Supplementary Material for
Self-Supervised Viewpoint Learning From Image Collections

Siva Karthik Mustikovela[1,2*] Varun Jampani[1] Shalini De Mello[1]
Sifei Liu[1] Umar Iqbal[1] Carsten Rother[2] Jan Kautz[1]
[1]NVIDIA [2]Heidelberg University

{siva.mustikovela, carsten.rother}@iwr.uni-heidelberg.de; varunjampani@gmail.com;
{shalinig, sifeil, uiqbal, jkautz}@nvidia.com

1. Overview

In this supplement, we provide the architectural and training details of our SSV framework. In Section 2 we describe the architectures of the both viewpoint ($\mathcal{V}$) and synthesis ($\mathcal{S}$) networks. In Section 3 we present the various training hyperparameters and the training schedule. In Section 4 we examine the memory requirements and runtime of SSV. In Section 5 we provide additional visual viewpoint estimation results for all object categories (*i.e.*, face, car, bus and train).

2. Network Architecture

The network architectures of the viewpoint and synthesis networks are detailed in tables 1 and 2, respectively. Both $\mathcal{V}$ and $\mathcal{S}$ operate at an image resolution of 128x128 pixels. $\mathcal{V}$ has an input size of 128x128. $\mathcal{S}$ synthesizes images at the same resolution. We use Instance Normalization [4] in the viewpoint network. For the synthesis network, the size of the style code $\boldsymbol{z}_s$ is 128 for faces and 200 for the other objects (car, bus and train). $\boldsymbol{z}_s$ is mapped to affine transformation parameters $(\gamma(\boldsymbol{z}_s), \sigma(\boldsymbol{z}_s))$, which are in turn used by adaptive instance normalization(AdaIN) [1] to control the style of the synthesized images.

3. Training Details

SSV is implemented in Pytorch [3]. We open-source our code required to reproduce the results at https://github.com/NVlabs/SSV. We train both our viewpoint and synthesis networks from scratch by initializing all weights with a normal distribution $\mathcal{N}(0, 0.2)$ and zero bias. The learning rate is 0.0001 for both ($\mathcal{V}$) and ($\mathcal{S}$). We use the ADAM [2] optimizer with betas (0.9, 0.99) and no weight decay. We train the networks for 20 epochs.

Training Cycle In each training iteration, we optimize $\mathcal{V}$ and $\mathcal{S}$ alternatively. In the $\mathcal{V}$ optimization step, we compute the generative consistency, discriminator loss and the symmetry constraint (Sections 3.1, 3.2, 3.3 in the main paper). We freeze the parameters of $\mathcal{S}$, compute the gradients of the losses with respect to parameters of $\mathcal{V}$ and do an update step for it. In an alternative step, while optimizing $\mathcal{S}$, we compute the paired style and viewpoint consistency, flip image consistency and the adversarial loss (Section 4 in the paper). We freeze the parameters of $\mathcal{V}$, compute the gradients of the losses with respect to parameters of $\mathcal{S}$ and do an update step for it. We train separate networks for each object category.

4. Runtime and Memory

Our viewpoint network $\mathcal{V}$ runs real-time with 76 FPS. That is, the inference takes 13 milliseconds on an NVIDIA Titan X Pascal GPU for a single image. The memory consumed is 900MB. We use a small network for viewpoint estimation for real-time performance and low-memory consumption.

*Siva Karthik Mustikovela was an intern at NVIDIA during the project.

	Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
	Conv	1x1	1	LReLU	-	128x128x128
Backbone Layers	Conv2D	3x3	1	LReLU	Instance Norm	128X128x256
	Conv2D	3x3	1	LReLU	Instance Norm	128X128x256
	Interpolate (scale = 0.5)					
	Conv2D	3x3	1	LReLU	Instance Norm	64X64x512
	Conv2D	3x3	1	LReLU	Instance Norm	64X64x512
	Interpolate (scale = 0.5)					
	Conv2D	3x3	1	LReLU	Instance Norm	32X32x512
	Conv2D	3x3	1	LReLU	Instance Norm	32X32x512
	Interpolate (scale = 0.5)					
	Conv2D	3x3	1	LReLU	Instance Norm	16X16x512
	Conv2D	3x3	1	LReLU	Instance Norm	16X16x512
	Interpolate (scale = 0.5)					
	Conv2D	3x3	1	LReLU	Instance Norm	8X8x512
	Conv2D	3x3	1	LReLU	Instance Norm	8X8x512
	Interpolate (scale = 0.5)					
	Conv2D	3x3	1	LReLU	Instance Norm	4X4x512
	Conv2D	4x4	1	LReLU	-	1X1x512
	Backbone ouput					
	FC-real/fake	-	-	-	-	1
	FC-style	-	-	-	-	code_dim
Azimuth	FC	-	-	LReLU	-	256
	FC - $\lvert\hat{a}\rvert$	-	-	-	-	2
	FC - sign($\hat{a}$)	-	-	-	-	4
Elevation	FC	-	-	LReLU	-	256
	FC - $\lvert\hat{e}\rvert$	-	-	-	-	2
	FC - sign($\hat{e}$)	-	-	-	-	4
Tilt	FC	-	-	LReLU	-	256
	FC - $\lvert\hat{e}\rvert$	-	-	-	-	2
	FC - sign($\hat{e}$)	-	-	-	-	4

Table 1. **Viewpoint Network Architecture.** The network contains a backbone whose resultant fully-connected features are shared by the heads that predict (a) real/fake scores, (b) style codes, and (c) heads that predict azimuth, elevation and tilt values. All LReLU units have a slope of 0.2. FC indicates a fully connected layer.

5. Visual Results

In figures 1, 2, 4, we present some additional visual results for the various object categories (faces, cars, buses and trains). It can be seen that the viewpoint estimation network reliably predicts viewpoint. For cars, it generalizes to car models like race cars and formula-1 cars, which are not seen by SSV during training. In each figure, we also show some failure cases in the last row. For faces, We observe that failures are caused in cases where the viewpoints contain extreme elevation or noisy face detection. For cars, viewpoint estimation is noisy when there is extreme blur in the image or the if the car is heavily occluded to the extent where it is difficult to identify it as a car. For buses, viewpoint estimation is erroneous when there is ambiguity between the rear and front parts of the object.

	Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
	Input - 3D Code	-	-	-	-	4x4x4x512
Styled 3D Convs	Conv 3D	3x3	1	LReLU	AdaIN	4x4x4x512
	Conv 3D	3x3	1	LReLU	AdaIN	4x4x4x512
	Interpolate (scale = 2)					
	Conv 3D	3x3	1	LReLU	AdaIN	8x8x8x512
	Conv 3D	3x3	1	LReLU	AdaIN	8x8x8x512
	Interpolate (scale = 2)					
	Conv 3D	3x3	1	LReLU	AdaIN	16x16x16x256
	Conv 3D	3x3	1	LReLU	AdaIN	16x16x16x256
	3D Rotation					
	Conv 3D	3x3	1	LReLU	-	16x16x16x128
	Conv 3D	3x3	1	LReLU	-	16x16x16x128
	Conv 3D	3x3	1	LReLU	-	16x16x16x64
	Conv 3D	3x3	1	LReLU	-	16x16x16x64
Project	Collapse	-	-	-	-	16x16x(16.64)
	Conv	3x3	1	LReLU	-	16x16x1024
Styled 2D Convs	Conv 2D	3x3	1	LReLU	AdaIN	16x16x512
	Conv 2D	3x3	1	LReLU	AdaIN	16x16x512
	Interpolate (scale = 2)					
	Conv 2D	3x3	1	LReLU	AdaIN	32x32x256
	Conv 2D	3x3	1	LReLU	AdaIN	32x32x256
	Interpolate (scale = 2)					
	Conv 2D	3x3	1	LReLU	AdaIN	64x64x128
	Conv 2D	3x3	1	LReLU	AdaIN	64x64x128
	Interpolate (scale = 2)					
	Conv 2D	3x3	1	LReLU	AdaIN	128x128x64
	Conv 2D	3x3	1	LReLU	AdaIN	128x128x64
Out	Conv 2D	3x3	1	-	-	128x128x3

Table 2. **Synthesis Network Architecture.** This network contains a set of 3D and 2D convolutional blocks. A learnable 3D latent code is passed through stylized 3D convolution blocks, which also use style codes as inputs to their adaptive instance normalization(AdaIN [1]) layers. The resulting 3D features are then rotated using a rigid rotation via the input viewpoint. Following this, the 3D features are orthographically projected to become 2D features. These are then passed through a stylized 2D convolution network which has adaptive instance normalization layers to control the style of the synthesized image.

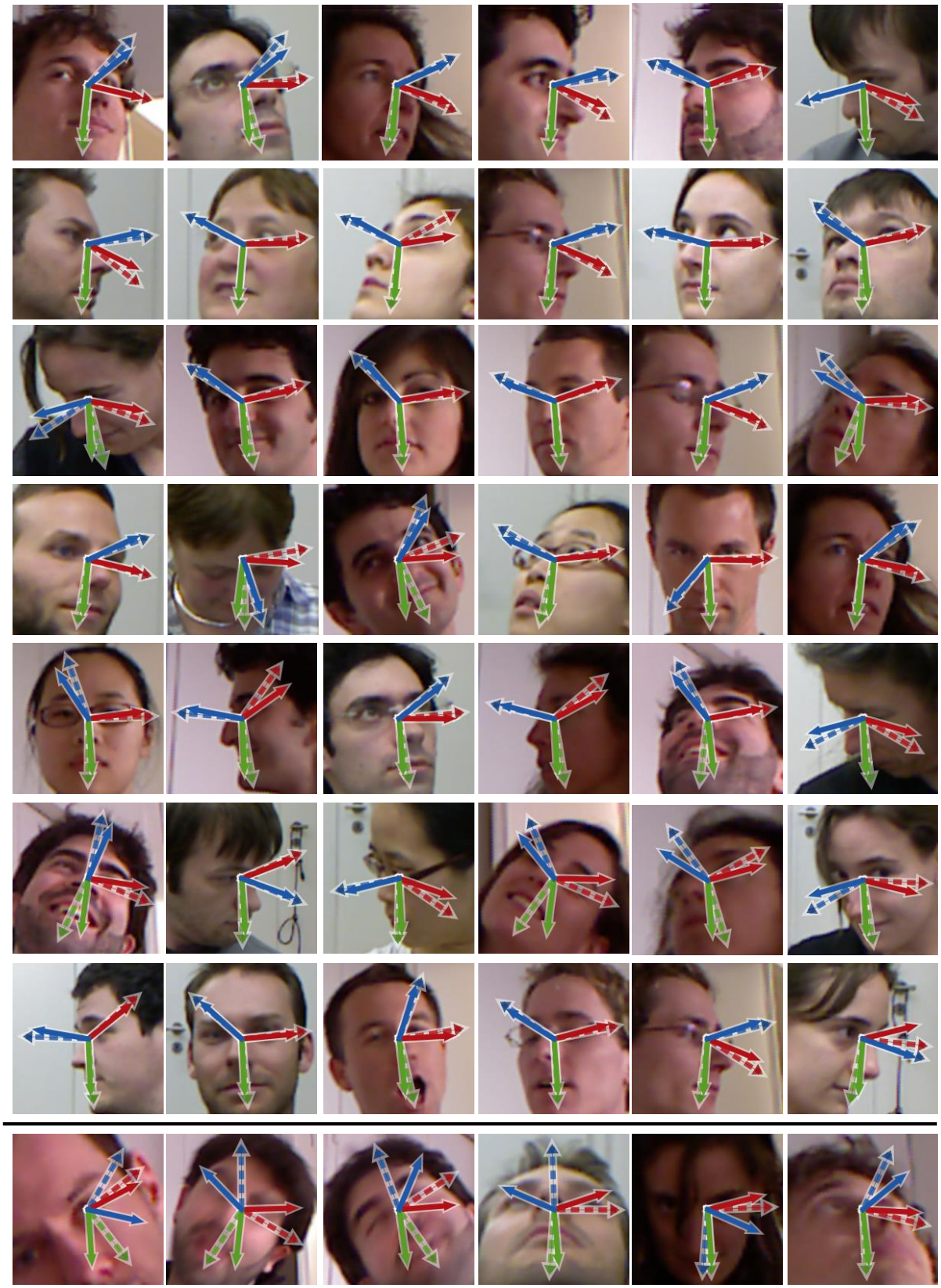

Figure 1. **Viewpoint estimation results for the face category.** SSV predicts reliable viewpoints for a variety of face poses with large variations in azimuth, elevation and tilt. The last row (below the black line) shows some erroneous cases where the faces are partially detected by the face detector or there are extreme elevation angles.

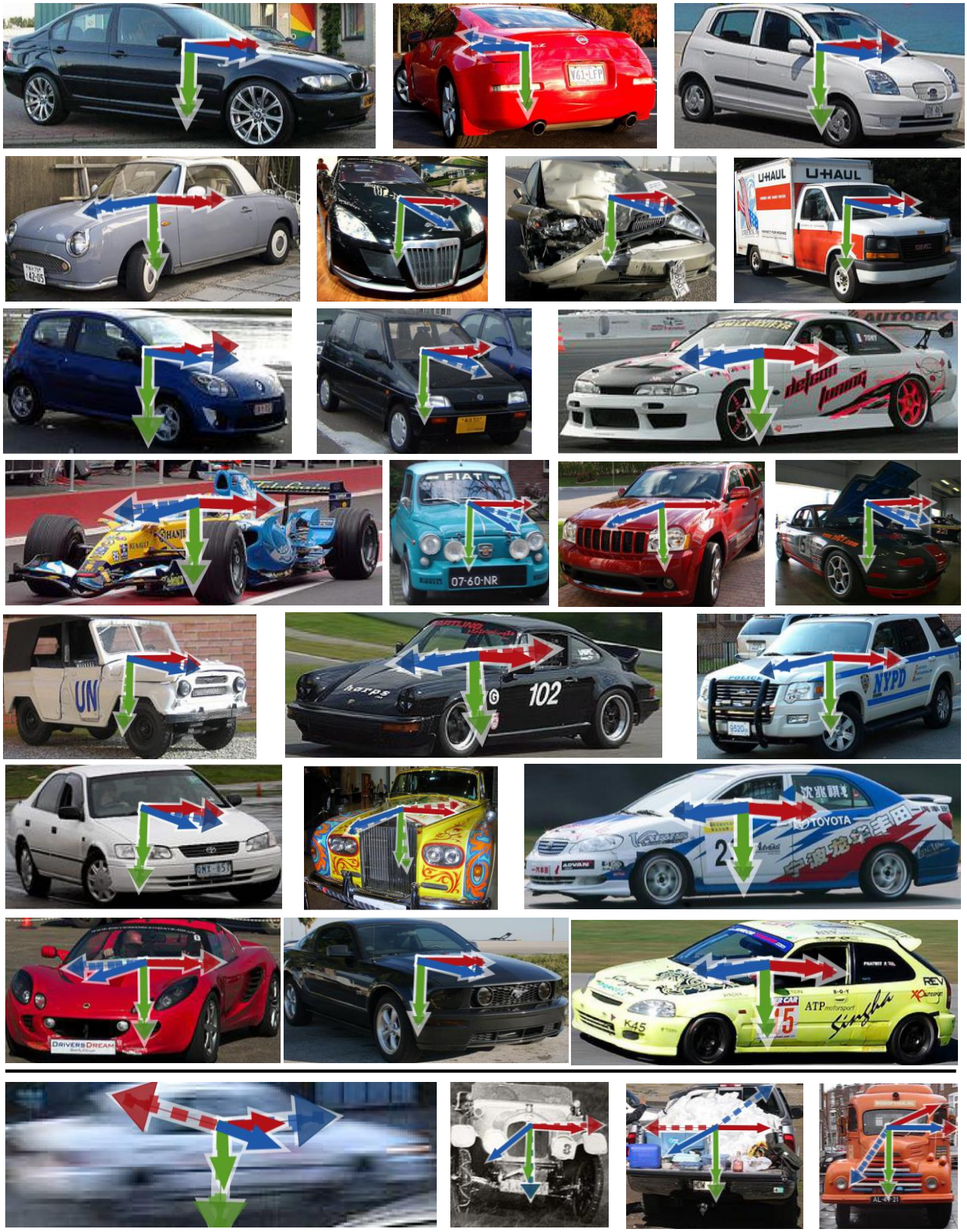

Figure 2. **Viewpoint estimation results for the car category.** SSV predicts reliable viewpoints for a variety of objects with large variations in azimuth, elevation and tilt. It generalizes to car models like race cars and formula-1 cars, which are not seen by SSV during training. The last row (below the black line) shows some erroneous cases where the objects have extreme motion blur or are heavily occluded to the extent where it is difficult to identify it as a car.

Figure 3. **Viewpoint estimation results for the bus category .** SSV predicts reliable viewpoints for a variety of buses with large variations in azimuth, elevation and tilt. The last row (below the black line) shows erroneous viewpoints when there is ambiguity between the rear and front parts of the object.

Figure 4. **Viewpoint estimation results for the train category.** SSV predicts reliable viewpoints for a variety of objects with large variations in azimuth, elevation and tilt. The last row (below the black line) shows the erroneous viewpoints predicted by SSV.

References

[1] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1, 3

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPSW*, 2017. 1

[4] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017. 1