

# Self-Supervised Viewpoint Learning From Image Collections

Siva Karthik Mustikovela<sup>1,2\*</sup> Varun Jampani<sup>1</sup> Shalini De Mello<sup>1</sup>

Sifei Liu<sup>1</sup> Umar Iqbal<sup>1</sup> Carsten Rother<sup>2</sup> Jan Kautz<sup>1</sup>

<sup>1</sup>NVIDIA <sup>2</sup>Heidelberg University

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

## Abstract

Training deep neural networks to estimate the viewpoint of objects requires large labeled training datasets. However, manually labeling viewpoints is notoriously hard, error-prone, and time-consuming. On the other hand, it is relatively easy to mine many unlabelled images of an object category from the internet, e.g., of cars or faces. We seek to answer the research question of whether such unlabeled collections of in-the-wild images can be successfully utilized to train viewpoint estimation networks for general object categories purely via self-supervision. Self-supervision here refers to the fact that the only true supervisory signal that the network has is the input image itself. We propose a novel learning framework which incorporates an analysis-by-synthesis paradigm to reconstruct images in a viewpoint aware manner with a generative network, along with symmetry and adversarial constraints to successfully supervise our viewpoint estimation network. We show that our approach performs competitively to fully-supervised approaches for several object categories like human faces, cars, buses, and trains. Our work opens up further research in self-supervised viewpoint learning and serves as a robust baseline for it. We open-source our code at <https://github.com/NVlabs/SSV>.

## 1. Introduction

3D understanding of objects from 2D images is a fundamental computer vision problem. Object viewpoint (azimuth, elevation and tilt angles) estimation provides a pivotal link between 2D imagery and the corresponding 3D geometric understanding. In this work, we tackle the problem of object viewpoint estimation from a single image. Given its central role in 3D geometric understanding, viewpoint estimation is useful in several vision tasks such as object manipulation [68], 3D reconstruction [34], image synthesis [8] to name a few. Estimating viewpoint from a single image is highly challenging due to the inherent ambiguity of 3D understanding from a 2D image. Learning-based ap-

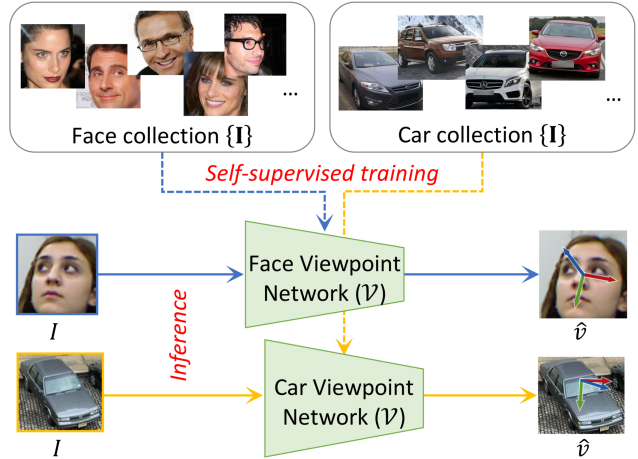


Figure 1. **Self-supervised viewpoint learning.** We learn a single-image object viewpoint estimation network for each category (face or car) using only a collection of images without ground truth.

proaches, e.g., [37, 16, 77, 39, 56, 63, 17, 70], using neural networks that leverage a large amount of annotated training data, have demonstrated impressive viewpoint estimation accuracy. A key requirement for such approaches is the availability of large-scale human annotated datasets, which is very difficult to obtain. A standard way to annotate viewpoints is by manually finding and aligning a rough morphable 3D or CAD model to images [12, 79, 67], which is a tedious and slow process. This makes it challenging to create large-scale datasets with viewpoint annotations. Most existing works [16, 14, 56, 37, 79, 17] either rely on human-annotated viewpoints or augment real-world data with synthetic data. Some works [16] also leverage CAD models during viewpoint inference.

In this work, we propose a self-supervised learning technique for viewpoint estimation of general objects that learns from an object image collection without the need for any viewpoint annotations (Figure 1). By image collection, we mean a set of images containing objects of a category of interest (say, faces or cars). Since viewpoint estimation assumes known object bounding boxes, we also assume that the image collection consists of tightly bounded object im-

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

ages. Being self-supervised in nature, our approach provides an important advancement in viewpoint estimation as it alleviates the need for costly viewpoint annotations. It also enables viewpoint learning on object categories that do not have any existing ground-truth annotations.

Following the analysis-by-synthesis paradigm, we leverage a viewpoint aware image synthesis network as a form of self-supervision to train our viewpoint estimation network. We couple the viewpoint network with the synthesis network to form a complete cycle and train both together. To self-supervise viewpoint estimation, we leverage cycle-consistency losses between the viewpoint estimation (analysis) network and a viewpoint aware generative (synthesis) network, along with losses for viewpoint and appearance disentanglement, and object-specific symmetry priors. During inference, we only need the viewpoint estimation network, without the synthesis network, making viewpoint inference simple and fast for practical purposes. As per our knowledge, ours is the first self-supervised viewpoint learning framework that learns 3D viewpoint of general objects from image collections in-the-wild. We empirically validate our approach on the human head pose estimation task, which on its own has attracted considerable attention [79, 4, 57, 71, 33, 6, 17, 70] in computer vision research. We demonstrate that the results obtained by our self-supervised technique are comparable to those of fully-supervised approaches. In addition, we also demonstrate significant performance improvements when compared to viewpoints estimated with self-supervisedly learned keypoint predictors. To showcase the generalization of our technique, we analyzed our approach on object classes such as cars, buses, and trains from the challenging Pascal3D+ [67] dataset. We believe this work opens up further research in self-supervised viewpoint learning and would also serve as a robust baseline for future work.

To summarize, our main contributions are:

- We propose a novel analysis-by-synthesis framework for learning viewpoint estimation in a purely self-supervised manner by leveraging cycle-consistency losses between a viewpoint estimation and a viewpoint aware synthesis network. To our understanding, this is one of first works to explore the problem of self-supervised viewpoint learning for general objects.
- We introduce generative, symmetric and adversarial constraints which self-supervise viewpoint estimation learning just from object image collections.
- We perform experiments for head pose estimation on the BIWI dataset [12] and for viewpoint estimation of cars, buses and trains on the challenging Pascal3D+ [67] dataset and demonstrate competitive accuracy in comparison to fully-supervised approaches.

## 2. Related Work

**Viewpoint estimation** Several successful learning-based viewpoint estimation techniques have been developed for general object categories that either regress orientation directly [40, 39, 56, 63, 37, 49]; locate 2D keypoints and fit them to 3D keypoints [16, 48, 77]; or predict 3D shape and viewpoint parameters [34]. These techniques require object viewpoint annotations during training, either in the form of angular values; or 2D and 3D keypoints and use large annotated datasets, *e.g.*, Pascal3D+ [67] and Object-Net3D [66] with 12 and 100 categories, respectively. These datasets were annotated via a tedious manual process of aligning best-matched 3D models to images – a procedure that is not scalable easily to larger numbers of images or categories. To circumvent this problem, existing viewpoint algorithms augment real-world data with synthetic images [16, 14, 56, 37]; assume auxiliary supervision and learn the related aspects (*e.g.*, 3D keypoints) along with viewpoint [77, 58]; or try to learn from very few labeled examples of novel categories [62].

**Head pose estimation** Separate from the above-mentioned works, learning-based head pose estimation techniques have also been studied extensively [79, 4, 57, 71, 33, 6, 17, 70]. These works learn to either predict facial landmarks from data with varying levels of supervision ranging from full [79, 4, 57, 71, 33], partial [20], or no supervision [22, 76]; or learn to regress head orientation directly in a fully-supervised manner [6, 51, 17, 70]. The latter methods perform better than those that predict facial points [70]. To avoid manual annotation of head pose, prior works also use synthetic datasets [79, 17]. On the other hand, several works [59, 13, 61, 53] propose learning-based approaches for dense 3D reconstruction of faces via in-the-wild image collections and some use analysis-by-synthesis [59, 61]. However, they are not purely self-supervised and use either facial landmarks [59], dense 3D surfaces [13] or both [61] as supervision.

**Self-supervised object attribute discovery** Several recent works try to discover 2D object attributes like landmarks [76, 60, 24] and part segmentation [22, 9] in a self-supervised manner. These works are orthogonal to ours as we estimate 3D viewpoint. Some other works such as [35, 23, 18] make use of differentiable rendering frameworks to learn 3D shape and/or camera viewpoint from a single or multi-view image collections. Because of heavy reliance on differentiable rendering, these works mainly operate on synthetic images. In contrast, our approach can learn viewpoints from image collections in the wild. Some works learn 3D reconstruction from in-the-wild image collections, but use annotated object silhouettes along with other annotations such as 2D semantic keypoints [26],

category-level 3D templates [31]; or multiple views of each object instance [28, 65, 43]. In contrast, we use no additional supervision other than the image collections that comprise of independent object images. To the best we know, no prior works propose to learn viewpoint of general objects in a purely self-supervised manner from in-the-wild image collections.

### 3. Self-Supervised Viewpoint Learning

**Problem setup** We learn a viewpoint estimation network  $\mathcal{V}$  using an in-the-wild image collection  $\{\mathbf{I}\}$  of a specific object category without annotations. Since viewpoint estimation assumes tightly cropped object images, we also assume that our image collection is composed of cropped object images. Figure 1 shows some samples in the face and car image collections. During inference, the viewpoint network  $\mathcal{V}$  takes a single object image  $I$  as input and predicts the object 3D viewpoint  $\hat{v}$ .

**Viewpoint representation** To represent an object viewpoint  $\hat{v}$ , we use three Euler angles, namely azimuth ( $\hat{a}$ ), elevation ( $\hat{e}$ ) and in-plane rotation ( $\hat{t}$ ) describing the rotations around fixed 3D axes. For the ease of viewpoint regression, we represent each Euler angle, e.g.,  $a \in [0, 2\pi]$ , as a point on a unit circle with 2D coordinates  $(\cos(a), \sin(a))$ . Following [37], instead of predicting co-ordinates on a 360° circle, we predict a positive unit vector in the first quadrant with  $|\hat{a}| = (|\cos(\hat{a})|, |\sin(\hat{a})|)$  and also the category of the combination of signs of  $\sin(\hat{a})$  and  $\cos(\hat{a})$  indicated by  $\text{sign}(\hat{a}) = (\text{sign}(\cos(\hat{a})), \text{sign}(\sin(\hat{a}))) \in \{(+, +), (+, -), (-, +), (-, -)\}$ . Given the predicted  $|\hat{a}|$  and  $\text{sign}(\hat{a})$  from the viewpoint network, we can construct  $\cos(\hat{a}) = \text{sign}(\cos(\hat{a}))|\cos(\hat{a})|$  and  $\sin(\hat{a}) = \text{sign}(\sin(\hat{a}))|\sin(\hat{a})|$ . The predicted Euler angle  $\hat{a}$  can finally be computed as  $\tan^{-1}(\sin(\hat{a})/\cos(\hat{a}))$ . In short, the viewpoint network performs both regression to predict a positive unit vector  $|a|$  and also classification to predict the probability of  $\text{sign}(a)$ .

**Approach overview and motivation** We learn the viewpoint network  $\mathcal{V}$  using a set of self-supervised losses as illustrated in Figure 2. To formulate these losses we use three different constraints, namely generative consistency, a symmetry constraint and a discriminator loss. Generative consistency forms the core of the self-supervised constraints to train our viewpoint network and is inspired from the popular analysis-by-synthesis learning paradigm [34]. This framework tries to tackle inverse problems (such as viewpoint estimation) by modelling the forward process of image or feature synthesis. A synthesis function  $\mathcal{S}$  models the process of generating an image of an object from a basic representation and a set of parameters. The goal of the analysis function is to infer the underlying parameters which can best explain the formation of an observed input image. Bayesian frame-

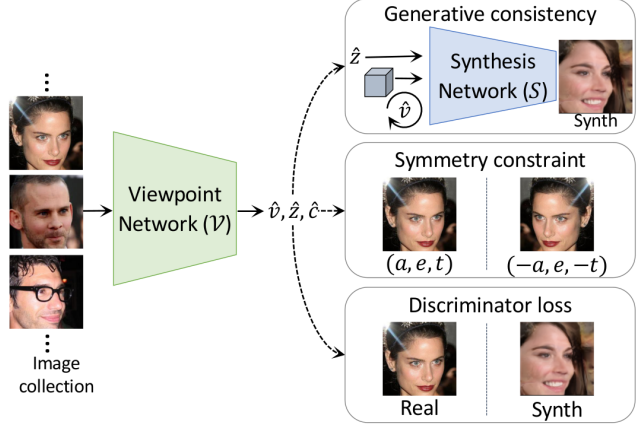


Figure 2. **Approach overview.** We use generative consistency, symmetry and discriminator losses to supervise the viewpoint network with a collection of images without annotations.

works such as [73] and inverse graphics [34, 28, 72, 38, 25] form some of the popular techniques that are based on the analysis-by-synthesis paradigm. In our setup, we consider the viewpoint network  $\mathcal{V}$  as the analysis function.

We model the synthesis function  $\mathcal{S}$ , with a viewpoint aware image generation model. Recent advances in Generative Adversarial Networks (GAN) [7, 27, 42] have shown that it is possible to generate high-quality images with fine-grained control over parameters like appearance, style, viewpoint, etc. Inspired by these works, our synthesis network generates an image, given an input  $v$ , which controls the viewpoint of the object and an input vector  $z$ , which controls the style of the object in the synthesized image. By coupling both the analysis ( $\mathcal{V}$ ) and synthesis ( $\mathcal{S}$ ) networks in a cycle, we learn both the networks in a self-supervised manner using cyclic consistency constraints described in 3.1 and shown in Figure 3. Since the synthesis network can generate high quality images based on controllable inputs  $v$  and  $z$ , these synthesized images can in turn be used as input to the analysis network ( $\mathcal{V}$ ) along with  $v$ ,  $z$  as the pseudo ground-truth. On the other hand, for a real world image, if  $\mathcal{V}$  predicts the correct viewpoint and style, these can be utilized by  $\mathcal{S}$  to produce a similar looking image. This effectively functions as image reconstruction-based supervision. In addition to this, similar to [7, 42] the analysis network also functions as a discriminator, evaluating whether the synthesized images are real or fake. Using a widely prevalent observation that several real-world objects are symmetric, we also enforce a prior constraint via a symmetry loss function to train the viewpoint network. Object symmetry has been used in previous supervised techniques such as [39] for data augmentation, but not as a loss function. In the following, we first describe the various loss constraints used to train the viewpoint network  $\mathcal{V}$  while assuming that we already have a trained synthesis network  $\mathcal{S}$ . In Section 4, we describe the loss constraints used to train

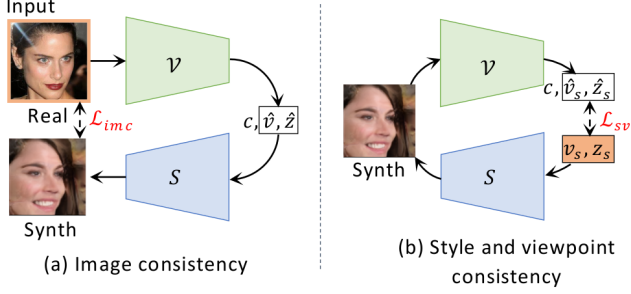


Figure 3. **Generative consistency.** The two cyclic (a) image consistency ( $\mathcal{L}_{imgc}$ ) and (b) style and viewpoint consistency ( $\mathcal{L}_{sv}$ ) losses make up generative consistency. The input to each cycle is highlighted in yellow. Image consistency enforces that an input real image, after viewpoint estimation and synthesis, matches its reconstructed synthetic version. Style and viewpoint consistency enforces that the input style and viewpoint provided for synthesis are correctly reproduced by the viewpoint network.

the synthesis network  $\mathcal{S}$ .

### 3.1. Generative Consistency

As Figure 3 illustrates, we couple the viewpoint network  $\mathcal{V}$  with the synthesis network  $\mathcal{S}$  to create a circular flow of information resulting in two consistency losses: (a) image consistency and (b) style and viewpoint consistency.

**Image consistency** Given a real image  $I$  sampled from a given image collection  $\{\mathbf{I}\}$ , we first predict its viewpoint  $\hat{v}$  and style code  $\hat{z}$  via the viewpoint network  $\mathcal{V}$ . Then, we pass the predicted  $\hat{v}$  and  $\hat{z}$  into the synthesis network  $\mathcal{S}$  to create the synthetic image  $\hat{I}_s$ . To train the viewpoint network, we use the image consistency between the input image  $I$  and corresponding synthetic image  $I_s$  with a perceptual loss:

$$\mathcal{L}_{imgc} = 1 - \langle \Phi(I), \Phi(\hat{I}_s) \rangle, \quad (1)$$

where  $\Phi(\cdot)$  denotes the conv5 features of an ImageNet-trained [10] VGG16 classifier [54] and  $\langle \cdot, \cdot \rangle$  denotes the cosine distance. Figure 3(a) illustrates the image consistency cycle.

**Style and viewpoint consistency** As illustrated in Figure 3(b), we create another circular flow of information with the viewpoint and synthesis networks, but this time starting with a random viewpoint  $v_s$  and a style code  $z_s$ , both sampled from uniform distributions, and input them to the synthesis network to create an image  $I_s = \mathcal{S}(v_s, z_s)$ . We then pass the synthetic image  $I_s$  to the viewpoint network  $\mathcal{V}$  that predicts its viewpoint  $\hat{v}_s$  and the style code  $\hat{z}_s$ . We use the sampled viewpoint and style codes for the synthetic image  $I_s$  as a pseudo GT to train the viewpoint network. Following [37], the viewpoint consistency loss  $\mathcal{L}_v(\hat{v}_1, \hat{v}_2)$  between two viewpoints  $\hat{v}_1 = (\hat{a}_1, \hat{e}_1, \hat{t}_1)$  and  $\hat{v}_2 = (\hat{a}_2, \hat{e}_2, \hat{t}_2)$  has two components for each Euler angle: (i) cosine proximity between the positive unit vectors  $\mathcal{L}_v^{|\alpha|} = -\langle |\hat{a}_1|, |\hat{a}_2| \rangle$  and

(ii) the cross-entropy loss  $\mathcal{L}_v^{\text{sign}(\alpha)}$  between the classification probabilities of  $\text{sign}(\hat{a}_1)$  and  $\text{sign}(\hat{a}_2)$ . The viewpoint consistency loss  $\mathcal{L}_v$  is a sum of the cross-entropy and cosine proximity losses for all the three Euler angles:

$$\mathcal{L}_v(\hat{v}_1, \hat{v}_2) = \sum_{\phi \in a, e, t} \mathcal{L}_v^{|\phi|} + \mathcal{L}_v^{\text{sign}(\phi)}. \quad (2)$$

The overall style and viewpoint loss between the sampled  $(v_s, z_s)$  and the predicted  $(\hat{v}_s, \hat{z}_s)$  is hence:

$$\mathcal{L}_{sv} = \|z_s - \hat{z}_s\|_2^2 + \mathcal{L}_v(v_s, \hat{v}_s). \quad (3)$$

While viewpoint consistency enforces that  $\mathcal{V}$  learns correct viewpoints for synthetic images, image consistency helps to ensure that  $\mathcal{V}$  generalizes well to real images as well, and hence avoids over-fitting to images synthesized by  $\mathcal{S}$ .

### 3.2. Discriminator Loss

$\mathcal{V}$  also predicts a score  $\hat{c}$  indicating whether an input image is real or synthetic. It thus acts as a discriminator in a typical GAN [15] setting, helping the synthesis network create more realistic images. We use the discriminator loss from Wasserstein-GAN [1] to update the viewpoint network using:

$$\mathcal{L}_{dis} = -\mathbb{E}_{x \sim p_{\text{real}}}[\mathcal{V}(x)] + \mathbb{E}_{\hat{x} \sim p_{\text{synth}}}[\mathcal{V}(\hat{x})], \quad (4)$$

where  $c = \mathcal{V}(x)$  and  $\hat{c} = \mathcal{V}(\hat{x})$  are the predicted class scores for the real and the synthesized images, respectively.

### 3.3. Symmetry Constraint

Symmetry is a strong prior observed in many commonplace object categories, e.g., faces, boats, cars, airplanes, etc. For categories with symmetry, we propose to leverage an additional symmetry constraint. Given an image  $I$  of an object with viewpoint  $(a, e, t)$ , the GT viewpoint of the object in a horizontally flipped image  $\text{flip}(I)$  is given by  $(-a, e, -t)$ . We enforce a symmetry constraint on the viewpoint network’s outputs  $(\hat{v}, \hat{z})$  and  $(\hat{v}^*, \hat{z}^*)$  for a given image  $I$  and its horizontally flipped version  $\text{flip}(I)$ , respectively. Let  $\hat{v} = (\hat{a}, \hat{e}, \hat{t})$  and  $\hat{v}^* = (\hat{a}^*, \hat{e}^*, \hat{t}^*)$  and we denote the flipped viewpoint of the flipped image as  $\hat{v}_f^* = (-\hat{a}^*, \hat{e}^*, -\hat{t}^*)$ . The symmetry loss is given as

$$\mathcal{L}_{sym} = \mathcal{D}(\hat{v}, \hat{v}_f^*) + \|\hat{z} - \hat{z}^*\|_2^2. \quad (5)$$

Effectively, for a given horizontally flipped image pair, we regularize that the network predicts similar magnitudes for all the angles and opposite directions for azimuth and tilt. Additionally, the above loss enforces that the style of the flipped image pair is consistent.

Our overall loss to train the viewpoint network  $\mathcal{V}$  is a linear combination of the aforementioned loss functions:

$$\mathcal{L}_V = \lambda_1 \mathcal{L}_{sym} + \lambda_2 \mathcal{L}_{imgc} + \lambda_3 \mathcal{L}_{sv} + \lambda_4 \mathcal{L}_{dis}, \quad (6)$$

where the parameters  $\{\lambda_i\}$  determine the relative importance of the different losses, which we empirically determine using a grid search.



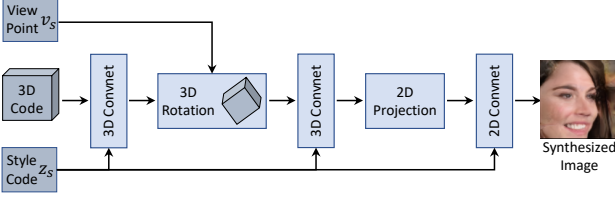
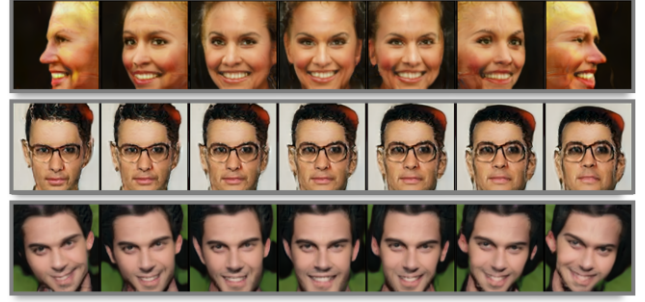


Figure 4. **Synthesis network overview.** The network takes viewpoint  $v_s$  and style code  $z_s$  to produce a viewpoint aware image.

#### 4. Viewpoint-Aware Synthesis Network

Recent advances in GANs such as InfoGAN [7], StyleGAN [27] and HoloGAN [42] demonstrate the possibility of conditional image synthesis where we can control the synthesized object’s attributes such as object class, viewpoint, style, geometry, etc. A key insight that we make use of in our synthesis network and which is also used in recent GANs such as HoloGAN [42] and other works [78, 32, 55], is that one can instill 3D geometric meaning into the network’s latent representations by performing explicit geometric transformations such as rotation on them. A similar idea has also been used successfully with other generative models such as auto-encoders [19, 50, 46]. Our viewpoint-aware synthesis network has a similar architecture to HoloGAN [42], but is tailored for the needs of viewpoint estimation. HoloGAN is a pure generative model with GAN losses to ensure realism and an identity loss to reproduce the input style code, but lacks a corresponding viewpoint prediction network. In this work, since we focus on viewpoint estimation, we introduce tight coupling of HoloGAN with a viewpoint prediction network and several novel loss functions to train it in a manner that is conducive to accurate viewpoint prediction.

**Synthesis network overview** Figure 4 illustrates the design of the synthesis network. The network  $\mathcal{S}$  takes a style code  $z_s$  and a viewpoint  $v_s$  to produce a corresponding object image  $I_s$ . The goal of  $\mathcal{S}$  is to learn a disentangled 3D representation of an object, which can be used to synthesize objects in various viewpoints and styles, hence aiding in the supervision of the viewpoint network  $\mathcal{V}$ . We first pass a learnable canonical 3D latent code through a 3D network, which applies 3D convolutions to it. Then, we rotate the resulting 3D representation with  $v_s$  and pass through an additional 3D network. We project this viewpoint-aware learned 3D code on to 2D using a simple orthographic projection unit. Finally, we pass the resulting 2D representation through a StyleGAN [27]-like 2D network to produce a synthesized image. The style and appearance of the image is controlled by the sampled style code  $z_s$ . Following StyleGAN [27], the style code  $z_s$  affects the style of the resulting image via adaptive instance normalization [21] in both the 3D and 2D representations. For stable training, we freeze  $\mathcal{V}$  while training  $\mathcal{S}$  and vice versa.



(a) Faces



(b) Cars

Figure 5. **Synthesis results.** Example synthetic images of (a) faces and (b) cars generated by the viewpoint-aware generator  $\mathcal{S}$ . For each row the style vector  $z$  is constant, whereas the viewpoint is varied monotonically along the azimuth (first row), elevation (second row) and tilt (third row) dimensions.

**Loss functions** Like the viewpoint network, we use several constraints to train the synthesis network, which are designed to improve viewpoint estimation. The first is the standard adversarial loss used in training WassersteinGAN[1]:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{x} \sim p_{\text{synth}}} [\hat{c}] \quad (7)$$

where  $\hat{c} = \mathcal{V}(\hat{x})$  is the class membership score predicted by  $\mathcal{V}$  for a synthesized image. The second is a paired version of the style and viewpoint consistency loss (Eqn. 3) described in Section 3.1, where we propose to use multiple paired  $(z_s, v_s)$  samples to enforce style and viewpoint consistency and to better disentangle the latent representations of  $\mathcal{S}$ . The third is a flip image consistency loss. Note that, in contrast to our work, InfoGAN [7] and HoloGAN [42] only use adversarial and style consistency losses.

**Style and viewpoint consistency with paired samples** Since we train the viewpoint network with images synthesized by  $\mathcal{S}$ , it is very important for  $\mathcal{S}$  to be sensitive and responsive to its input style  $z_s$  and viewpoint  $v_s$  parameters. An ideal  $\mathcal{S}$  would perfectly disentangle  $v_s$  and  $z_s$ . That means, if we fix  $z_s$  and vary  $v_s$ , the resulting object images should have the same style, but varying viewpoints. On the other hand, if we fix  $v_s$  and vary  $z_s$ , the resulting object images should have different styles, but a fixed viewpoint. We enforce this constraint with a paired version of the style and viewpoint consistency (Eqn. 3) loss where we sample 3 different pairs of  $(z_s, v_s)$  values by varying one parameter at a time as:  $\{(z_s^0, v_s^0), (z_s^0, v_s^1), (z_s^1, v_s^1)\}$ . We refer to this paired style and viewpoint loss as  $\mathcal{L}_{sv, \text{pair}}$ . The ab-

lation study in Section 5 suggests that this paired style and viewpoint loss helps to train a better synthesis network for our intended task of viewpoint estimation. We also observe qualitatively that the synthesis network successfully disentangles the viewpoints and styles of the generated images. Some example images synthesized by  $\mathcal{S}$  for faces and cars are shown in Figure 5. Each row uses a fixed style code  $z_s$  and we monotonically vary the input viewpoint  $v_s$  by changing one of its  $a$ ,  $e$  or  $t$  values across the columns.

**Flip image consistency** This is similar to the symmetry constraint used to train the viewpoint network, but applied to synthesized images. Flip image consistency forces  $\mathcal{S}$  to synthesize horizontally flipped images when we input appropriately flipped viewpoints. For the pairs  $\mathcal{S}(v_s, z_s) = I_s$  and  $\mathcal{S}(v_s^*, z_s) = I_s^*$ , where  $v_s^*$  has opposite signs for the  $a$  and  $t$  values of  $v_s$ , the flip consistency loss is defined as:

$$\mathcal{L}_{fc} = \|I_s - \text{flip}(I_s^*)\|_1 \quad (8)$$

where  $\text{flip}(I_s^*)$  is the horizontally flipped version of  $I_s^*$ .

The overall loss for the synthesis network is given by:

$$\mathcal{L}_S = \lambda_5 \mathcal{L}_{adv} + \lambda_6 \mathcal{L}_{sv, pair} + \lambda_7 \mathcal{L}_{fc} \quad (9)$$

where the parameters  $\{\lambda_i\}$  are the relative weights of the losses which we determine empirically using grid search.

## 5. Experiments

We empirically validate our approach with extensive experiments on head pose estimation and viewpoint estimation on other object categories of buses, cars and trains. We refer to our approach as ‘SSV’.

**Implementation and training details** We implement our framework in Pytorch[47]. We provide all network architecture details, and run-time and memory analyses in the supplementary material.

**Viewpoint calibration** The output of SSV for a given image  $I$  is  $(\hat{a}, \hat{e}, \hat{t})$ . However, since SSV is self-supervised, the co-ordinate system for predictions need not correspond to the actual canonical co-ordinate system of GT annotations. For quantitative evaluation, following the standard practice in self-supervised learning of features [11, 75, 5] and landmarks [22, 76, 60], we fit a linear regressor that maps the predictions of SSV to GT viewpoints using 100 randomly chosen images from the target test dataset. Note that this calibration with a linear regressor only rotates the predicted viewpoints to the GT canonical frame of reference. We do not update or learn our SSV network during this step.

### 5.1. Head Pose Estimation

Human faces have a special place among objects for viewpoint estimation and head pose estimation has attracted considerable research attention [79, 4, 57, 71, 33, 6, 17, 70]. The availability of large-scale datasets [52, 12] and the existence of ample research provides a unique opportunity to

|                 | Method  | Azimuth    | Elevation  | Tilt       | MAE        |
|-----------------|---|------------|------------|------------|------------|
| Self-Supervised | LMDIS [76] + PnP                                | 16.8       | 26.1       | 5.6        | 16.1       |
|                 | IMM [24] + PnP                                  | 14.8       | 22.4       | 5.5        | 14.2       |
|                 | SCOPS [22] + PnP                                | 15.7       | 13.8       | 7.3        | 12.3       |
|                 | HoloGAN [42]                                    | 8.9        | 15.5       | 5.0        | 9.8        |
|                 | HoloGAN [42] with $v$                           | 7.0        | 15.1       | 5.1        | 9.0        |
|                 | SSV w/o $\mathcal{L}_{sym} + \mathcal{L}_{imc}$ | 6.8        | 13.0       | 5.2        | 8.3        |
|                 | SSV w/o $\mathcal{L}_{imc}$                     | 6.9        | 10.3       | <b>4.4</b> | 7.2        |
|                 | SSV-Full  | <b>6.0</b> | <b>9.8</b> | <b>4.4</b> | <b>6.7</b> |
| Supervised      | 3DDFA [79]                                      | 36.2       | 12.3       | 8.7        | 19.1       |
|                 | KEPLER [33]                                     | 8.8        | 17.3       | 16.2       | 13.9       |
|                 | DLib [29]                                       | 16.8       | 13.8       | 6.1        | 12.2       |
|                 | FAN [4]   | 8.5        | 7.4        | 7.6        | 7.8        |
|                 | Hopenet [51]                                    | 5.1        | 6.9        | 3.3        | 5.1        |
|                 | FSA [70]  | <b>4.2</b> | <b>4.9</b> | <b>2.7</b> | <b>4.0</b> |

Table 1. **Head pose estimation ablation studies and SOTA comparisons.** Average absolute angular error for azimuth, elevation and tilt Euler angles in degrees together with the mean absolute error (MAE) for the BIWI [12] dataset.

perform extensive experimental analysis of our technique on head pose estimation.

**Datasets and evaluation metric** For training, we use the 300W-LP [52] dataset, which combines several in-the-wild face datasets. It contains 122,450 face images with diverse viewpoints, created by fitting a 3D face morphable model [3] to face images and rendering them from various viewpoints. Note that we only use the images from this dataset to train SSV and not their GT viewpoint annotations. We evaluate our framework on the BIWI [12] dataset which contains 15,677 images across 24 sets of video sequences of 20 subjects in a wide variety of viewpoints. We use the MTCNN face detector to detect all faces [74]. We compute average absolute errors (AE) for azimuth, elevation and tilt between the predictions and GT. We also report the mean absolute error (MAE) of these three errors.

**Ablation study** We empirically evaluate the different self-supervised constraints used to train the viewpoint network. Table 1 shows that for head pose estimation, using all the proposed constraints (SSV-Full) results in our best MAE of 6.7°. Removing the image consistency constraint  $\mathcal{L}_{imc}$  leads to an MAE to 7.2° and further removing the symmetry constraint  $\mathcal{L}_{sym}$  results in an MAE of 8.3°. These results demonstrate the usefulness of the generative image consistency and symmetry constraints in our framework.

Additionally, we evaluate the effect of using the paired style and viewpoint loss  $\mathcal{L}_{sv, pair}$  to train the viewpoint-aware synthesis network  $\mathcal{S}$ . We observe that when we train  $\mathcal{S}$  without  $\mathcal{L}_{sv, pair}$ , our viewpoint network (SSV-full model) results in AE values of 7.8° (azimuth), 11.1° (elevation), 4.2° (tilt) and an MAE of 7.7°. This represents a 1° increase from the corresponding MAE value of 6.7°

|            | Method              | Azimuth    | Elevation  | Tilt       | MAE        |
|------------|---------------------|------------|------------|------------|------------|
| Self-Sup   | SSV non-refined     | 6.9        | 9.4        | <b>4.2</b> | 6.8        |
|            | SSV refined on BIWI | <b>4.9</b> | <b>8.5</b> | <b>4.2</b> | <b>5.8</b> |
| Supervised | FSA [70]            | <b>2.8</b> | 4.2        | 3.6        | <b>3.6</b> |
|            | DeepHP [41]         | 5.6        | 5.1        | -          | -          |
|            | RNNFace [17]        | 3.9        | <b>4.0</b> | <b>3.0</b> | <b>3.6</b> |

Table 2. **Improved head pose estimation with fine-tuning.** Average angular error for each of the Euler angles together with mean average error (MAE) on data of 30% held-out sequences of the BIWI [12] dataset and fine-tuning on the remaining 70% without using their annotations. All values are in degrees.

for the SSV-full, where  $\mathcal{S}$  is trained with  $\mathcal{L}_{sv,pair}$  (Table 1, SSV-full). This shows that our paired style and viewpoint loss helps to better train the image synthesis network for the task of viewpoint estimation.

**Comparison with self-supervised methods** Since SSV is a self-supervised viewpoint estimation work, there is no existing work that we can directly compare against. One could also obtain head pose from predicted face landmarks and we compare against recent state-of-the-art self-supervised landmark estimation (LMDIS [76], IMM [24]) and part discovery techniques (SCOPS [22]). We fit a linear regressor that maps the self-supervisedly learned semantic face part centers from SCOPS and landmarks from LMDIS, IMM to five canonical facial landmarks (left-eye center, right-eye center, nose tip and mouth corners). Then we fit an average 3D face model to these facial landmarks with the Perspective-n-Point (PnP) algorithm [36] to estimate head pose. We also quantify HoloGAN’s [42] performance at viewpoint estimation, by training a viewpoint network with images synthesized by it under different input viewpoints (as pseudo GT). Alternatively, we train HoloGAN with an additional viewpoint output and a corresponding additional loss for it. For both these latter approaches, we additionally use viewpoint calibration, similar to SSV. We consider these works as our closest baselines because of their self-supervised training. The MAE results in Table 1 indicate that SSV performs considerably better than all the competing self-supervised methods.

**Comparison with supervised methods** As a reference, we also report the metrics for the recent state-of-the-art fully-supervised methods. Table 1 shows the results for both the keypoint-based [79, 33, 29, 4] and keypoint-free [51, 70] methods. The latter methods learn to directly regress head orientation values from networks. The results indicate that ‘SSV-Full’, despite being purely self-supervised, can obtain comparable results to fully supervised techniques. In addition, we notice that SSV-Full (with MAE 6.7°) outperforms all the keypoint-based supervised methods [79, 33, 29, 4], where FAN [4] has the best MAE of 7.8°.

**Refinement on BIWI dataset** The results reported thus far are with training on the 300W-LP [52] dataset. Following some recent works [70, 41, 17], we use 70% (16) of the image sequences in the BIWI dataset to fine-tune our model. Since our method is self-supervised, we just use images from BIWI without the annotations. We use the remaining 30% (8) image sequences for evaluation. The results of our model along with those of the state-of-the-art supervised models are reported in Table 2. After refinement with the BIWI dataset’s images, the MAE of SSV significantly reduces to 5.8°. This demonstrates that SSV can improve its performance with the availability of images that match the target domain, even without GT annotations. We also show qualitative results of head pose estimation for this refined SSV-Full model in Figure 6(a). It performs robustly to large variations in head pose, identity and expression.

## 5.2. Generalization to Other Object Categories

SSV is not specific to faces and can be used to learn viewpoints of other object categories. To demonstrate its generalization ability, we additionally train and evaluate SSV on the categories of cars, buses and trains.

**Datasets and evaluation metric** Since SSV is completely self-supervised, the training image collection has to be reasonably large to cover all possible object viewpoints while covering diversity in other image aspects such as appearance, lighting etc. For this reason, we leverage large-scale image collections from both the existing datasets and the internet to train our network. For the car category, we use the CompCars [69] dataset, which is a fine-grained car model classification dataset containing 137,000 car images in various viewpoints. For the ‘train’ and ‘bus’ categories, we use the OpenImages [44, 45, 2] dataset which contains about 12,000 images of each of these categories. Additionally, we mine about 30,000 images from Google image search for each category. None of the aforementioned datasets have viewpoint annotations. This also demonstrates the ability of SSV to consume large-scale internet image collections that come without any viewpoint annotations.

We evaluate the performance of the trained SSV model on the test sets of the challenging Pascal3D+ [67] dataset. The images in this dataset have extreme shape, appearance and viewpoint variations. Following [39, 49, 63, 37], we estimate the azimuth, elevation and tilt values, given the GT object location. To compute the error between the predicted and GT viewpoints, we follow the standard geodesic distance  $\Delta(R_{gt}, R_p) = \|\log R_{gt}^T R_p\|_{\mathcal{F}} / \sqrt{2}$  between the predicted rotation matrix  $R_p$  constructed using viewpoint predictions and  $R_{gt}$  constructed using GT viewpoints [37]. Using this distance metric, we report the median geodesic error (Med. Error) for the test set. Additionally, we also compute the percentage of inlier predictions whose error is less than  $\pi/6$  ( $\text{Acc}@ \pi/6$ ).





Figure 6. **Viewpoint estimation results.** We visually show the results of (a) head pose estimation on the BIWI [12] dataset and of viewpoint estimation on the test sets of the (b) car, (c) bus and (d) train categories from the PASCAL3D+ [67] dataset. Solid arrows indicate predicted viewpoints, while the dashed arrows indicate their GT values. Our self-supervised method performs well for a wide range of head poses, identities and facial expressions. It also successfully handles different object appearances and lighting conditions from the car, bus and train categories. We show additional results in the supplementary material.

**Baselines** For head pose estimation, we compared with self-supervised landmark [76, 22, 24] discovery techniques coupled with the PnP algorithm for head pose estimation by fitting them to an average 3D face. For objects like cars with full 360° azimuth rotation, we notice that the landmarks produced by SCOPS [22] and LMDIS [76] cannot be used for reasonable viewpoint estimates. This is because SCOPS is primarily a self supervised part segmentation framework which does not distinguish between front and rear parts of the car. Since the keypoints we compute are the centers of part segments, the resulting keypoints cannot distinguish such parts. LMDIS on the other hand produces keypoints only for the side profiles of cars. Hence, we use another baseline technique for comparisons on cars, trains and buses. Following the insights from [22, 60] that features learned by image classification networks are equivariant to object rotation, we learn a linear regressor that maps the Conv5 features of a pre-trained VGG network [54] to the viewpoint of an object. To train this baseline, we use the VGG image features and the GT viewpoint annotations in the Pascal3D+ training dataset [67]. We use the same Pascal3D+ annotations used to calibrate SSV’s predicted viewpoints to GT canonical viewpoint axes. We consider this as a self-supervised baseline since we are not using GT annotations for feature learning but only to map the features to viewpoint predictions. We refer to this baseline as VGG-View. As an additional baseline, we train HoloGAN [42] with an additional viewpoint output and a corresponding loss for it. The viewpoint predictions are calibrated, similar to SSV.

**Comparisons** We compare SSV to our baselines and also to several state-of-the-art supervised viewpoint estimation methods on the Pascal3D+ test dataset. Table 3 indicates that SSV significantly outperforms the baselines. With respect to supervised methods, SSV performs comparably to Tulsiani *et al.* [63] and Mahendran *et al.* [39] in terms of Median error. Interestingly for the ‘train’ category, SSV performs even better than supervised methods. These results demonstrate the general applicability of SSV for viewpoint learning on different object categories. We show some

|            | Method                       | Car         | Bus        | Train      |
|------------|------------------------------|-------------|------------|------------|
| Self-Sup   | VGG-View                     | 34.2        | 19.0       | 9.4        |
|            | HoloGAN [42] with $v$        | 16.3        | 14.2       | 9.7        |
|            | SSV-Full                     | <b>10.1</b> | <b>9.0</b> | <b>5.3</b> |
| Supervised | Tulsiani <i>et al.</i> [63]  | 9.1         | 5.8        | 8.7        |
|            | Mahendran <i>et al.</i> [39] | 8.1         | 4.3        | 7.3        |
|            | Liao <i>et al.</i> [37]      | 5.2         | 3.4        | <b>6.1</b> |
|            | Grabner <i>et al.</i> [16]   | <b>5.1</b>  | <b>3.3</b> | 6.7        |

Table 3. **Generalization to other object categories, median error.** We show the median geodesic errors (in degrees) for the car, bus and train categories.

|            | Method                       | Car         | Bus         | Train       |
|------------|------------------------------|-------------|-------------|-------------|
| Self-sup   | VGG-View                     | 0.43        | 0.69        | 0.82        |
|            | HoloGAN [42] with $v$        | 0.52        | 0.73        | 0.81        |
|            | SSV-Full                     | <b>0.67</b> | <b>0.82</b> | <b>0.96</b> |
| Supervised | Tulsiani <i>et al.</i> [63]  | 0.89        | <b>0.98</b> | 0.80        |
|            | Mahendran <i>et al.</i> [39] | -           | -           | -           |
|            | Liao <i>et al.</i> [37]      | <b>0.93</b> | 0.97        | <b>0.84</b> |
|            | Grabner <i>et al.</i> [16]   | <b>0.93</b> | 0.97        | 0.80        |

Table 4. **Generalization to other object categories, inlier count.** We show the percentage of images with geodesic error less than  $\pi/6$  for the car, bus and train categories.

qualitative results for these categories in Figure 6(b)-(d).

## 6. Conclusions

In this work we investigate the largely unexplored problem of learning viewpoint estimation in a self-supervised manner from collections of un-annotated object images. We design a viewpoint learning framework that receives supervision from a viewpoint-aware synthesis network; and from additional symmetry and adversarial constraints. We further supervise our synthesis network with additional losses to better control its image synthesis process. We show that our technique outperforms existing self-supervised techniques and performs competitively to fully-supervised ones on several object categories like faces, cars, buses and trains.



## References

- [1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 4, 5
- [2] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019. 7
- [3] Volker Blanz, Thomas Vetter, et al. A morphable model for the synthesis of 3d faces. In *Siggraph*, 1999. 6
- [4] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? In *CVPR*, 2017. 2, 6, 7
- [5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 6
- [6] Feng-Ju Chang, Anh Tuan Tran, Tal Hassner, Iacopo Masi, Ram Nevatia, and Gerard Medioni. Faceposenet: Making a case for landmark-free face alignment. In *CVPR*, 2017. 2, 6
- [7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016. 3, 5
- [8] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, 2019. 1
- [9] Edo Collins, Radhakrishna Achanta, and Sabine Susstrunk. Deep feature factorization for concept discovery. In *ECCV*, 2018. 2
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 4
- [11] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017. 6
- [12] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *IJCV*, 2013. 1, 2, 6, 7, 8
- [13] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *ECCV*, 2018. 2
- [14] Renaud Marlet Francisco Massa and Mathieu Aubry. Crafting a multi-task cnn for viewpoint estimation. In *BMVC*, 2016. 1, 2
- [15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 4
- [16] Alexander Grabner, Peter M Roth, and Vincent Lepetit. 3d pose estimation and 3d model retrieval for objects in the wild. In *CVPR*, 2018. 1, 2, 8
- [17] Jinwei Gu, Xiaodong Yang, Shalini De Mello, and Jan Kautz. Dynamic facial analysis: From bayesian filtering to recurrent neural network. In *CVPR*, 2017. 1, 2, 6, 7
- [18] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *BMVC*, 2018. 2
- [19] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *ICANN*, 2011. 5
- [20] Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. Improving landmark localization with semi-supervised learning. In *CVPR*, 2018. 2
- [21] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *CVPR*, 2017. 5, 12, 14
- [22] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019. 2, 6, 7, 8
- [23] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*, 2018. 2
- [24] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *NeurIPS*, 2018. 2, 6, 7
- [25] Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter V Gehler. The informed sampler: A discriminative approach to bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32–44, 2015. 3
- [26] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2
- [27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3, 5
- [28] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018. 2, 3
- [29] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014. 6, 7
- [30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 12
- [31] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *ICCV*, 2019. 2
- [32] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NeurIPS*, 2015. 5
- [33] Amit Kumar, Azadeh Alavi, and Rama Chellappa. Kepler: keypoint and pose estimation of unconstrained faces by learning efficient h-cnn regressors. In *FG*, 2017. 2, 6, 7
- [34] Abhijit Kundu, Yin Li, and James M Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 1, 2, 3
- [35] K L Navaneet, Priyanka Mandikal, Varun Jampani, and Venkatesh Babu. Differ: Moving beyond 3d reconstruction with differentiable feature rendering. In *CVPRW*, 2019. 2
- [36] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnnp: An accurate o(n) solution to the pnp problem. *IJCV*, 2009. 7
- [37] Shuai Liao, Efstratios Gavves, and Cees G. M. Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In *CVPR*, 2019. 1, 2, 3, 4, 7, 8

- [38] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019. 3
- [39] Siddharth Mahendran, Haider Ali, and René Vidal. 3d pose regression using convolutional neural networks. In *CVPR*, 2017. 1, 2, 3, 7, 8
- [40] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 2
- [41] Sankha S Mukherjee and Neil Martin Robertson. Deep head pose: Gaze-direction estimation in multimodal video. *ACM MM*, 2015. 7
- [42] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, 2019. 3, 5, 6, 7, 8
- [43] David Novotny, Diane Larlus, and Andrea Vedaldi. Learning 3d object categories by looking around them. In *CVPR*, pages 5218–5227, 2017. 2
- [44] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. We don’t need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*, 2016. 7
- [45] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *CVPR*, 2017. 7
- [46] Seonwook Park12, Shalini De Mello, Pavlo Molchanov, Umar Iqbal, Otmar Hilliges, and Jan Kautz. Few-shot adaptive gaze estimation. In *ICCV*, 2019. 5
- [47] 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 *NeurIPS*, 2017. 6, 12
- [48] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 2
- [49] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *ECCV*, 2018. 2, 7
- [50] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In *ECCV*, 2018. 5
- [51] Nataniel Ruiz, Eunji Chong, and James M Rehg. Fine-grained head pose estimation without keypoints. In *CVPRW*, 2018. 2, 6, 7
- [52] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *ICCVW*, 2013. 6, 7
- [53] Mihir Sahasrabudhe, Zhixin Shu, Edward Bartram, Riza Alp Guler, Dimitris Samaras, and Iasonas Kokkinos. Lifting autoencoders: Unsupervised learning of a fully-disentangled 3d morphable model using deep non-rigid structure from motion. In *ICCVW*, 2019. 2
- [54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4, 8
- [55] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019. 5
- [56] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *CVPR*, 2015. 1, 2
- [57] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013. 2, 6
- [58] Supasorn Suwajanakorn, Noah Snaveley, Jonathan Tompson, and Mohammad Norouzi. Discovery of latent 3D key-points via end-to-end geometric reasoning. In *NeurIPS*, 2018. 2
- [59] Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeonwoo Kim, Patrick Pérez, and Christian Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *CVPR*, 2018. 2
- [60] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *ICCV*, 2017. 2, 6, 8
- [61] Luan Tran and Xiaoming Liu. Nonlinear 3d face morphable model. In *CVPR*, 2018. 2
- [62] Hung-Yu Tseng, Shalini De Mello, Jonathan Tremblay, Sifei Liu, Stan Birchfield, Ming-Hsuan Yang, and Jan Kautz. Few-shot viewpoint estimation. In *BMVC*, 2019. 2
- [63] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015. 1, 2, 7, 8
- [64] 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. 12
- [65] Olivia Wiles and Andrew Zisserman. Silnet: Single- and multi-view reconstruction by learning from silhouettes. In *BMVC*, 2017. 2
- [66] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. ObjectNet3D: A large scale database for 3D object recognition. In *ECCV*, 2016. 2
- [67] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014. 1, 2, 7, 8
- [68] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018. 1
- [69] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015. 7
- [70] Tsun-Yi Yang, Yi-Ting Chen, Yen-Yu Lin, and Yung-Yu Chuang. Fsa-net: Learning fine-grained structure aggregation for head pose estimation from a single image. In *CVPR*, 2019. 1, 2, 6, 7
- [71] Tsun-Yi Yang, Yi-Hsuan Huang, Yen-Yu Lin, Pi-Cheng Hsiu, and Yung-Yu Chuang. Ssr-net: A compact soft stage-wise regression network for age estimation. *IJCAI*, 2018. 2, 6

- [72] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *NeurIPS*, 2018. 3
- [73] Alan Yuille and Daniel Kersten. Vision as bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 2006. 3
- [74] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *SPL*, 2016. 6
- [75] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 6
- [76] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018. 2, 6, 7, 8
- [77] Xingyi Zhou, Arjun Karapur, Linjie Luo, and Qixing Huang. Starmap for category-agnostic keypoint and viewpoint estimation. In *ECCV*, 2018. 1, 2
- [78] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: image generation with disentangled 3d representations. In *NeurIPS*, 2018. 5
- [79] Xiangyu Zhu, Xiaoming Liu, Zhen Lei, and Stan Z Li. Face alignment in full pose range: A 3d total solution. *PAMI*, 2017. 1, 2, 6, 7



## Appendix

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

### A. Network Architecture

The network architectures of the viewpoint and synthesis networks are detailed in tables 5 and 6, 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 [64] in the viewpoint network. For the synthesis network, the size of the style code  $z_s$  is 128 for faces and 200 for the other objects (car, bus and train).  $z_s$  is mapped to affine transformation parameters ( $\gamma(z_s), \sigma(z_s)$ ), which are in turn used by adaptive instance normalization (AdaIN) [21] to control the style of the synthesized images.

### B. Training Details

SSV is implemented in Pytorch [47]. 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 [30] 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.

### C. 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.

### D. Visual Results

In figures 7, 8, 10, 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 |
|-----------------|-----------------------------|-------------|--------|------------|---------------|------------------|
|                 | Conv                        | 1x1         | 1      | LReLU      | -             | 128x128x128      |
|                 | Conv2D                      | 3x3         | 1      | LReLU      | Instance Norm | 128X128x256      |
|                 | Conv2D                      | 3x3         | 1      | LReLU      | Instance Norm | 128X128x256      |
| Backbone Layers | 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 - $ \hat{a} $            | -           | -      | -          | -             | 2                |
|                 | FC - $\text{sign}(\hat{a})$ | -           | -      | -          | -             | 4                |
| Elevation       | FC                          | -           | -      | LReLU      | -             | 256              |
|                 | FC - $ \hat{e} $            | -           | -      | -          | -             | 2                |
|                 | FC - $\text{sign}(\hat{e})$ | -           | -      | -          | -             | 4                |
| Tilt            | FC                          | -           | -      | LReLU      | -             | 256              |
|                 | FC - $ \hat{t} $            | -           | -      | -          | -             | 2                |
|                 | FC - $\text{sign}(\hat{t})$ | -           | -      | -          | -             | 4                |

Table 5. **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.

|                   | Layer                   | Kernel Size | stride | Activation | Normalization | Output Dimension |
|-------------------|-------------------------|-------------|--------|------------|---------------|------------------|
|                   | Input - 3D Code         | -           | -      | -          | -             | 4x4x4x512        |
| Stylized 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     |
| Project           | Conv 3D                 | 3x3         | 1      | LReLU      | -             | 16x16x16x128     |
|                   | Conv 3D                 | 3x3         | 1      | LReLU      | -             | 16x16x16x64      |
| Project           | Collapse                | -           | -      | -          | -             | 16x16x(16.64)    |
|                   | Conv                    | 3x3         | 1      | LReLU      | -             | 16x16x1024       |
| Stylized 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 6. **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 [21]) 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.



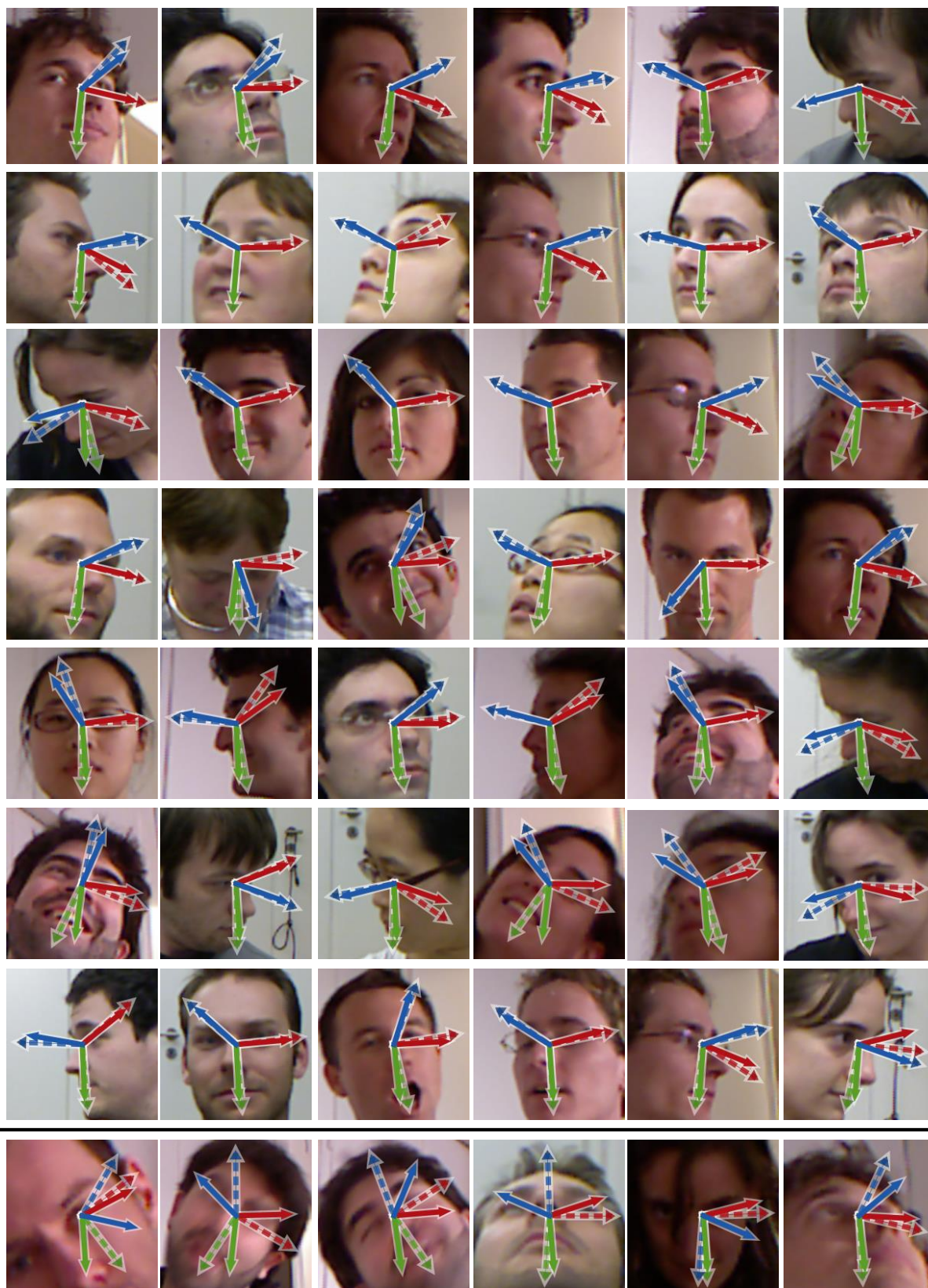


Figure 7. **Viewpoint estimation results for the face category.** SSV predicts reliable viewpoints for a variety of face poses with large variations in azimuth, elevation and tilt. The last row (below the black line) shows some erroneous cases where the faces are partially detected by the face detector or there are extreme elevation angles.



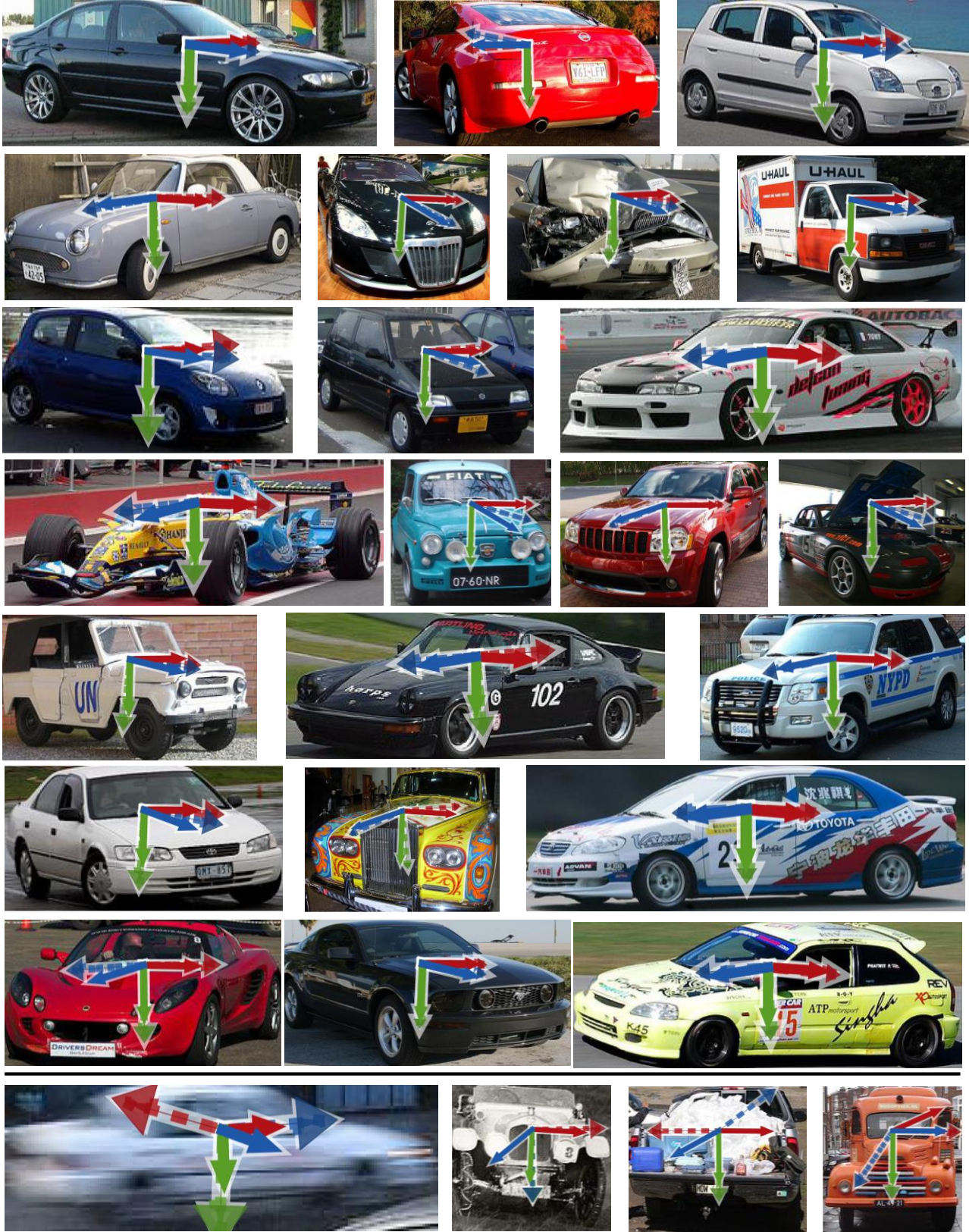


Figure 8. **Viewpoint estimation results for the car category.** SSV predicts reliable viewpoints for a variety of objects with large variations in azimuth, elevation and tilt. It generalizes to car models like race cars and formula-1 cars, which are not seen by SSV during training. The last row (below 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 9. **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 10. **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.