Solution of 5th place

Please respond to the following questions, including visualizations where relevant. Responses should be 4-8 pages including text, figures, tables, and references.

1. ***Provide a brief abstract of your final approach (one paragraph), highlighting the most important or interesting ideas and impactful decisions made.***

In this challenge, we argue that the most challenging problem lies in monocular height perdition, which is caused by height variance in multi-city, radiation variance of images, and scale variance of different objects. To alleviate these problems, we first conducted an empirical upper bound analysis to locate the concrete bottlenecks of the baseline model [2]. Motivated by these insights, we designed a high-resolution vector flow network, called HR-VFLOW, and adopt a divide-and-conquer training strategy to train this model. Overall, our final model won 5th place with an R^2 of 0.8556 using single-model entry, which outperforms the baseline model by 5.6%. Our main contributions are as follows:

- To understand the main bottlenecks of the baseline model, we conduct an empirical upper bound analysis, which suggests that the main errors are from height prediction and the rest are from angle prediction. The errors of scale prediction are very small.
- To overcome the bottlenecks, we proposed HR-VFLOW, which takes HRNet [3] as backbone and adopts simple multi-scale fusion as multi-task decoders to predict height, magnitude, angle, and scale simultaneously.
- Considering the height variance in the different cities, we adopt a divide-and-conquer training strategy to handle the height variance. We first pretrained our model on the whole four cities training set and transfer the pretrained model on the specific city for better city-wise performance.

More details are summarized below.

2. ***Explain your technical approach in detail. Discuss what you did and why. Discuss what worked well and not so well in your final solution. It is not necessary to provide background material on the challenge formulation unless it helps explain your solution.***

2.1 Our approach

2.1.1 Upper bound analysis

Geocentric pose is defined as $g(\boldsymbol{I}) = \{s, \theta, \boldsymbol{h}\}$ for the image $\boldsymbol{I}$. There are three factors to impact the final performance. To investigate the importance of these three factors, we first conduct an empirical upper bound analysis. The baseline model is ResNet-34-based UNet [2]. We use a subset of training data to conduct this study.

The main results are listed in Table. 1. We gradually replace the prediction with the corresponding ground truth to obtain upper bound of the accuracy. Considering a single factor, we can find that height prediction is the biggest error source. By replacing the height prediction, the overall R^2 is improved from 0.569 to 0.9724. The VFlow R^2 is improved from 0.6077 to 0.9449. The rest of VFlow errors lies

in angle prediction. There is a strange point that replacing the scale prediction, the upper bound decreases. This is because scale is only a coefficient to convert height to magnitude in an affine

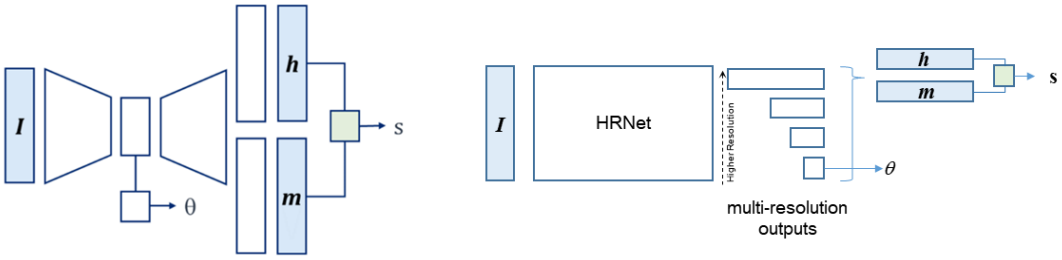
Tab. 1: Upper bound analysis for the baseline solution model [2]. The experiments were conduct on US3D-OMA dataset. gt scale s, gt angle θ, and gt height $\boldsymbol{h}$ denote whether to use the ground truth scale, angle, and height to replace the corresponding predictions, respectively.

gt scale **s**	*gt angle* θ	*gt height* **h**	R^2			RMSE				MAE			
			Overall	Height	VFlow	Angle	Mag	EPE	Height	Angle	Mag	EPE	Height
✗	✗	✗	0.5690	0.5303	0.6077	12.05	3.04	3.15	6.01	10.02	1.03	1.14	2.08
✓	✗	✗	0.5672	0.5303	0.6042	12.05	3.06	3.16	6.01	10.02	1.01	1.12	2.08
✗	✓	✗	0.5816	0.5303	0.6330	0.	3.04	3.15	6.01	0.	1.03	1.14	2.08
✗	✗	✓	0.9724	1.	0.9449	12.05	0.48	1.18	0.	10.02	0.15	0.40	0.
✓	✓	✗	0.5795	0.5303	0.6288	0.	3.06	3.06	6.01	0.	1.01	1.01	2.08
✓	✗	✓	0.9777	1.	0.9555	12.05	0	0.33	0.	10.02	0	1.06	0.
✗	✓	✓	0.9953	1.	0.9907	0.	0.48	0.48	0.	0.	0.15	0.15	0.

camera model. If the accuracy of height prediction is not guaranteed, the performance magnitute is hard to obtain postitive improvement even if replacing scale prediction by ground truth. To joint consider two factors, we can find that replacing both angle and height prediction, the empirical upper bound is very close to the theoretical upper bound. Therefore, the main bottlenecks of the baseline model are height and angle prediction, which are the keys to further improve the model performance.

2.1.2 High-Resolution Vector Flow Network (HR-VFLOW)

To achieve robust height and angle regression, we proposed a high-resolution vector flow network (HR-VFLOW), as shown in Fig. 1 (b). Compared with the baseline model (Fig. 1 (a)), we adopt three parallel decoders to predict angle, height, and magnitute.



(a) baseline (b) ours: HR-VFLOW

Fig. 1: Overview of the baseline model and the proposed HR-VFLOW. (a) baseline model is a Unet-based model with multi-task outputs, where angle is predicted from the encoder part ,and height and magnitute are predirected from the decoder part. (b) HR-VFLOW is a HRNet-based model also with multi-task outputs, where the angle, height, magnitute both are predicted from decoder part.

We first use HRNet-32 to extract multi-resolution feature maps. These feature maps have output strides of 4, 8, 16, 32, respectively. For angle prediction, similar to the baseline model, we append a gobal average pooling layer and a fully-connected layer to regress x and y directions $(\cos\theta, \sin\theta)$ rather than directly regressing the angle value. For height and magnitude prediction, due to its dense prediction task, we first fuse these four multi-resolution features by upsampling these feature maps to

the same output stride of 4 and then concatenate them followed by a 3x3 conv layer. Based on this fused feature map, a 3x3 conv layer with 2 filters is adopt to regress the height and magnitude map. Finally, the scale is estimated by a custom least squre solver layer used in [2].

The loss function is weighted mean square error (MSE) loss, which is the same as the baseline model. The model was trained for 242 epochs using the AdamW optimizer with a total batch size of 8. We used an initial learning rate of 2e-4, which is then divided by 10 at 2/3 of total iterations and again at 8/9 of total iterations. For training data augmentation, random flip, rotation, scale, height, and color jitter were used. In this challenge, we find the color jitter is very important to overcome the radiation variance of images. Other settings are aligned with the baseline model.

2.1.3 Divide-and-Conquer Training Strategy

Different cities have specific height distributions, which is illustrated in [2]. Therefore, we adopt a pretrain and fine-tune following the idea of divide-and-conquer to handle each city. We first trained a HR-VFLOW model on the whole training set to obtain a general and robust model. And then we fine-tune this model on each city. However, we found that only fine-tuning on ATL can be obtained considerable improvement.

2.2 Discussion

2.2.1 Heavy encoder is not always worked.

Before we designed HR-VFLOW, we first try to use different heavy encoders in the baseline model. Table.1 lists the results of using heavy encoders. These results suggest that using too heavy encoder will lead to overfitting. To obtain a trade-off between speed and accuracy, we finally choose the HR-VFLOW as the basic model for next improvement.

Tab. 1: Comparison of basic architecture

architecture	*encoder*	*public score*
VFLOW [2]	ResNet-34	0.8119
VFLOW [2]	ResNet-101	0.8016
VFLOW [2]	EfficientNet-B0	0.8036
VFLOW [2]	EfficientNet-B3	0.8217
VFLOW [2]	EfficientNet-B5	0.8088
HR-VFLOW	HRNet-32	0.8224

2.2.2 MSE loss is robust

Considering the outliers in the ground truth AGL, we try to use the L1 loss to optimize the model. But its convergence speed is too slow to obtain compromising result, as shown in Table. 2.

Tab. 2: Comparison of L1 loss and MSE loss

loss	*public score*
L1	0.7314
MSE	0.8119

2.2.3 Color jitter and multi-step learning rate decay is worked.

Table. 3 shows the incremental improvement by color jitter and multi-step learning rate decay. Although AdamW optimizer has adaptive learning rate, the suitable learning rate decay is still worked well. And color jitter as training data augmentation is worked well to alleviate radiation variance problem.

Tab. 3: Incremental improvements

method	*public score*
HR-VFLOW	0.8224
+ multi-step learning rate decay	0.8438
+ color jitter	0.8548
+ fine-tuning on ATL	0.8568

3. Copy and paste the 3 most impactful parts of your code. Explain what each does and how it helped your model.

The following code implements our simple multi-scale fusion.

```
class SimpleFusion(nn.Module):
    def __init__(self, in_channels):
        super(SimpleFusion, self).__init__()
        self.fuse_conv = nn.Sequential(
            nn.Conv2d(in_channels, in_channels, 1),
            nn.BatchNorm2d(in_channels),
            nn.ReLU(True)
        )

    def forward(self, feat_list):
        x0 = feat_list[0]
        x0_h, x0_w = x0.size(2), x0.size(3)
        x1 = F.interpolate(feat_list[1], size=(x0_h, x0_w), mode='bilinear', align_corners=True)
        x2 = F.interpolate(feat_list[2], size=(x0_h, x0_w), mode='bilinear', align_corners=True)
        x3 = F.interpolate(feat_list[3], size=(x0_h, x0_w), mode='bilinear', align_corners=True)
        x = torch.cat([x0, x1, x2, x3], dim=1)
        x = self.fuse_conv(x)
        return x
```

The following applies color jitter to improve generalization. Details are provided in the code and implemented using Cython for speed. See Fig. 2 and discussion above.

```
extra_aug=A.Compose([
    A.RandomBrightnessContrast(p=0.2),
    A.RandomGamma(p=0.2),
    A.RGBShift(p=0.2)
]),
aug_params=dict(
    rotate_prob=0.3,
    flip_prob=0.3,
    scale_prob=0.3,
    agl_prob=0.3,
)
```

The following code applies multi-step learning decay to improve generalization.

```
learning_rate=dict(
    type='multistep',
    params=dict(
        steps=[60000, 80000],
        gamma=0.1,
        base_lr=0.0002,
    )
),
```

4. How did you evaluate performance of the model other than the provided metric, if at all?

Local endpoint error (EPE), mean absolute error (MAE), root mean squared error (RMSE) are used to monitoring the performance of the model during training.

5. Provide the machine specifications and time you used to run your model.

- CPU (model): Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz

- GPU (model or N/A): NVIDIA Titan RTX
- Memory (GB): 256 GB VRAM
- OS: Centos 7
- Train duration: 46 hours for 242 epochs with 2x down-sampled images
- Inference duration: 20 min

6. What are some other things you tried that didn't make it into your final solution?

About the encoder, we also tried vision transformer as encoder, such as Swin Transformer [4]. But it does not work well, only obtaining a public score of 0.8468.
Multi-model ensemble is also tried but does not work well.
We also tried to use pseudo-label on test set but the accuracy decreases.

7. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Is there other data or metadata that you felt would have been very helpful to have? Would you frame the problem differently?

Obviously, as our upper bound analysis suggests, a more robust monocular height prediction framework should be the most important point to study. Combining our experimental results, different network architecture have varying performance. Therefore, finding a better network architecture via network architecture search (NAS) might be a good choice.

8. Provide any references cited in your discussion above.

[1] G. Christie, R. R. R. M. Abujder, K. Foster, S. Hagstrom, G. D. Hager, and M. Z. Brown, "Learning Geocentric Object Pose in Oblique Monocular Images," in CVPR, 2020. [Link]

[2] G. Christie, K. Foster, S. Hagstrom, G. D. Hager, and M. Z. Brown, "Single View Geocentric Pose in the Wild," in CVPRW, 2021. [Link]

[3] Wang J, Sun K, Cheng T, et al. Deep high-resolution representation learning for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2020. [Link]

[4] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[J]. arXiv preprint arXiv:2103.14030, 2021. [Link]