	0.0925	30.8	79.6	95.8
		2	0.238	0.271	0.608	0.0624	0.0854	50.7	84.0	95.9
		3	0.193	0.22	0.441	0.0392	0.055	60.6	93.1	98.7
	$f_{72}f_{128}$*	1	0.408	0.582	0.862	0.0407	0.181	03.4	30.2	79.1
		2	0.345	0.45	0.798	0.0632	0.141	15.5	60.3	88.4
		3	0.406	0.6	1.02	0.0801	0.208	09.4	36.8	73.0
	f_{72},f_{128}	1	0.18	0.181	0.432	0.0312	0.0468	66.9	95.2	99.2
		2	0.21	0.198	0.458	0.0525	0.0766	63.6	90.0	97.9
		3	0.153	0.148	0.328	0.0311	0.0428	78.7	97.0	99.6
	$\mathcal{U}f_{72}f_{128}$	1	0.183	0.184	0.445	0.0312	0.0483	65.6	94.8	99.1
		2	0.219	0.205	0.475	0.0545	0.0809	61.7	89.2	97.7
		3	0.161	0.149	0.332	0.0322	0.0464	77.4	96.6	99.5
f_{128}	f_{72}	1	0.179	0.124	0.497	0.0353	0.0504	69.3	95.6	99.3
		2	0.259	0.15	0.623	0.0553	0.105	51.3	84.7	97.4
		3	0.206	0.129	0.472	0.0359	0.0734	66.6	94.2	99.4
	f_{128}	1	0.211	0.158	0.605	0.0352	0.0608	53.9	92.1	98.7
		2	0.232	0.159	0.593	0.053	0.0846	55.6	88.1	97.7
		3	0.172	0.127	0.424	0.0346	0.0485	72.0	96.1	99.4
	$f_{72}f_{128}$*	1	0.261	0.202	0.755	0.046	0.0906	37.8	81.5	97.1
		2	0.287	0.208	0.754	0.0658	0.116	39.3	78.0	94.7
		3	0.275	0.215	0.974	0.0806	0.134	40.9	76.7	93.4
	$f_{72}f_{128}$	1	0.183	0.134	0.513	0.0329	0.0493	66.3	95.0	99.3
		2	0.23	0.142	0.576	0.0501	0.0872	57.5	88.6	98.1
		3	0.178	0.12	0.437	0.0329	0.0552	73.1	96.1	99.5
	$\mathcal{U}f_{72}f_{128}$	1	0.184	0.131	0.509	0.0334	0.0497	65.7	95.2	99.3
		2	0.245	0.147	0.596	0.0521	0.0958	54.5	87.1	97.8
		3	0.189	0.123	0.449	0.0332	0.0633	70.5	95.6	99.5
			smallest the best					*biggest the best*		

* the network has been trained without focal length normalization, it is included in this table as a baseline.

Table 9. Experiments on multiple focal lengths, we evaluate on f_{64},f_{72} and f_{128}. This table shows the results for each fold on each one of the test sets.

different sensor sizes.

Sensor size overfitting: Our first experiment, see Table 11, shows a by-fold cross-comparison of training a testing in three different sensor sizes, s_1, s_2 and s_3. Results show that training and testing on the same sensor size always performs better. This suggest overfitting to the sensor size, and shows that the oftenly claimed invariance to the image size of fully convolutional networks (FCN) does not hold for single-view depth prediction.

In our second experiment we allow training on multiple image sizes. In order to do that, we propose two different approaches.

Training on different image sizes (1): Our fist ap-

Test f	Train f	Fold #	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
pixels	*pixels*		$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
	f_{64}*	M_e	**0.17**	**0.184**	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
	f_{64}	M_e	0.172	**0.184**	0.392	0.0364	0.0488	71.6	95.2	**99.2**
	f_{72}	M_e	0.174	0.193	0.395	0.0354	0.0486	70.4	95.0	99.1
f_{64}	f_{128}	M_e	0.247	0.318	0.572	0.0483	0.0826	42.9	83.6	96.4
	$f_{72}f_{128}$*	M_e	0.41	0.659	0.864	0.0614	0.193	6.3	34.4	74.7
	$f_{72}f_{128}$	M_e	0.176	0.189	0.387	0.0361	0.0503	70.5	94.9	99.1
	$\mathcal{U}f_{72}f_{128}$	M_e	0.182	0.194	0.398	0.0374	0.0533	68.8	94.4	99.1
	f_{72}*	M_e	**0.17**	**0.17**	**0.4**	**0.0354**	**0.0483**	**71.9**	**95.3**	**99.2**
	f_{72}	M_e	0.175	0.174	0.407	0.0364	0.0503	70.6	94.9	99.1
f_{72}	f_{128}	M_e	0.235	0.272	0.564	0.0459	0.076	46.7	86.5	97.1
	f_{72},f_{128}*	M_e	0.391	0.552	0.888	0.0609	0.179	8.6	41.2	79.5
	f_{72},f_{128}	M_e	0.178	0.175	0.404	0.0364	0.052	70.3	94.8	**99.2**
	$\mathcal{U}f_{72}f_{128}$	M_e	0.184	0.179	0.414	0.0378	0.0553	68.9	94.4	99.1
	f_{128}*	M_e	0.195	0.141	0.51	0.0387	**0.0606**	64.4	93.3	99.0
	f_{72}	M_e	0.213	0.133	0.524	0.0411	0.0744	63.4	92.5	99.0
f_{128}	f_{128}	M_e	0.202	0.149	0.532	0.0396	0.0625	61.4	92.7	98.9
	f_{72},f_{128}*	M_e	0.273	0.208	0.813	0.063	0.11	39.6	78.6	95.1
	f_{72},f_{128}	M_e	**0.194**	**0.132**	**0.504**	**0.038**	0.0616	**66.2**	**93.9**	**99.2**
	$\mathcal{U}f_{72}f_{128}$	M_e	0.202	0.134	0.512	0.0385	0.0663	64.1	93.5	99.1
			smallest the best					*biggest the best*		

* the network has been trained without focal length normalization, it is included in this table as a baseline.

Table 10. Experiments on multiple focal lengths, we evaluate on f_{64},f_{72} and f_{128}. This table shows the median M_e results for each fold on each one of the test sets.

Test	Train	Fold #	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
			$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
		1	0.181	0.151	0.475	0.0319	0.0481	66.8	95.2	99.3
	s_1	2	0.229	0.169	0.526	0.0533	0.085	59.3	88.5	98.0
		3	0.171	0.132	0.391	0.0317	0.0528	75.0	96.5	99.5
		1	0.21	0.18	0.537	0.0351	0.0595	53.3	92.4	98.8
s_1	s_2	2	0.24	0.189	0.581	0.0565	0.0873	54.5	86.9	97.2
		3	0.182	0.159	0.422	0.0393	0.0505	66.6	95.2	99.3
		1	0.199	0.168	0.519	0.0328	0.0555	59.0	93.9	99.0
	s_3	2	0.228	0.172	0.535	0.0523	0.0831	59.5	88.3	97.8
		3	0.164	0.135	0.38	0.0333	0.0452	75.5	96.8	99.5
		1	0.163	0.129	0.432	0.0339	0.0426	72.6	96.2	99.5
	s_1	2	0.243	0.185	0.506	0.0546	0.0945	56.4	89.6	98.0
		3	0.196	0.146	0.393	0.0312	0.0652	70.6	95.5	99.5
		1	0.152	0.121	0.409	0.0263	0.037	77.4	97.2	99.5
s_2	s_2	2	0.207	0.162	0.493	0.0462	0.0748	64.8	91.7	98.5
		3	0.152	0.123	0.347	0.0293	0.0415	79.6	97.3	99.7
		1	0.157	0.126	0.428	0.0294	0.0398	73.9	96.6	99.5
	s_3	2	0.22	0.17	0.485	0.0478	0.0811	62.8	91.3	98.5
		3	0.158	0.123	0.353	0.028	0.0457	78.7	96.9	99.7
		1	0.16	0.129	0.428	0.0312	0.0413	72.9	96.2	99.4
	s_1	2	0.231	0.171	0.51	0.0513	0.0868	60.1	89.9	98.2
		3	0.179	0.135	0.387	0.0302	0.0566	73.6	96.2	99.6
		1	0.172	0.144	0.438	0.0284	0.0429	70.4	96.1	99.3
s_3	s_2	2	0.209	0.164	0.511	0.0483	0.0761	63.2	90.4	98.3
		3	0.16	0.131	0.367	0.0335	0.0433	76.9	96.9	99.5
		1	0.169	0.141	0.442	0.0293	0.0429	70.8	95.9	99.3
	s_3	2	0.212	0.16	0.491	0.047	0.0764	63.5	90.8	98.4
		3	0.154	0.123	0.355	0.0287	0.0431	79.1	97.2	99.6
			smallest the best					*biggest the best*		

Table 11. Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. We show the versions of the networks in which we use one single sensor size for training. Notice the lack of generalization of the network despite being a FCN. This table shows the results for each fold on each one of the test sets.

proach is naïve *image resizing*. This means we train the network on a single image input size and resize those images that have a different size to make them fit. This, surprisingly, proved to work with a small number of different sensor sizes (see training rows $s_1 s_2{}^\dagger$ of Table 12). However it performs poorly when we augment the number of sensor sizes at training time, see training rows $s_1 s_2 s_3{}^\dagger$ and $s_3 s_1 s_2{}^\dagger$. This make sense, as resizing deforms the

Test	**Train**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
			$:1$	$1/m$	m	$log(m)$	$:1$	$\%$	$\%$	$\%$
s_1	$s_1\ s_2$	1	0.182	0.151	0.514	0.0365	0.0528	67.4	94.2	98.9
		2	0.229	0.18	0.585	0.056	0.0862	57.6	87.7	97.4
		3	0.171	0.133	0.394	0.0332	0.0526	75.1	96.4	99.5
	$s_1\ s_2^\dagger$	1	0.172	0.139	0.456	0.033	0.0462	70.7	95.7	99.4
		2	0.237	0.179	0.574	0.0575	0.0921	56.9	86.8	97.4
		3	0.172	0.135	0.396	0.033	0.0514	74.2	96.4	99.5
	$s_1\ s_2\ s_3$	1	0.178	0.151	0.487	0.0401	0.0514	69.1	94.4	98.8
		2	0.224	0.179	0.531	0.0547	0.0818	59.6	88.2	97.7
		3	0.165	0.14	0.388	0.0355	0.0464	75.1	96.2	99.4
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.191	0.159	0.536	0.0352	0.0531	62.9	93.7	98.9
		2	0.236	0.19	0.613	0.059	0.0912	54.7	86.1	97.1
		3	0.296	0.187	3.08	0.1	1.57	54.6	85.8	95.4
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.203	0.172	0.573	0.0407	0.0589	59.3	92.2	98.4
		2	0.252	0.214	0.671	0.0638	0.0963	49.1	83.3	95.7
		3	0.172	0.138	0.393	0.0349	0.0511	73.9	96.1	99.5
s_2	$s_1\ s_2$	1	0.159	0.127	0.455	0.0321	0.0416	74.4	96.2	99.3
		2	0.209	0.17	0.521	0.051	0.0746	62.3	91.0	98.0
		3	0.161	0.125	0.362	0.0282	0.0476	78.2	97.0	99.7
	$s_1\ s_2^\dagger$	1	0.147	0.116	0.393	0.0279	0.037	78.5	97.4	99.6
		2	0.21	0.165	0.494	0.0476	0.0762	63.1	91.6	98.4
		3	0.163	0.125	0.357	0.0264	0.0457	77.9	97.3	99.7
	$s_1\ s_2\ s_3$	1	0.156	0.128	0.443	0.0345	0.0418	74.9	96.1	99.2
		2	0.21	0.175	0.489	0.0507	0.0757	62.6	90.8	98.1
		3	0.15	0.126	0.345	0.0312	0.0411	79.6	96.9	99.6
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.159	0.13	0.457	0.0303	0.0403	73.6	96.1	99.4
		2	0.203	0.171	0.51	0.0485	0.0708	63.1	91.2	98.2
		3	0.267	0.174	1.71	0.0796	0.483	58.4	88.1	96.6
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.154	0.124	0.466	0.0327	0.0401	75.8	96.2	99.3
		2	0.213	0.179	0.552	0.0549	0.0759	60.5	89.9	97.6
		3	0.168	0.127	0.365	0.0255	0.0504	76.9	97.0	99.7
s_3	$s_1\ s_2$	1	0.165	0.134	0.463	0.0328	0.0436	72.5	95.7	99.2
		2	0.213	0.171	0.543	0.0509	0.0768	61.1	90.2	98.0
		3	0.166	0.128	0.372	0.0302	0.049	76.7	96.9	99.6
	$s_1\ s_2^\dagger$	1	0.149	0.117	0.4	0.0283	0.0374	77.5	97.2	99.6
		2	0.219	0.167	0.525	0.0512	0.0816	61.8	90.4	98.2
		3	0.171	0.129	0.377	0.0285	0.0506	75.4	96.9	99.6
	$s_1\ s_2\ s_3$	1	0.161	0.136	0.445	0.0359	0.0436	74.1	95.6	99.1
		2	0.209	0.17	0.506	0.0503	0.0748	62.8	90.6	98.1
		3	0.155	0.127	0.363	0.0324	0.0424	78.4	96.7	99.5
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.162	0.136	0.462	0.0311	0.042	72.4	95.7	99.3
		2	0.214	0.173	0.549	0.0514	0.0756	60.7	90.3	98.0
		3	0.286	0.178	2.39	0.087	0.912	56.3	86.7	96.2
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.167	0.135	0.494	0.035	0.0446	71.4	95.2	99.0
		2	0.224	0.187	0.602	0.0588	0.0845	58.0	88.1	97.1
		3	0.171	0.13	0.375	0.0295	0.0531	75.0	96.7	99.7
					smallest the best				*biggest the best*	

† the image size has been resized to the first one in the list.

Table 12. Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. We show the versions of the networks in which we use multiple sensor sizes for training. Notice the lack of generalization despite being using FCN. This table shows the results for each fold on each one of the test sets.

images a well as their intrinsic parameters. With only two image sizes ($s_1 s_2^\dagger$) resizing does not create inconsistencies, and the network it is able to learn from it. But, when adding more image sizes, and vertical and horizonal resizing are needed, the problem becomes more complex and may create inconsistencies in the data. Therefore it is an approach that does not scale.

Training on different image sizes (2): Our second approach uses a *Siamese Architecture* with weight sharing during training. This model seems more coherent, as it keeps the original images (no resizing). However, standard FCN networks were not able to learn from several sensor sizes, and their performance was exactly the same as if training in the test sensor size. *E.g.*, see rows $s_1 s_2$ and $s_1 s_2 s_3$ in Table 12. In this case adding more sensor sizes during training $s_1 s_2 s_3$ improves. This suggest that adding more could help. However, stacking infinite neural networks (weight sharing) during training is also not scalable.

Test	Train	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
s_1	s_1	0.189	0.15	**0.46**	**0.037**	**0.0585**	67.6	**94.3**	**99.2**
	s_2	0.206	0.175	0.51	0.0422	0.0621	58.5	92.2	98.7
	s_3	0.193	0.158	0.476	0.0378	0.058	65.4	93.8	99.0
	$s_1\ s_2$	0.191	0.153	0.484	0.0401	0.0606	67.1	93.6	98.9
	$s_1\ s_2^\dagger$	0.188	**0.149**	0.468	0.0391	0.059	68.0	94.2	99.1
	$s_1\ s_2\ s_3$	**0.184**	0.154	0.464	0.0424	0.0571	**68.5**	93.8	98.9
	$s_1\ s_2^\dagger s_3^\dagger$	0.239	0.179	0.742	0.064	0.111	56.9	88.3	97.2
	$s_3\ s_1^\dagger s_2^\dagger$	0.206	0.174	0.53	0.044	0.0654	60.9	91.6	98.4
s_2	s_1	0.197	0.151	0.44	0.038	0.0637	66.7	94.5	99.3
	s_2	**0.166**	**0.133**	0.412	0.0323	0.0468	**74.7**	96.1	99.4
	s_3	0.174	0.138	0.42	0.0334	0.0514	72.5	95.7	99.4
	$s_1\ s_2$	0.173	0.139	0.436	0.0352	0.052	72.5	95.3	99.3
	$s_1\ s_2^\dagger$	0.169	0.134	**0.408**	**0.0318**	**0.0484**	73.9	**96.2**	**99.5**
	$s_1\ s_2\ s_3$	0.168	0.14	0.422	0.0376	0.0498	73.4	95.3	99.2
	$s_1\ s_2^\dagger s_3^\dagger$	0.209	0.16	0.622	0.0514	0.083	63.7	91.7	98.1
	$s_3\ s_1^\dagger s_2^\dagger$	0.174	0.141	0.443	0.0355	0.0521	71.7	95.1	99.2
s_3	s_1	0.184	0.143	0.44	0.0357	0.0574	69.7	94.9	99.3
	s_2	0.177	0.145	0.435	0.0356	0.05	70.6	95.2	99.2
	s_3	0.174	0.14	**0.425**	**0.0336**	**0.0504**	71.9	95.4	99.3
	$s_1\ s_2$	0.178	0.143	0.451	0.0365	0.0537	70.8	94.9	99.2
	$s_1\ s_2^\dagger$	0.175	**0.136**	0.427	0.0342	0.0516	72.3	**95.6**	**99.4**
	$s_1\ s_2\ s_3$	**0.171**	0.143	0.432	0.0383	**0.0504**	**72.5**	95.0	99.1
	$s_1\ s_2^\dagger s_3^\dagger$	0.219	0.164	0.662	0.0552	0.0949	62.1	90.8	97.9
	$s_3\ s_1^\dagger s_2^\dagger$	0.183	0.149	0.473	0.039	0.0577	68.6	94.1	99.0
				smallest the best				*biggest the best*	

† the image size has been resized to the first one in the list.

Table 13. Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. Notice the lack of generalization despite being using FCN. This table shows median results for all the folds on each one of the test sets. This table shows that weight sharing during training is the policy that scale the best to more sensor sizes. All the train and test sets focal lengths have been randomly sample between f_{72} and f_{128}.

We show a summary of all the results, related to sensor size overfitting, in Table 13. Notice how training and testing on the same sensor size outperforms training and testing on the different sensor sizes. It also performs better in most cases compared with training with multiple sensor sizes, even if the test sensor size is included.

B.6. Generalization with CAM-Convs

In this section we show the complete evaluation of CAM-Convs in the 2D-3D Semantics Dataset from Stanford [2].

Improving same-camera baseline: In Table 14 we compare our model with CAM-Convs trained on two different sensor sizes s_1 and s_2 and tested in those two plus an different sensor size s_3. In all cases focal length has been randomly sampled between f_{72} and f_{128}. This experiment shows that our model with CAM-Convs is the best one on the three test sets, proving that CAM-Convs generalize to multiple sensor sizes. Compare with results in Table 13.

Generalizing to a different camera: We sought for a more extreme case of generalization. We tested our trained networks on a complete different camera with sensor size s_5 and focal f_{64}, numbers can be seen in Table 15. In this experiments we show that our model with CAM-Convs is the only one capable of performing at a similar level to the same-camera baseline.

Qualitative results comparing network trained on same data with and without CAM-Convs can be seen in Figure 4.

C. NYU Experiment

In our last experiment we demonstrate how CAM-Convs can generalize across datasests by training on four datasets with different cameras (KITTI [13], ScanNet[3], MegaDepth [9] and Sun3D[15] and testing on a different one (NYUv2 [12]).

C.1. Training

We trained our network for three different sensor sizes (320×320, 256×256 and 224×224) using weight sharing. We did not apply focal length normalization. We augmented the training data by synthetically generating new camera parameters (see Figure 5) for the given training images:

Principal point shifting: In order to move the principal point in the image, we use cropping. We randomly shift the principal point in the original images with a maximum of 50 pixels on each direction.

Random focal length: Every branch of the network has a

Test	**Train**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
		: 1	$1/m$	m	$log(m)$	: 1	%	%	%	
s_1	s_1	1	0.181	0.151	0.475	0.0319	0.0481	66.8	95.2	99.3
		2	0.229	0.169	0.526	0.0533	0.085	59.3	88.5	98.0
		3	0.171	0.132	0.391	0.0317	0.0528	75.0	96.5	99.5
	CAM-Convs s_1 s_2	1	0.173	0.147	0.459	0.0266	0.0433	69.5	96.2	99.4
		2	0.205	0.165	0.503	0.0458	0.0693	62.6	91.3	98.6
		3	0.154	0.12	0.356	0.0261	0.0429	80.0	97.6	99.7
s_2	s_2	1	0.152	0.121	0.409	0.0263	0.037	77.4	97.2	99.5
		2	0.207	0.162	0.493	0.0462	0.0748	64.8	91.7	98.5
		3	0.152	0.123	0.347	0.0293	0.0415	79.6	97.3	99.7
	CAM-Convs s_1 s_2	1	0.159	0.132	0.425	0.0226	0.037	74.8	97.4	99.6
		2	0.185	0.158	0.448	0.0408	0.0589	67.1	93.3	98.9
		3	0.139	0.109	0.321	0.022	0.0357	84.2	98.2	99.7
s_3	s_3	1	0.169	0.141	0.442	0.0293	0.0429	70.8	95.9	99.3
		2	0.212	0.16	0.491	0.047	0.0764	63.5	90.8	98.4
		3	0.154	0.123	0.355	0.0287	0.0431	79.1	97.2	99.6
	CAM-Convs s_1 s_2	1	0.163	0.136	0.419	0.0233	0.0378	73.3	97.0	99.5
		2	0.193	0.161	0.467	0.0414	0.0629	66.4	92.7	98.8
		3	0.145	0.112	0.338	0.0234	0.0386	82.2	98.0	99.7
s_1	s_1	M_e	0.189	0.15	0.46	0.037	0.0585	67.6	94.3	99.2
	CAM-Convs	M_e	**0.175**	**0.144**	**0.433**	**0.0312**	**0.0498**	**71.0**	**95.7**	**99.4**
s_2	s_2	M_e	0.166	0.133	0.412	0.0323	0.0468	74.7	96.1	99.4
	CAM-Convs	M_e	**0.158**	**0.131**	**0.39**	**0.0265**	**0.0417**	**76.5**	**97.0**	**99.5**
s_3	s_3	M_e	0.174	0.14	0.425	0.0336	0.0504	71.9	95.4	99.3
	CAM-Convs	M_e	**0.164**	**0.134**	**0.402**	**0.0283**	**0.0441**	**74.6**	**96.6**	**99.5**
					smallest the best				*biggest the best*	

† the image size has been resized to the first one in the list.

Table 14. .Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. Notice the excelent generalization by including CAM-Convs, compare with Table 13. The upper rows show by-fold results, and the bottom ones show median results for all the folds on each one of the test sets. CAM-Convs outperforms even the **same-camera baseline**, and even when the test sensor size s_3 is not among the training ones. All the train and test sets focal lengths have been randomly sample between f_{72} and f_{128}

fixed sensor size. In order to randomize the focal length we apply a random-size crop (in the center of the image) and resize it to the sensor size of the branch. The crop aspect ratio will be the same as the branch sensor size. Focal length is recalculated afterwards.

C.2. Testing

We test 6 different cameras on the NYUv2 dataset [12]. To create these cameras we apply the same augmentation pipeline as shown in Figure 5. For each of the 6 cameras we apply the corresponding augmentation parameters to the whole test set. Table 16 shows numbers for each camera. We compare our results with Laina [8], which is a state-of-the-art method on this dataset. Our network has not been fine-tuned for any of the 6 cameras. Results show that our method adapts better to different cameras, while for [8] the performance drops significantly when the camera changes (the 3rd camera in the table corresponds to the intrinsics of the training data used in [8]).

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016. 1

[2] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2, 3, 8, 11

[3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 8

[4] C. Godard, O. Mac Aodha, and G. Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018. 1

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[7] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017. 1

[8] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. 1, 9, 10

[9] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

Train	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
		$: 1$	$1/m$	m	$log(m)$	$: 1$	%	%	%
$s_5 f_{64}$	1	0.155	0.224	0.341	0.0303	0.0383	74.1	95.9	99.2
	2	0.203	0.274	0.335	0.0514	0.0742	68.0	91.5	98.3
	3	0.144	0.197	0.242	0.0305	0.0394	81.1	97.0	99.6
$s_1 f_{64}$	1	0.205	0.246	0.316	0.0549	0.0691	63.7	93.2	99.1
	2	0.299	0.353	0.373	0.0756	0.149	48.4	82.1	95.7
	3	0.249	0.294	0.323	0.0524	0.103	57.0	90.3	98.6
$s_1 s_2 \mathcal{U} f_{72} f_{128}{}^\dagger$	1	0.301	0.293	0.4	0.0348	0.127	43.3	91.6	99.1
	2	0.389	0.415	0.467	0.0651	0.217	30.8	75.7	94.6
	3	0.421	0.413	0.46	0.0343	0.231	23.2	77.2	97.3
$s_1 s_2 \mathcal{U} f_{72} f_{128}$	1	0.171	0.227	0.358	0.0481	0.0528	71.3	94.4	98.8
	2	0.239	0.325	0.418	0.0735	0.0971	57.2	86.7	96.4
	3	0.187	0.241	0.267	0.0379	0.0625	72.8	94.9	99.3
$s_1 s_2 s_3 \mathcal{U} f_{72} f_{128}$	1	0.172	0.237	0.34	0.0525	0.0526	71.0	94.1	98.7
	2	0.245	0.346	0.379	0.076	0.0998	55.4	86.1	96.2
	3	0.165	0.236	0.264	0.0418	0.0509	75.3	94.9	99.2
CAM-Convs $s_1 s_2 \mathcal{U} f_{72} f_{128}$	1	0.142	0.2	0.279	0.0307	0.035	77.4	96.8	99.6
	2	0.228	0.298	0.322	0.0549	0.0863	60.5	90.9	98.2
	3	0.175	0.217	0.269	0.0271	0.0544	76.3	96.7	99.7
$s_5 f_{64}$	M_e	0.163	0.227	0.309	0.0356	0.0462	75.1	95.3	99.3
$s_1\ f_{64}$	M_e	0.245	0.292	0.337	0.0598	0.101	56.6	89.2	98.2
$s_1 s_2 \mathcal{U} f_{72} f_{128}{}^\dagger$	M_e	0.369	0.369	0.44	0.0427	0.188	32.0	81.9	97.6
$s_1 s_2 \mathcal{U} f_{72} f_{128}$	M_e	0.196	0.262	0.343	0.0511	0.0688	67.4	92.6	98.6
$s_1 s_2 s_3 \mathcal{U} f_{72} f_{128}$	M_e	0.191	0.269	0.328	0.055	0.0647	67.7	92.5	98.5
CAM-Convs$\mathcal{U} f_{72} f_{128}$	M_e	0.177	0.236	**0.289**	0.0362	0.0541	71.9	**95.4**	**99.4**
		smallest the best					*biggest the best*		

† the image size has been resized to the first one in the list.
* the network has been trained without focal length normalization.

Table 15. In order to test the generalization capabilities of the networks we have tested our best performing networks on a different camera model. Upper part shows by-fold results, bottom part shows median results for all the folds on each one of the test sets. CAM-Convs proved to achieve close to the **same-camera baseline** performance while none of the other models show good performance. The test has been done with s_5 images taken with f_{64}.

PP Shift	Crop	Resize	Model	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
(w, h)	$W \times H$	$W \times H$		$: 1$	$1/m$	m	$log(m)$	$: 1$	%	%	%
-	-	-	[8]	0.43	0.323	1.34	0.0518	0.202	5.4	24.8	64.8
			ours	**0.228**	**0.114**	**0.732**	**0.0496**	**0.076**	**52.0**	**87.9**	**98.2**
-	-	256×192	[8]	0.298	0.107	0.784	**0.041**	0.157	53.0	89.1	97.1
			ours	**0.234**	**0.101**	**0.681**	0.0441	**0.0852**	**56.3**	**90.5**	**99.4**
-	640×448	320×224	[8]	**0.175**	**0.0743**	**0.533**	**0.0392**	**0.0626**	**76.4**	**94.6**	98.9
			ours	0.211	0.0911	0.657	0.0425	0.0687	62.3	92.3	**99.5**
-	380×380	256×256	[8]	0.245	0.125	0.826	**0.0328**	0.0766	36.7	89.9	98.7
			ours	**0.196**	**0.0854**	**0.658**	0.0334	**0.058**	**62.8**	**93.0**	**99.6**
$(40, -50)$	352×352	256×256	[8]	0.279	0.156	0.902	0.0317	0.0919	23.9	82.9	98.1
			ours	**0.194**	**0.0864**	**0.64**	**0.0303**	**0.0552**	**63.3**	**93.7**	**99.7**
$(-20, 15)$	186×465	128×320	[8]	0.285	0.158	0.948	0.0281	0.0957	22.7	79.4	98.3
			ours	**0.197**	**0.0883**	**0.668**	**0.0245**	**0.0556**	**62.0**	**94.8**	**99.9**

Table 16. Experiments on NYUv2 [12]. We compare our model on six different cameras (see Figure 5 for details on how to generate images from different cameras) against Laina [8]. Notice how the errors of our network are almost constant despite changing the camera parameters. The performance of [8] degrades as the test camera parameters move away from the training camera parameters. The first test (first two rows) was done with the original image size of the dataset, which is 640×480.

[10] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 1

[11] O. Ronneberger, P. Fischer, and T. 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. 1

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 8, 9, 10

[13] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity Invariant CNNs. In *International Conference on 3D Vision (3DV)*, 2017. 8

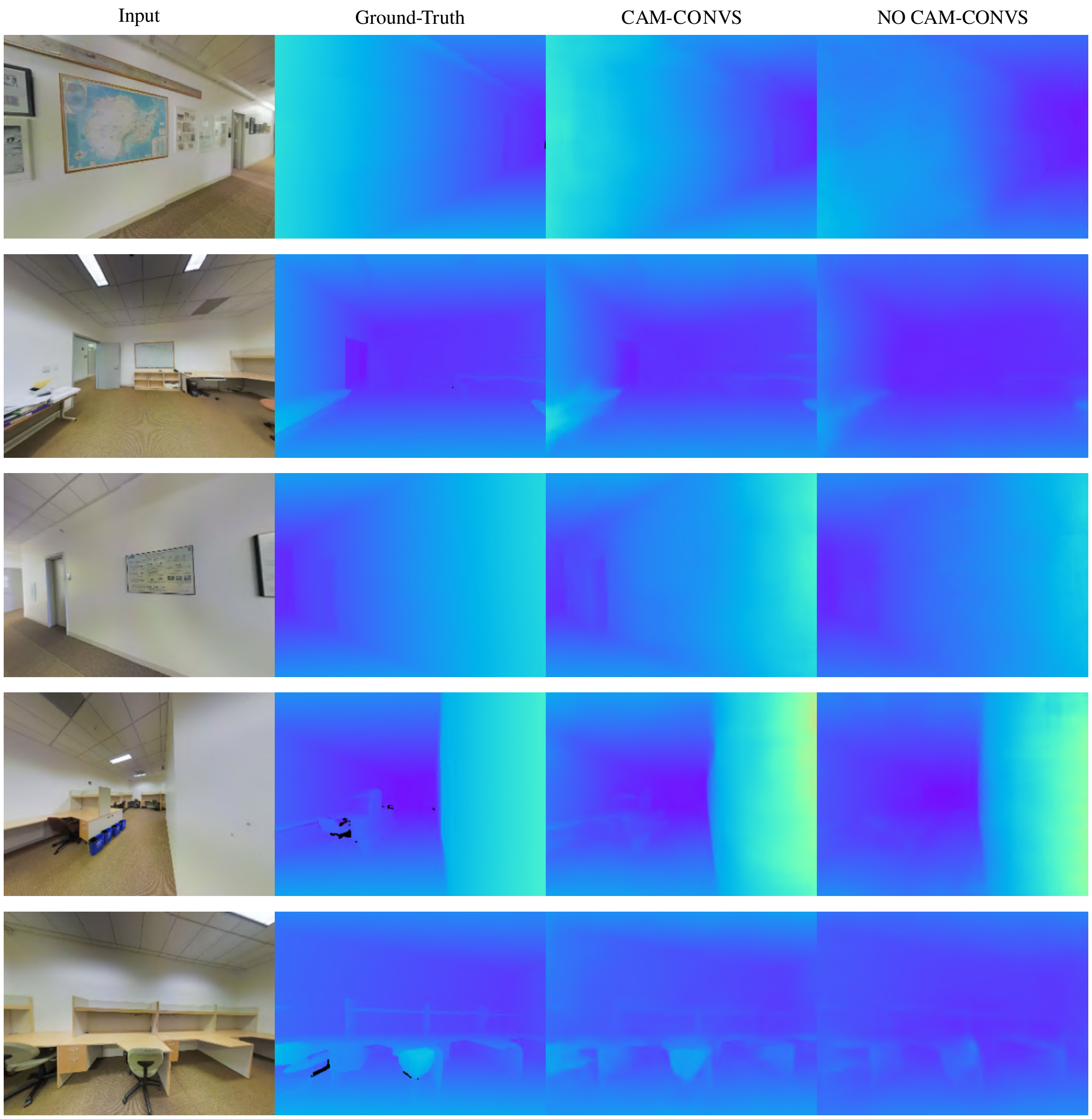

Figure 4. Qualitative results on 2D-3D Semantincs Stanford Dataset [2]. Sensor size 320×256 and focal length f_{64}. Networks have been trained on 256×192 and 192×256 with focal lengths between f_{72} and f_{128}.

[14] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and Motion Network for learning monocular stereo. In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, page 6, 2017. 1

[15] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013. 8

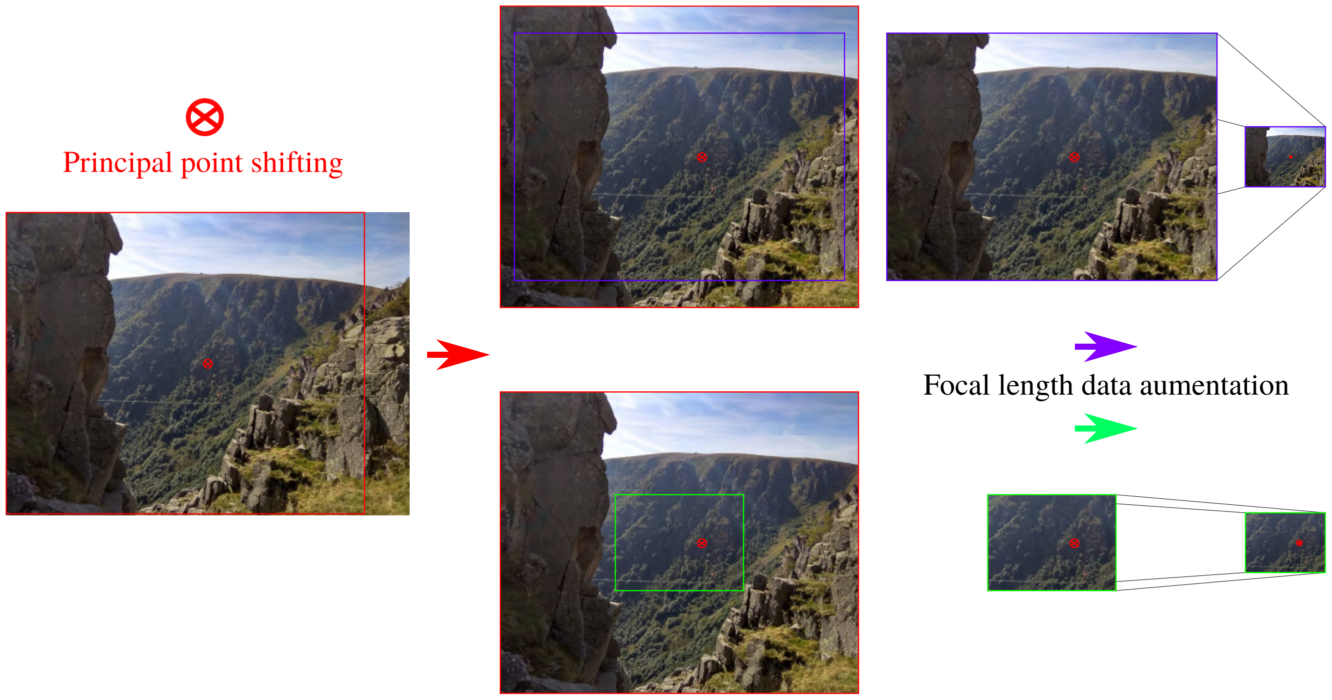

Figure 5. Overview of the steps for camera data augmentation.