

Supplementary Material

Generative Appearance Flow: A Hybrid Approach for Outdoor View Synthesis

Muhammad Usman Rafique¹

usman.rafique@uky.edu

Hunter Blanton¹

hunter.blanton@uky.edu

Noah Snavely²

snavely@cs.cornell.edu

Nathan Jacobs¹

nathan.jacobs@uky.edu

¹ Multimodal Vision Research Lab,
University of Kentucky,
Lexington, Kentucky

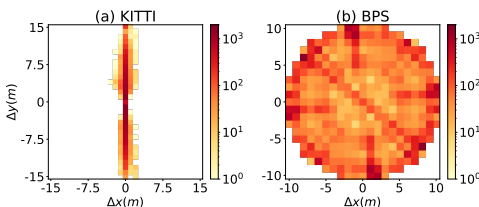
² Computer Science
Cornell Tech, Cornell University
New York City, New York

1 BPS Dataset

A comparison of KITTI and Brooklyn Panorama Synthesis (BPS) datasets is shown in Table 1. Note that BPS has more training examples, wider field of view, and larger average motion. Figure 1(a) shows the distribution of displacement vectors between image pairs in KITTI. The distribution of camera motion in KITTI is shown in vehicle relative frame and shows only forward and sideways motion, as sideways motion is negligible. Figure 1(b) shows the distribution of motion vectors for BPS, expressed as vector difference in UTM coordinates. From this, we can see that there is a much greater diversity of motions.

An example from the BPS dataset is shown in Figure 2. If we consider only a perspective cutout (as shown in green boundary), we can see that reasonable novel view synthesis cannot be expected because in the target image, a car appears which was not visible in the input picture. On the other hand, if we use fully panoramic image, this problem can be avoided. Note that in both KITTI and BPS, there is a possibility of uncertainty because of moving objects such as cars.

© 2020. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.



	Training Examples	Image Size	Horizontal FOV (w/ crop)	Mean Distance
KITTI	20409	256 × 256	82° (20.5°)	5.02m
BPS	40592	960 × 160	360°	8.49m

Figure 1: Distribution of pair-wise distances in the KITTI (a) and BPS dataset (b).

Table 1: Comparison of BPS and KITTI datasets.



Figure 2: An example pair of panoramic images from the BPS dataset. If we consider only a perspective cutout region (green boundary), it is not possible to synthesize the target image from the input cutout as an unseen object appears.

2 Network Architectures

We provide detailed architectures of the networks used in this paper. All conv2d layers have a filter size 3×3 , unless otherwise specified. Every convolution is followed by a batch-normalization layer. We use *ReLU* activation on all hidden layers, unless otherwise specified. We now describe the architectures of all our sub-networks.

2.1 Improved Appearance Flow (AF++)

We use a ResNet-18 encoder which produces 512 feature maps. For motion encoding, we use our distributed motion encoding that yields 50 feature maps for BPS (25 for each axis). For KITTI, we use 83 feature maps (21 for both x and y axes, and 42 for z axis). We concatenate these motion feature maps with image feature maps. The decoder architecture is shown in Table 2.

Since the last conv2d layer produces the flow prediction in the range $[-1, 1]$, we use *tanh* activation instead of *ReLU*.

2.2 Flow-Guided Direct Synthesis (FDS)

For FDS, we use ResNet-50 encoder followed by a 1×1 conv2d layer to reduce memory requirements by transforming 2048 feature maps to 16. The predicted flow (from AF++) is applied to these feature maps. The decoder architecture is shown in Table 3. We use *ReLU* activation for all convolutions, except that last layer where we use *sigmoid* activation function.

2.3 Adaptive Image Fusion

To get fusion weights for both flow-based prediction from AF++ and directly synthesized prediction from FDS, we train a fusion network. The inputs for the fusion network are

Type (name)	Inputs	Channels
coord-conv2d (conv1.1)	feature maps	512
conv2d (conv1.2)	conv1.1	256
conv2d (conv2.1)	up(conv1.2)	256
conv2d (conv2.2)	conv2.1	128
conv2d (conv3.1)	up(conv2.2)	64
conv2d (conv3.2)	conv3.1	32
coord-conv2d (conv4.1)	up(conv3.2)	8
conv2d (conv5)	conv4.1	8
conv2d (conv8)	tanh(conv7)	2

Table 2: AF++ decoder.

Type (name)	Inputs	Channels
conv2d (conv1.1)	up(warped features)	256
conv2d (conv1.2)	conv1.1	256
conv2d (conv2.1)	up(conv1.2)	128
conv2d (conv2.2)	conv2.1	128
conv2d (conv3.1)	up(conv2.2)	64
conv2d (conv3.2)	sigmoid(conv3.1)	3

Table 3: FDS Decoder.



Figure 3: Examples with predicted flow field. We can see that predicted flows correspond to the structure of the scene and depths of objects. Objects in the direction of travel (center) have lower flow predictions than the nearby buildings.

flow-based prediction, directly-synthesized image, and the flow field predicted from the flow network AF++. The fusion network is a standard U-Net with only some minor changes. Output prediction has the same size as the input image and predictions. We use a single channel output and use *sigmoid* activation. We then use the following equation to fuse predictions from AF++ and FDS to get the final output:

$$\hat{I} = A \cdot \hat{I}^F + (1 - A) \cdot \hat{I}^G, \quad (1)$$

where \hat{I}^F and \hat{I}^G are the predictions from AF++ and FDS, respectively. The fusion network architecture is shown in Table 4.

3 Flow Field

We show the flow field predicted by AF++ in Figure 3. It can be seen that the flow fields correspond to the objects in the scene and their depths.

Type (name)	Inputs	Output Channels
conv2d (conv1.1)	cat(AF++, FDS, Flow)	64
conv2d (conv1.2)	conv1.1	64
conv2d (conv2.1)	pool(conv1.2)	128
conv2d (conv2.2)	conv2.1	128
conv2d (conv3.1)	pool(conv2.2)	256
conv2d (conv3.2)	conv3.1	256
conv2d (conv4.1)	pool(conv3.2)	512
conv2d (conv4.2)	conv4.1	512
conv2d (conv5.1)	pool(conv4.2)	512
conv2d (conv5.2)	conv5.1	512
conv2d (conv6.1)	up(conv5.2+conv3.2)	256
conv2d (conv6.2)	conv6.1	256
conv2d (conv7.1)	up(conv6.2+conv2.2)	128
conv2d (conv7.2)	conv7.1	128
conv2d (conv8.1)	up(conv7.2+conv2.2)	64
conv2d (conv8.2)	conv8.1	64
conv2d (conv9.1)	up(conv8.2+conv1.2)	64
conv2d (conv9.2)	conv9.1	64
conv3d (conv10)	sigmoid(conv9.2)	1

Table 4: Fusion U-Net