FoVA-Depth: Field-of-View Agnostic Depth Estimation for Cross-Dataset Generalization

Supplementary Material

1. Epipolar Geometry

Figure 1 shows how epipolar geometry plays out for different GCCs. In particular, there is an *epipolar plane* that contains the camera centers and the query ray. The intersection of this plane with the source sphere makes an *epipolar great circle* on the source sphere corresponding to the possible matches of the query ray. The path projects, via ϕ^{-1} to different *epipolar curves* depending on the camera model.

Note that there are two epipoles corresponding to the two intersections of the line passing through the camera centers with the reference camera. In the pinhole case, due to the FoV being strictly less than 180°, at most one epipole is observed. In the rectified case neither epipole is observed, but in some sense one epipole is infinitely to the left and the other is infinitely to the right.

2. Interpolation

In this section, we elaborate on the necessity of property 3 and explain why general samplings of the sphere can be problematic. When we create the cost-volume, we have to warp the source images to the reference images. This process requires interpolating the source images. To perform interpolation, we need to conduct a k-nearest-neighbor search in the canonical representation. For example, for a planar image like the Equirectangular Projection (ERP), finding the nearest neighbor is essentially free. We just need to take the floor and ceiling of each pixel coordinate to obtain the four nearest neighbors for interpolation. The process for a cube is slightly more complex, since we first have to determine which face of the cube the point is on, i.e., find the coordinate with a value equal to ± 1. Then, it reduces to the flat plane case.

If we have more exotic samples of the sphere, we must perform a full nearest neighbor search of the points, which is prohibitively expensive for network training (at least with existing implementations). Komatsu *et al*. actually conduct this nearest-neighbors search. They can do so because they use a fixed camera rig, meaning the relative pose between the source and reference image is fixed. This allows them to perform the nearest-neighbors search once for the entire training period [2].

3. Reciprocal Tanget Sampling

In this section we motive our reciprocal tangent sampling strategy for generating hypotheses. First, consider the rectified stereo case. The disparity is related to depth by disparity $\sim 1/\text{depth}$. This implies uniformly sampling inverse depth hypotheses corresponds to sampling equidistant points in the source image. We would like to do something similar for spherical images *i.e*. sample distance hypotheses that correspond to equidistant samples on the sphere. To accomplish this, we need to establish the relation between the distance and the sampling locations on the source sphere *i.e*. the angular disparity.

Consider the schematic drawing of an epipolar plane for two spherical cameras shown in Figure 2. Call the vector going from the source to the reference camera the baseline vector, with length b. Consider a query ray extending from the reference camera center that hits a 3D point P a distance d away. This ray makes an angle θ with the baseline vector. Define the angle ϕ as the angle between the baseline vector and the ray from the source camera center to P. Given θ, d and ϕ are related, though conventionally, rather than writing this relation explicitly, we write the relation between d and the angular disparity, α, defined as $\alpha = \theta - \phi$. This relation is given by:

$$d(\alpha, \theta) = b\left(\frac{\sin\theta}{\tan\alpha} - \cos\theta\right) \tag{1}$$

Where $\theta \in [0, 180]$ and $\alpha \in [0, \theta)$ [6]. Note that unlike the rectified pinhole stereo case, the relation between distance and disparity depends on the reference pixel that determines θ. For two cameras one can choose different hypotheses per pixel, but for more than two, this is a problem because there are different θs and αs associated with each source camera. However, from Equation 1 we observe the rough relation between disparity and distance follows a shifted and scaled inverse tangent function. This is the motivation for reciprocal tangent sampling.

Qualitatively, reciprocal tangent sampling samples points less densely near to the camera than inverse distance sampling. This fit our intuition that inverse distance sampling was wasting samples very close to the camera. An additional benefit of reciprocal tangent sampling is that it can handle arbitrarily large depths with finite samples, similarly to inverse distance and as opposed to linear sampling. This enables us to handle both indoor and outdoor scenes without any changes to the sampling strategy.

We do not claim this sampling is optimal in all cases, just that it worked well for both indoor and outdoor scenes we studied.

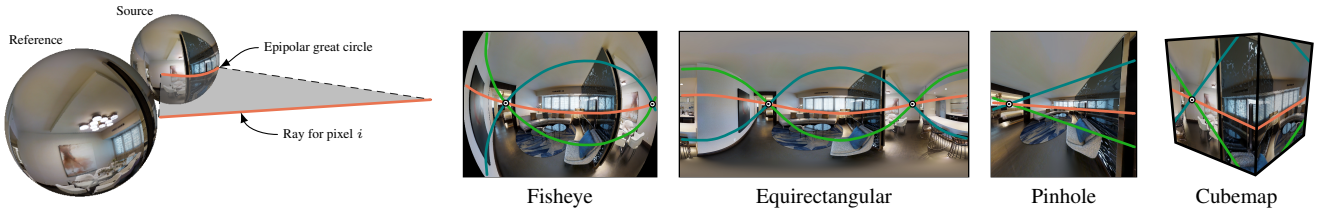

Figure 1. Epipolar Geometry for GCCs. The *epipolar plane*, in gray, contains the spherical camera centers and the query ray. We call the great circle where this plane intersects the source sphere the *epipolar great circle*. On the right, we show how the epipolar great circle manifests as different types of *epipolar curves* in image space.

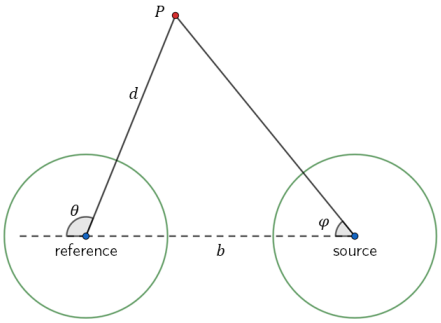

Figure 2. Schematic drawing of the epipolar plane, gray plane in Figure 1, and the angles that relate distance, d, to angular disparity $\alpha = \theta - \phi$. Given baseline b.

4. ERP Rectification for MODE

The input to MODE [3] must be two ERP images rectified such that the cameras are directly "on top of each other" that is the camera coordinate up axis (y-axis) is parallel to the cameras' translation, and both cameras have the same extrinsic rotation. The idea is to apply Equation 1 to simulate this configuration. Concretely, our inputs are two images with GCCs (U^i, ϕ^i) and extrinsics (R^i, t^i) with $i \in 0, 1$. We write R^i as the concatenation of three column vectors $R^i = \left[r_0^i, r_1^i, r_2^i\right]$. Then mathematically to be "on top of each other" means $r_1^0 = r_1^1 = \frac{t^1 - t^0}{||t^1 - t^0||}$ and $R^0 = R^1$.

To rectify the images define $\hat{r}_1 = \frac{t^1 - t^0}{||t^1 - t^0||}$. Now we can choose arbitrarily $\hat{r}_2$ any vector orthogonal to $\hat{r}_1$. Define $\hat{r}_0 = \hat{r}_1 \times \hat{r}_2$, where $\times$ is the cross product of vectors. Let $\hat{R} = [\hat{r}_0, \hat{r}_1, \hat{r}_2]$. Now we warp each image using Equation 2 with $R' = \hat{R}$, (U', ϕ') the ERP GCC, $(U, \phi) = (U^i, \phi^i)$ and $R = R^i$:

$$I'(u) = I(\phi^{-1}(R^{-1}R'\phi'(u)). \tag{2}$$

These warped images are now rectified. An example of two ERP images before and after rectification are shown in Figure 3.

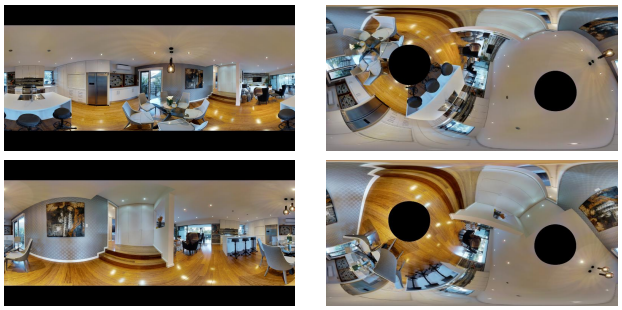

Figure 3. Original ERP images (left) and rectified images for MODE (right)

5. Fisheye ϕ

There are many models of fisheye cameras. Usually these models model the projection function *i.e.* ϕ^{-1}. The model used for KITTI360 has 7 parameters. $f_1, f_2, u_0, v_0, x_i, k_1, k_2$ and is given by $\phi^{-1}(x, y, z) = (u'', v'')$

$$\begin{aligned}(u, v) &= \left(\frac{x}{z + x_i}, \frac{y}{z + x_i}\right) \\ r^2 &= u^2 + v^2 \\ (u', v') &= (1 + k_1 r^2 + k_2 r^4)(u, v) \\ (u'', v'') &= (f_1 u' + u_0, f_2 v' + v_0)\end{aligned}$$

where $(x, y, z) \in \mathbb{S}^2$. To compute ϕ we use iterative undistortion from ϕ^{-1} [4, 5]. Other models of fisheye cameras can be found in [4, 5].

6. CubePadding Seams

Figure 4 shows our cube model with and without cube padding. In particular, we observe strong seams on the edges of the cube faces of the model without padding. In the first row we see a large error on the textureless ceiling. This is because context from surrounding cube faces is needed to deduce the depth of the top face, but without cube padding the network has no context from surrounding faces.

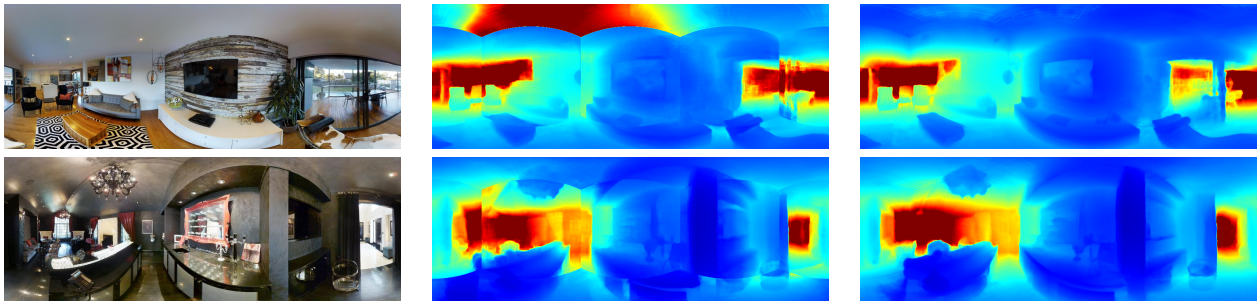

Figure 4. Qualitative comparison of our cubemap model with and without cube padding.

```
from torchvision.models.feature_extraction import create_feature_extractor
from torchvision.models import resnet34

m = torchvision.models.resnet34()
return_nodes = {
    'layer1': 'layer1',
    'layer2': 'layer2',
    'layer3': 'layer3'
}
net = create_feature_extractor(m, return_nodes=return_nodes)
features = net(image)
f1, f2, f3 = features['layer1'], features['layer2'], features['layer3']
```

Figure 5. Definition of "first three layers" of ResNet34

7. Implementation Details

7.1. Our Models

Feature Extractor For the architecture of the feature extractor, we use the first three layers of Torchvision's implementation of ResNet34 to extract three feature maps f1, f2, and f3 at resolutions $1/4$, $1/8$, and $1/16$ of the input resolution with 64, 128, and 256 channels, respectively. See Figure 5 for more details. We pass f1 though a transposed conv layer with stride 1 and 32 features, f2 though a transposed conv layer with stride 2 and 32 features, and f3 through a transposed conv layer with stride 4 and 64 features. We then concatenate the outputs to form a single feature map at $1/4$ input resolution with 128 channels.

All convolutions are replaced with either CubeConv or CircConv depending on the respective model.

Cost Regularization The cost regularization network is based on the network from MVSNet [7]. Besides replacing all convs with CubeConv3d or CircConv3d we replace all transposed convs with nearest-neighbor upsampling followed by a conv with the same number of features as the original transposed conv. We do this to avoid the need for output padding in transposed convs in order to upsample by an exact multiple of 2.

7.2. 360MVSNet

For a fair comparison, we used the same reciprocal tangent sampling to select initial distance hypotheses as we used for our method when training 360MVSNet. For subsequent stages we used the uncertainty aware sampling proposed by the authors. We considered two feature extractors: The FCN-model, which comes from casMVSNet [1] that 360MVSNet is based on and an upgraded one based on ResNet34. The multiscale features are f1, f2, f3 as in Figure 5. Each is passed though a stride 1 conv layer to reduce it to 8, 16, and 32 channels respectively.

References

[1] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[2] Ren Komatsu, Hiromitsu Fujii, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. 360° depth estimation from multiple fisheye images with origami crown representation of icosahedron. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020. 1

[3] Ming Li, Xueqian Jin, Xuejiao Hu, Jingzhao Dai, Sidan Du, and Yang Li. MODE: Multi-view omnidirectional depth estimation with 360° cameras. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[4] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[5] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[6] Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360SD-Net: 360° stereo depth estimation with learnable cost volume. *arXiv preprint arXiv:1911.04460*, 2019. 1

[7] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2018. 3