Supplementary Material : Learning Omnidirectional Flow in 360° Video via Siamese Representation

Keshav Bhandari[1], Bin Duan[2], Gaowen Liu[3], Hugo Latapie[3], Ziliang Zong[1], and Yan Yan[2]

[1]Texas State University, [2]Illinois Institute of Technology, [3]Cisco Research

Here we present some of the sections left for discussion in the main paper. This supplementary report includes a discussion on Flow Generator, Distortion Density Map, Experimental Setting, and Dataset Samples to illustrate additional motion and scene diversity.

1 Flow Generator

FLOW360 is created using Blender[1], free and open source 3D creation suite. Blender provides an interface to write add-ons for automating workflows. We create Flow-generator, an add-on written for Blender-v2.92 to collect optical flow and several other data like depth information and normal maps. Flow-generator can be installed using following steps:

1. Download Flow-generator[2]
2. In Blender-v2.92, follow:
   - Edit → Preferences → Add-ons → Install
   - Select flow_generator.py from Flow-generator project folder
   - Install Add-on
   - Search FlowGenerator
   - Check mark for installation
3. Setup compositor pipeline in Blender-v2.92
   - Go to Compositing
   - Select "SetFlow Generator" from Custom Node Group
   - If Custom Node Group is disabled press "n" to make it visible
   - Select desired configuration and click "SetEnv"
4. Camera Setup
   - Go to Layout
   - Open tab on right side of the Layout view by pressing "n" if not visible
   - Go to "CameraSetup" on the right tab
   - Select desired configuration and click "Set Camera System"

Note that, we use cloud rendering to speed up the rendering process which requires crowd-render[3] add-on. Flow-generator provides two basic functionality: (i) camera setup and (ii) compositor pipeline.

[1] https://www.blender.org/
[2] https://www.dropbox.com/s/v2mvjvs7ze8rzj1/flowgenerator.tar.gz?dl=0
[3] https://www.crowd-render.com/

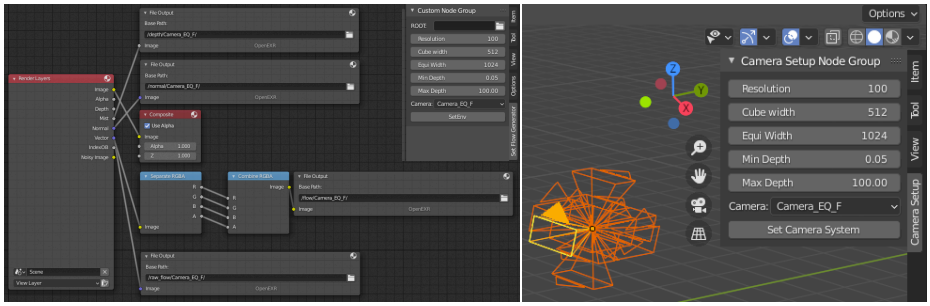

Fig. 1. FlowGenerator. Flow-generator serves two major purposes: (Left) Setting up a compositor pipeline which invovles setting up filepaths, pipelining desired output and setting initial configurations like render-engine and render passes. (Right) Setting up camera configurations and additional information like resolution, dimensions and depth of the scene.

1.1 Compositor Pipeline

Compositor in Blender provides a pipeline to process render output. Flow-generator creates a basic pipeline (shown in Fig. 1) to collect information like optical flow, images, depth maps and so on. It also provides an easy way to cache the desired outputs in structured format. Similarly, it also setups basic configurations like selecting render-engine and render passes.

1.2 Camera Setup

Camera Setup (shown in Fig. 1) can be accessed via 3D-Layout tab in Blender-v2.92 as suggested above. Camera Setup creates an omnidirectional camera with full 360° field of view. The camera setup consists of twelve different cameras (six perspective and six 360°) out of which any omnidirectional camera (default Camera_EQ_F) can be selected as main camera for rendering omnidirectional videos. It also provides an interface to adjust configurations like resolution, dimension and depth of the scene.

2 Distortion Density Map

We compute distortion mask (U_d,D_d,F_d,B_d,R_d,L_d) in a cube-map with six faces: Up(U), Down(D), Front(F), Back(B), Right(R) and Left(L) respectively. This distortion density cube-map projection is then projected to equirectangular projection using spherical co-ordinate transformation (shown in Eq. 1).

$$(x_s, y_s, z_s) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \tag{1}$$

To compute density mask (C_d, where C∈(U,D,F,B,R,L)) in each face, we define a meshgrid for co-oridnates x and y ranging from ($[-1, 1]$) with dimension

of $(256, 256)$. The co-ordinates (x, y) are used to compute a radius map (r) of size $(256, 256)$ as shown in Eq. 2.

$$r = \sqrt{x^2 + y^2} \tag{2}$$

In our paper we have shown that the radial distortion is higher towards central in the polar region which corresponds to (U,D) faces of cube-map projections. Similarly, in equatorial region i.e., rest of the faces (F,R,B,L) shows higher distortion rate away from the center. We compute two distinct distortion map (C_d), one for polar regions (U,D) and another for equatorial regions (F,B,R,L) as shown in Eq. 3

$$C_d = \begin{cases} 1 - r/max(r) & \text{if, C} \in \{\text{U,D}\}\ , \\ r/max(r) & \text{otherwise} \end{cases} \tag{3}$$

Please refer to code[4] for additional details.

3 Experimental Setting

We conduct our experiment in Pytorch (1.9.0+cuda10.2) using latest version of Python (3.8.10). Additional environment detail is provided in project[5].

Following are the list of configurations we used for our project:

- Train/Val Batch Size: Ours(16/12 - 8Gpus), Finetune(8/12 - 4 Gpus)
- Number of iterations on RAFT: 12
- Loss: CosineSimilarity, Optical Flow L1-loss
- Optimizer: AdamW
- Scheduler: OneCycleLR
- EarlyStopping: Patience (5) and Min-delta (1e-4)
- Others: Gradient Clipping, GradScaler
- Dataset: FLOW360[6]

We suggest our readers to refer sample videos[7,8] for demo purpose.

4 Impact of Rotational Invariance

We perform additional experiments to quantify the impact of rotational invariance (see table below, Table.1). The improvement is noticeable as seen in LiteFLowNet360 [1] and Tangent Images [2]-based optical flow estimation with rotational invariance.

[4] https://www.dropbox.com/s/q1d4eoqvj2a30ij/distortion_weight.ipynb?dl=0
[5] https://www.dropbox.com/s/a6qioejg6yrxo7s/SLOF.tar.gz?dl=0
[6] https://www.dropbox.com/s/nvzhazq99bg46f2/FLOW360_train_test.zip?dl=0. Note that for better visualization please clip optical flow in the range of (-40,40) or lower.
[7] https://www.dropbox.com/s/54mmjvoz6844mci/trailer.mp4?dl=0
[8] https://www.dropbox.com/s/gvihrzj528d92uj/videos.zip?dl=0 We recommend to use VLC-Media player to play these videos.

Table 1. Impact of rotational invariance

Method	w/o Rot.Inv (EPE)	w Rot.Inv (EPE)
LiteFLowNet360 [1]	3.95	**2.52**
Tangent Images [2]	3.57	**1.78**

Fig. 2. More Motion and Scene Diversity - Train Set. Illustrating motion and scene diversity via randomly sampled tangential plane from FLOW360 dataset. Flow360 contains range of scene complexity governing varied properties like texture, illumination, human, building, cars and other 3D assets. Similarly, the various level of motion complexity can be seen for similar scenes ranging from smaller to larger displacement. As explained in the main paper the dataset also contains other complexities like motion blur, camera focus/defocus, camera distortion and environmental effects.

Fig. 3. More Motion and Scene Diversity - Test Set. Illustrating motion and scene diversity via randomly sampled tangential plane from FLOW360 dataset. Flow360 contains range of scene complexity governing varied properties like texture, illumination, human, building, cars and other 3D assets. Similarly, the various level of motion complexity can be seen for similar scenes ranging from smaller to larger displacement. As explained in the main paper the dataset also contains other complexities like motion blur, camera focus/defocus, camera distortion and environmental effects.

References

1. Bhandari, K., Zong, Z., Yan, Y.: Revisiting optical flow estimation in 360 videos. In: ICPR (2021) 3, 4
2. Eder, M., Shvets, M., Lim, J., Frahm, J.M.: Tangent images for mitigating spherical distortion. In: CVPR (2020) 3, 4