# Supplementary Materials for ECCV Submission 360° Optical Flow Estimation Based on Multi-Projection Fusion

Yiheng Li[1], Connelly Barnes[2], Kun Huang[1], and Fang-Lue Zhang[1,*]

[1] School of Engineering and Computer Science, Victoria University of Wellington
{Yiheng.Li,Kun.Huang,Fanglue.Zhang}@vuw.ac.nz
[2] Adobe Research, Seattle, US
ConnellyBarnes@gmail.com

## OVERVIEW

Our supplementary materials accompanies the main paper, which presents more details on optical flow dataset generation, visualization of our projections, more experiments on our network structure, and more visualized 360° optical flow estimation results. There is also an extra video showing the result of our video editing application.

## 1 More Details of Optical Flow Generation in Unity3D

The ground truth optical flow can be generated by calculating the corresponding screen pixel position between two frames. In Unity3D, a built-in method named Motion Vector can provide pixel-wise optical flows for planar images. The Motion Vector represents the optical flow from the previous frame $I_{t,f}$ to the current frame $I_{s,f}$. Here, we use $s$ and $t$ to represent the source frame and the target frame, and the $f$ represents each face. We first calculate the corresponding pixel position of the target frame in planar view by using the motion vectors $\triangle V_f$ that the Unity engine provides.

$$I_{t,f} = I_{s,f} - \triangle V_f \tag{1}$$

We further convert the six planar pixel-wise motion vector maps into equirectangular form by using the camera rotation information $R$ and intrinsic matrix $K$.

$$F = \sum_{f}^{Face} (R_f K^{-1} I_{t,f} - R_f K^{-1} I_{s,f}) \tag{2}$$

$$Face = \{front, back, top, bottom, left, right\} \tag{3}$$

To optimize the performance for generating data, we implement the formula in a fragment shader to compute these in parallel. Furthermore, to optimize the

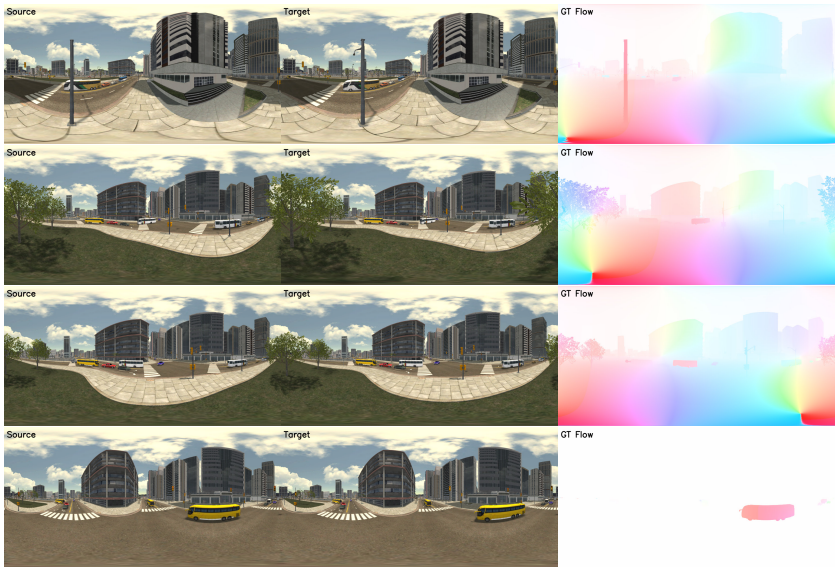



Fig. 1: Example of City Scene

inverse matrix computation, we render each cube face in a square shape, and normalize the image grid $I_f$ to the range $[-1, 1]$. So that the intrinsic matrix $K$ can be presented as a 3*3 identity matrix to facilitate the computation.

## 2 More Details of Our Panoramic Scenes

### 2.1 City Scene

We divide the whole scene into two parts to avoid data overlapping. One half of the city is used to generate training data, and another half is used for generating the validation and testing data. Some examples are shown in Fig 1.

### 2.2 Equirectangular FlyingThings Scene

In order to ensure data diversity, random translation and rotation have been applied to the objects. We took a long shot for nearly 3000 frames and manually separated them into training, validation, and testing datasets. Some examples are shown in Fig 2.

## 3 Relationship between cylindrical projection and equirectangular projection

The optical flow predictions from the Tri-Cylindrical projection have to be converted to the equirectangular format for fair EPE comparisons. In this process,

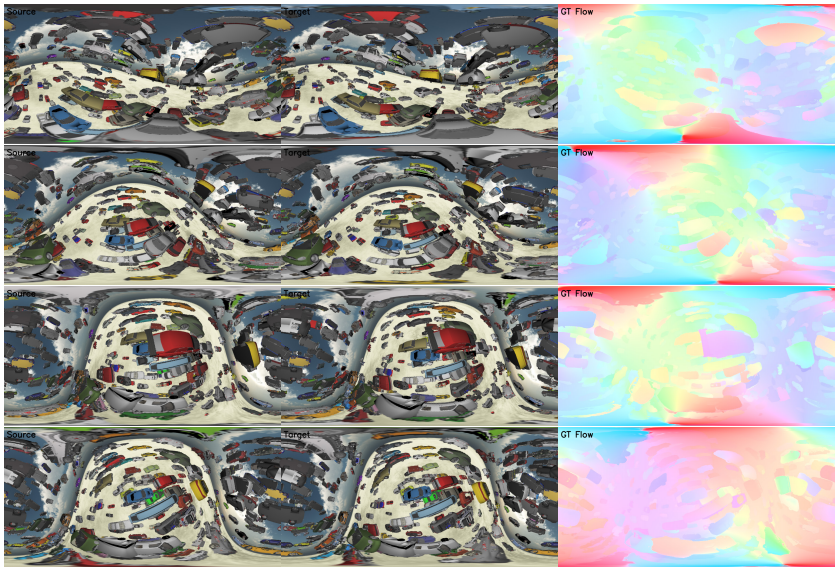
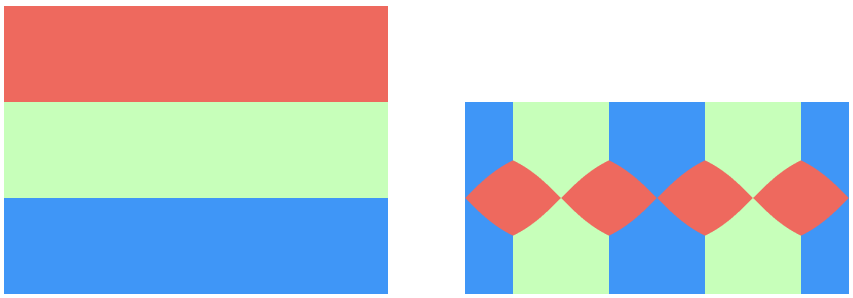



Fig. 2: Example of Equirectangular FlyingThings Scene

only the pixels which are valid in the Tri-Cylindrical weight mask are eligible to be converted into the equirectangular image. This principle ensures every pixel on the equirectangular image has a unique corresponding position on the Tri-Cylindrical image. As we design the algorithm based on the least distance to equators X, Y, and Z, the corresponding regions of a cylindrical image in an equirectangular image are as shown in Fig. 3. In this example, The red, green, and blue areas in Fig. 3 show the three projection cylinders in the Tri-Cylindrical image and the corresponding regions in the equirectangular image.



(a) A Tri-Cylindrical projection image (b) Equirectangular image

Fig. 3: Corresponding regions between our cylindrical projection and equirectangular projection.

| Loss | L1 | L2 | L3 | L4 |
|---|---|---|---|---|
| City100UR | 7.86 | 10.70 | **3.16** | 11.90 |

Table 1: Results of the models trained using different PWC losses

# 4 More Details of Our Experiments

## 4.1 Losses of PWC-Net in Baseline Selection

The loss comparison includes four different losses: First, we evaluate the original PWC implementation that sums up all the EPEs of each layer, which is denoted as $L_1$. Here, the $\alpha_l$ is a weight term applied to each layer, which is 0.32, 0.08, 0.02, 0.01 and 0.005, respectively. Furthermore, we apply the solid angle weight term as $\beta_x$, which is shown as $L_2$.

$$L_1 = \sum_{l}^{L} \alpha_l \sum_{x}^{n} \left\| W_{predict}^{l} - W_{GT}^{l} \right\|_2 \tag{4}$$

$$L_2 = \sum_{l}^{L} \alpha_l \sum_{x}^{n} \beta_x \left\| W_{predict}^{l} - W_{GT}^{l} \right\|_2 \tag{5}$$

Moreover, we test the loss function that only applies to the final output layer, which is denoted as $L_3$. Finally, we sum up the average EPE of all the pyramid layers, which is denoted as $L_4$. Our comparison result is presented in Tab. 1.

$$L_3 = \frac{1}{n} \sum_{x}^{n} \left\| W_{predict}^{top} - W_{GT}^{top} \right\|_2 \tag{6}$$

$$L_4 = \sum_{l_0}^{L} (\frac{1}{n} \sum_{x}^{n} \left\| W_{predict}^{l} - W_{GT}^{l} \right\|_2) \tag{7}$$

We choose $L_3$ as our final pixel-wise loss function to train the PWC-Net for our baseline model.

## 4.2 Input of the Fusion Network

One alternative design of our fusion layer structure not only takes the RGB image with predicted optical flow, but also includes two extra channels that denote the latitude and longitude of each pixel. We compared its performance with our current design. The result shows that current fusion layer implementation is better (EPE 2.61 vs 2.65 on City100UR datasets).

# 5 More Results

Fig. 4: Visual comparison of the optical flow estimation results from City100 using the following methods: PWC, tangent image-based method (TanImg), FlowNet2, single equirectangular projection (E), single cylinder projection (C), single cube-padding projection (P), and our final fusion model (Fusion).

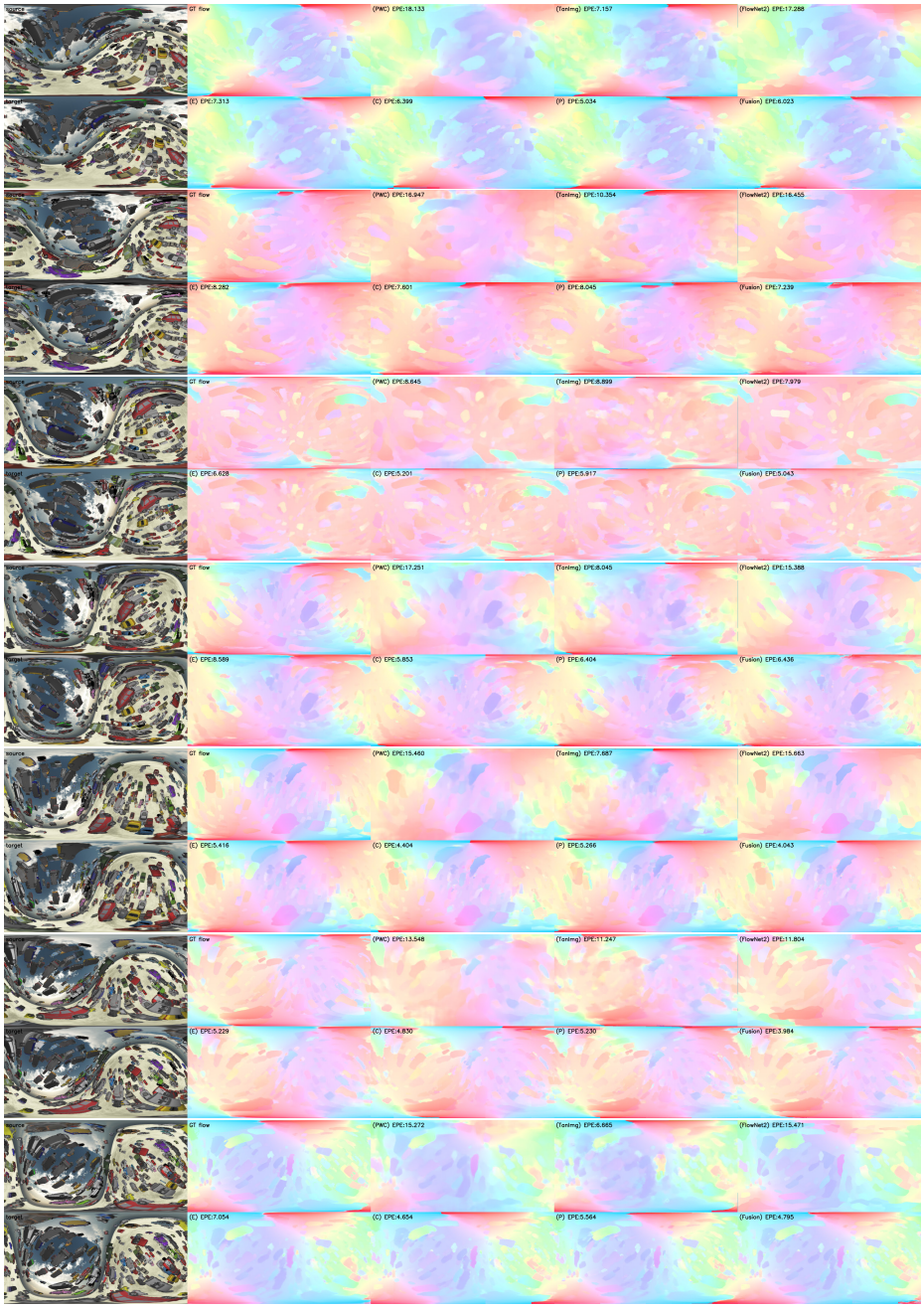


Fig. 5: Visual comparison of the optical flow estimation results from EFT100 using the following methods: PWC, tangent image-based method (TanImg), FlowNet2, single equirectangular projection (E), single cylinder projection (C), single cube-padding projection (P), and our final fusion model (Fusion).