Supplementary Material: Progressive Transformation Learning for Leveraging Virtual Images in Training

1. Gaussian Discriminant Analysis for Modeling Representation Space of Detector

In this section, we describe modeling the representation space of a general object detector by fitting a multivariate Gaussian distribution. We denote the random variable of the input and its label of a linear classifier as $\mathbf{x} \in \mathcal{X}$ and $y = \{y_c\}_{c=1,\cdots,C} \in \mathcal{Y}, y_c = \{0, 1\}$, respectively. Then, the posterior distribution defined by the linear classifier whose output formed by the sigmoid function can be expressed as follows:

$$\begin{aligned} P(y_c = 1|\mathbf{x}) &= \frac{1}{1 + \exp\left(-w_c\mathbf{x}-b_c\right)} \\ &= \frac{\exp\left(w_c\mathbf{x} + b_c\right)}{\exp\left(w_c\mathbf{x} + b_c\right) + 1}, \end{aligned} \tag{1}$$

where w_c and b_c are weights and bias of the linear classifier for a category c, respectively.

Gaussian Discriminant Analysis (GDA) models the posterior distribution of the classifier by assuming that the class conditional distribution ($P(\mathbf{x}|y)$) and the class prior distribution ($P(y)$) follow the multivariate Gaussian and the Bernoulli distributions, respectively, as follows:

$$\begin{aligned} P(\mathbf{x}|y_c = 0) &= \mathcal{N}(\mu_0, \Sigma_0), \\ P(\mathbf{x}|y_c = 1) &= \mathcal{N}(\mu_1, \Sigma_1), \\ P(y_c = 0) &= \beta_0/\left(\beta_0 + \beta_1\right), \\ P(y_c = 1) &= \beta_1/\left(\beta_0 + \beta_1\right), \end{aligned} \tag{2}$$

where $\mu_{\{0,1\}}$ and $\Sigma_{\{0,1\}}$ are the mean and covariance of the multivariate Gaussian distribution, and $\beta_{\{0,1\}}$ is the unnormalized prior for the category c and the background.

For the special case of GDA where all categories share the same covariance matrix (i.e., $\Sigma_0 = \Sigma_1 = \Sigma_c$), known as Linear Discriminant Analysis (LDA), the posterior distribution ($P(y_c|\mathbf{x})$) can be expressed with $P(\mathbf{x}|y_c)$ and $P(y_c)$ as follows:

$$\begin{aligned} &P(y_c = 1|\mathbf{x}) \\ &= \frac{P(y_c = 1)P(\mathbf{x}|y_c = 1)}{P(y_c = 0)P(\mathbf{x}|y_c = 0) + P(y_c = 1)P(\mathbf{x}|y_c = 1)} \\ &= \frac{\exp\Big(\left(\mu_1-\mu_0\right)^\top \Sigma_c^{-1}\mathbf{x} \ldots \right. \\ &\qquad \left. -\frac{1}{2}\mu_1^\top\Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top\Sigma_c^{-1}\mu_0 + \log\beta_1/\beta_0\Big)}{\exp\Big(\left(\mu_1-\mu_0\right)^\top \Sigma_c^{-1}\mathbf{x} \ldots \right. \\ &\qquad \left. -\frac{1}{2}\mu_1^\top\Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top\Sigma_c^{-1}\mu_0 + \log\beta_1/\beta_0\Big) + 1}. \end{aligned} \tag{3}$$

Note that the quadratic term is canceled out since the shared covariance matrix is used. The posterior distribution derived by GDA in eq. 3 then becomes equivalent to the posterior distribution of the linear classifier with the sigmoid function in eq. 1 when $w_c = \left(\mu_1-\mu_0\right)^\top \Sigma_c^{-1}$ and $b_c = -\frac{1}{2}\mu_1^\top\Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top\Sigma_c^{-1}\mu_0 + \log\beta_1/\beta_0$. This implies that the representation space formed by $\mathbf{x}$ can be modeled by a multivariate Gaussian distribution.

Based on the above derivation, if $\mathbf{x}$ is the output of the penultimate layer of an object detector for a region proposal, and a linear classifier defined by w_c and b_c is the last layer of the object detector, it can be said that the representation space of the object detector for a category c can be modeled with a multivariate Gaussian distribution. In other words, the representation space for a category c can be represented by two parameters μ_1 (i.e., μ_c) and Σ_c of the multivariate Gaussian distribution.

Discussion. The sigmoid function can be viewed as a special case of the softmax function defined for a single category as both functions take the form of an exponential term for the category-of-interest normalized by the sum of exponential terms for all considered categories. Therefore, it is straightforward to derive the modeling for the sigmoid-based detector from the previous work by [4], who shows that the softmax-based classifier can be modeled with a multivariate Gaussian distribution in the representation space. However, our derivation is still meaningful in that it extends the applicability of an existing modeling limited to

(a) Detector training

config	baseline	pretrain-finetune		naive merge	PTL
		pretrain	finetune		
optimizer	SGD				
momentum	0.9				
weight decay	0.0001				
base lr	0.001				
lr schedule	multi-step lr				
gamma	0.1				
warmup iter.	1000				
total iter.	6000	6000	600	6000	6000
steps	5000	5000	500	5000	5000
batch size	16				
filter empty annot.	False				
box threshold	10				
aspect ratio grouping	False				

(b) Generator training

config	naive merge w/ transform	PTL
optimizer	Adam	
momentum	β_1, β_2 = 0.5, 0.999	
lr	0.0002	
total epochs	80	100
batch size	8	
load size	256	
preprocess	None	

Table 1. **Training settings.**

a certain type of classifier (i.e., based on softmax) to general object detectors (i.e., based on sigmoid). Most object detectors, especially one-stage detectors, generally use the sigmoid function, which does not consider other categories when calculating the model output for a certain category, since more than one category can be active on a single output.

2. Implementation Details

Multi-scale training. We apply the multi-scale training strategy when preparing input images, in addition to the 5-level multi-scaling property provided by the detector's FPN module, to train the detector. The goal is to make the detector more robust to the variations of human size in the images. Since the real and virtual datasets have widely varying human sizes in the images, we apply different scaling factors for each dataset to share similar human sizes after image rescaling. Specifically, for the real dataset, the input image is resized by one of the scaling factors randomly selected from $\{768, 800, 832, 864\}$ for the short side, in which the long side is constrained not to be larger than 1440, for every training iteration. For the virtual dataset, the scaling factors are $\{128, 256, 384, 512\}$ for the short side and 512 for the long side.

Network architecture. To obtain a generator, we adopt CycleGAN, where two generators and two discriminators are involved during model training. For each generator, we use a 24-layer UNet-like architecture (i.e., the `resnet_9blocks` generator in the official repository of CycleGAN [2]), which contains nine 2-layer residual modules in the middle of the architecture where the encoder and decoder are connected. For each discriminator, a 5-layer fully convolutional network (i.e., the `basic` discriminator in the official repository of CycleGAN [2]) is used.

For the detector, we adopt the official RetinaNet architecture implemented in Detectron2 [6] with few modifications. First of all, the feature dimension of RetinaNet's classification subnet and box regression subnet are decreased from 256 to 32 since we are dealing with single category (i.e., human) instead of multiple categories (e.g., 80 categories for detectors trained on the MS COCO dataset). In addition, we switch the activation function used in these two subnets from ReLU to LeakyReLU to avoid the singular covariance matrix problem, which may occur when calculating the Mahalanobis distance, occasionally triggered by the dying ReLU problem. Finally, the kernel size of the last convolutional layer in the classification subnet, the layer just in front of the sigmoid layer, is reduced from 3×3 to 1×1 so that each output prediction is associated with only one feature vector in the feature representation space. Here, ResNet50 is used as the backbone.

Training details. The settings used for training the human detector through our method and the other baseline methods are listed in Table 1a. Without further specification, we follow all the settings and initialization strategies defined by the original RetinaNet training [5]. Unlike other model training, using fewer iterations for fine-tuning when adopting the pretrain-finetune method is common because training converges faster. Although fine-tuning usually uses a lower learning rate than pre-training, we use the same learning rate (i.e., 0.001) for both stages because we also fine-tune the ImageNet-pretrained backbone on the virtual images in the pre-training stage.

The settings used for training the generator through the baseline method using transformation (i.e., 'naive merge w/ transformation') and our method are listed in Table 1b. These settings and initialization strategies are taken from the original CycleGAN training [7]. 'Naive merge w/ transform' used fewer training epochs than PTL (80 vs 100) because all images in the virtual set are used for training. In general, as the size of the dataset increases, the number of

Figure 1. **Sample images from each dataset.** For the Archangel-Synthetic dataset, metadata for each image is shown to the right of the image.

epochs required for training decreases.

Datasets. In this section, we provide the details of the three real UAV-based datasets, VisDrone [8], Okutama-Action [3], and ICG [1]), and the virtual UAV-based dataset, *Archangel-Synthetic*, used in this paper. Sample images for each dataset are shown in Figure 1.

VisDrone has four tracks (i.e., object detection (DET), video object detection (VID), single object tracking (SOT), and multi-object tracking (MOT)) with a separate dataset for each track. We use the dataset from the DET track and focus on detecting the person and the pedestrian categories among the ten object categories covered by the track. The VisDrone dataset in the DET track consists of 10,209 images, where 6,471 images are used in the `training` set, 548 images are used in the `validation` set, 1,580 images are used in the `test-challenge` set, and 1,610 images

are used in the `test-dev` set. We use the `training` set for model training and the `test-dev` set for model testing. The maximal resolution of images in the VisDrone dataset is 2000×1500.

Okutama-Action was originally created for human action detection from the aerial view. Although the main task associated with this dataset is human action detection, the dataset also includes a sub-task of pedestrian detection. To acquire images from various aerial views, a drone with an embedded camera flew freely at some altitudes between $10m$ and $45m$, and the camera angle was set at 45 or 90 degrees. The Okutama-Action dataset contains 43 video sequences in 4K resolution (i.e., 3840×2160), where 33 video sequences are used for the `train-val` set and the remaining 10 video sequences are used for the `test` set. Since adjacent frames in the video sequence are very similar, we use every tenth frame in both the `train-val` set and the `test` set. Our model is trained on the `train-val` set and tested with the `test` set.

ICG was collected for studying semantic understanding of urban scenes. Additionally, the ICG dataset also provides information, such as ground-truth human bounding boxes, for the human detection task. Images in the ICG dataset were captured from a camera located at some altitudes between 5-50m above the ground at a resolution of 6000×4000. The ICG dataset provides a `training` set of 400 images for the human detection task, of which we use the first 320 images for training and the remaining 80 images for testing.

Archangel-Synthetic is one of the three sub-datasets included in the Archangel dataset, along with *Archangel-Real* and *Archangel-Mannequin*. The Archangel dataset is a hybrid UAV-based dataset captured with similar imaging conditions in real and synthetic domains. An important property of the Archangel dataset that sets it apart from other datasets is that it provides metadata about the camera positions in terms of UAV altitudes and radii of the rotation circles for each image. The Archangel-Synthetic dataset was generated by using the Unity game engine. The dataset includes eight virtual characters in three different poses captured with camera viewing angles ranging from 0° to 358° in increments of 2°, UAV altitudes and rotation circle radii from $5m$ to $80m$ in increments of $5m$, and four different sun angles. The total number of images in the Archangel-Synthetic dataset is 4.4M. Considering the significant difference in dataset size between the Archangel-Synthetic dataset with the other real datasets, a small subset of the entire dataset (17.6K) was used as the virtual image set in our experiments. The size of images in the Archangel-Synthetic dataset is 512×512.

method	total	detail
baseline	40	
pretrain-finetune	21	4/17 (pretrain / finetune)
naive merge	17	
w/ transform	2,777	2,760/17 (GAN train / Dtr train)
PTL	600	20/iter. (domain gap calc. on virtual set)
		40/36/32/28/25/22 (Dtr train, $\sim 6^{th}$ iter.)
		28/41/56/69/83 (GAN train, $\sim 5^{th}$ iter.)

Table 2. **Wall-clock training time in $mins$ (VisDrone, 20-shot).** GeForce RTX 2080 Ti GPUs are used for this comparison.

metric	Vis	Oku	ICG
Euclidean	5.28 /1.52	28.80 /6.83	26.00 /7.06
Mahalanobis	**6.83 /1.94**	**30.72 /7.45**	**26.86 /7.22**

Table 3. **Comparison of various distance metrics (VisDrone, 20-shot)**.

3. Additional Analyses

Training time comparison. Training times for PTL and the baselines are shown in Tab 2. For PTL, it is observed that the detector training time (i.e., 'Dtr train' in the table) decreases as training progresses because the number of virtual images (i.e., *Archangel-Synthetic*) with a usually smaller image size than real images (i.e., *VisDrone*) increases during training. Detector training in PTL uses the same number of iterations regardless of the PTL iteration. In contrast, the CycleGAN training time (i.e., 'GAN train' in the table) gradually increases due to the increased number of virtual images.

PTL is slow due to CycleGAN training, but it is still much faster than 'naive merge w/ transform' (i.e., the baseline using transformation) because only a subset of virtual images is used instead of the full set for each PTL iteration. PTL can lead to scalabillity issues due to its training time when a large number of virtual images are used with longer iterations. To address this issue, we can reduce the CycleGAN training time for each PTL iteration by fine-tuning the model trained in the previous PTL iteration.

Analysis of domain gap measurement. To investigate the effect of using the Mahalanobis distance to measure the domain gap, we compare it to other metric available under the assumption that the representation for a certain category in a real dataset is modeled by a multivariate Gaussian distribution. Specifically, we use Euclidean distance, which depends only on the mean but not on the covariance of the distribution. As shown in Table 3, using Mahalanobis distance consistently presents better accuracy in both the in-domain and cross-domain setups.

Analysis of transformation candidate selection. To investigate the effect of using weighted random sampling to select transformation candidates, we compare it with a variety of other selection strategies. We carry out experiments

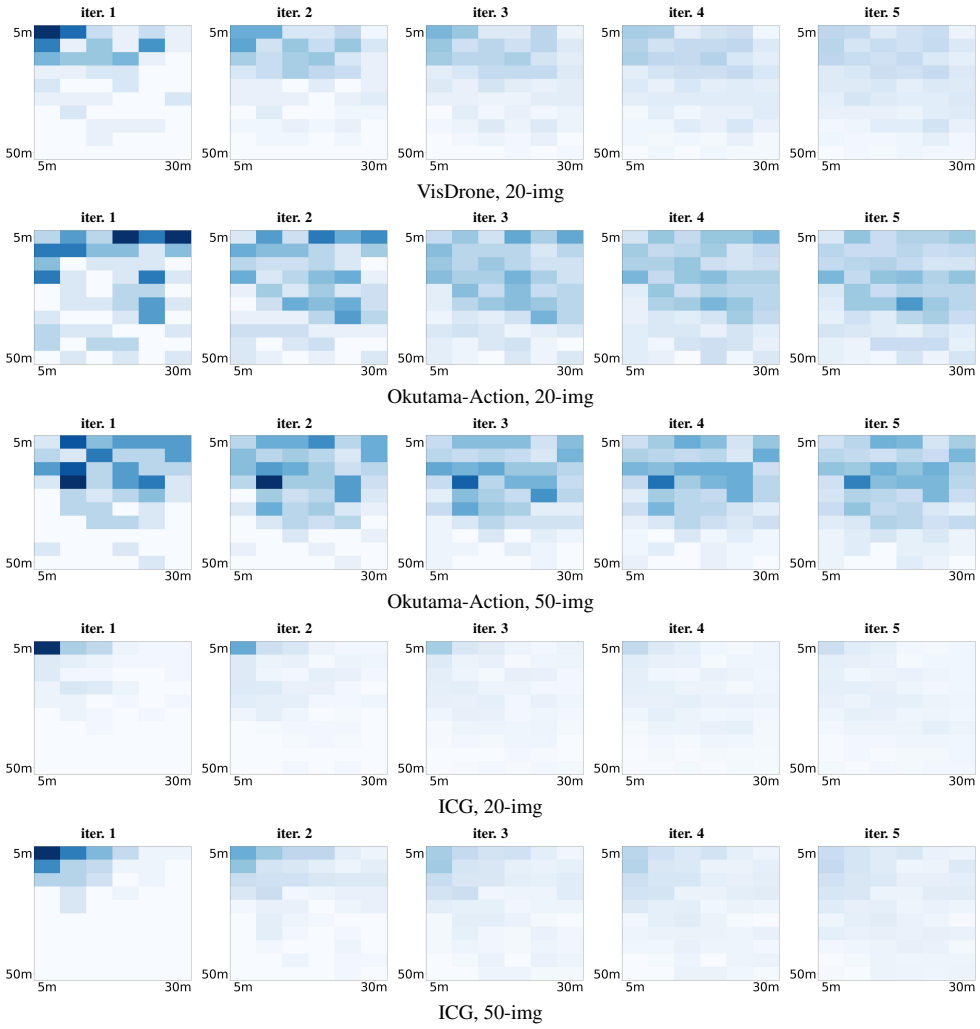


Figure 2. **Accumulated distributions of transformation candidates with respect to camera locations in more setups.** This figure shows the distributions in other five experimental setups except for the setup (i.e., VisDrone, 50-img) shown in Figure 3 of the main manuscript. The x and y axes indicate altitude and rotation circle radius from the target human.

test	close 100	mid 100	far 100	our
Vis	9.11/ 2.71	9.11/ 2.89	8.64/ 2.53	**9.38/ 2.94**
Oku	39.57/11.05	**43.31/11.55**	41.98/11.32	42.39/11.47
ICG	29.66/ 7.40	**34.98/ 8.97**	31.24/ 8.19	30.01/ 7.36

Table 4. **Various transformation candidate selections (VisDrone, 50-shot).**

selecting 100 virtual images with three different ranges of domain gaps ({close, mid, far}) instead of using weighted random sampling ('our' in the table) for each PTL iteration in Tab 4. We observe that our selection strategy is the best in the in-domain setup without sacrificing accuracy much in the cross-domain setup.

Further analysis of the properties of progressive learning. In this section, we show the distributions of transformation candidates in relation to the camera position (in Figure 2) and the domain gap (in Figure 3) for other five cases not shown in Figure 3 of the main manuscript. We intend to show that the analysis described in the main manuscript can be also applied to these cases.

For the distributions of transformation candidates with respect to camera locations (Figure 2), the observation shown in the main manuscript is also applied to the other five cases. That is, as PTL progresses, the camera locations of virtual images included in the training set are gradually spread over the entire area. Therefore, the validity of the transformation candidate selection process of PTL extends to all the experimental setups considered in this paper.

Note that the distributions of humans in the real training set with respect to camera locations is likely to be similar to the distributions of transformation candidates with

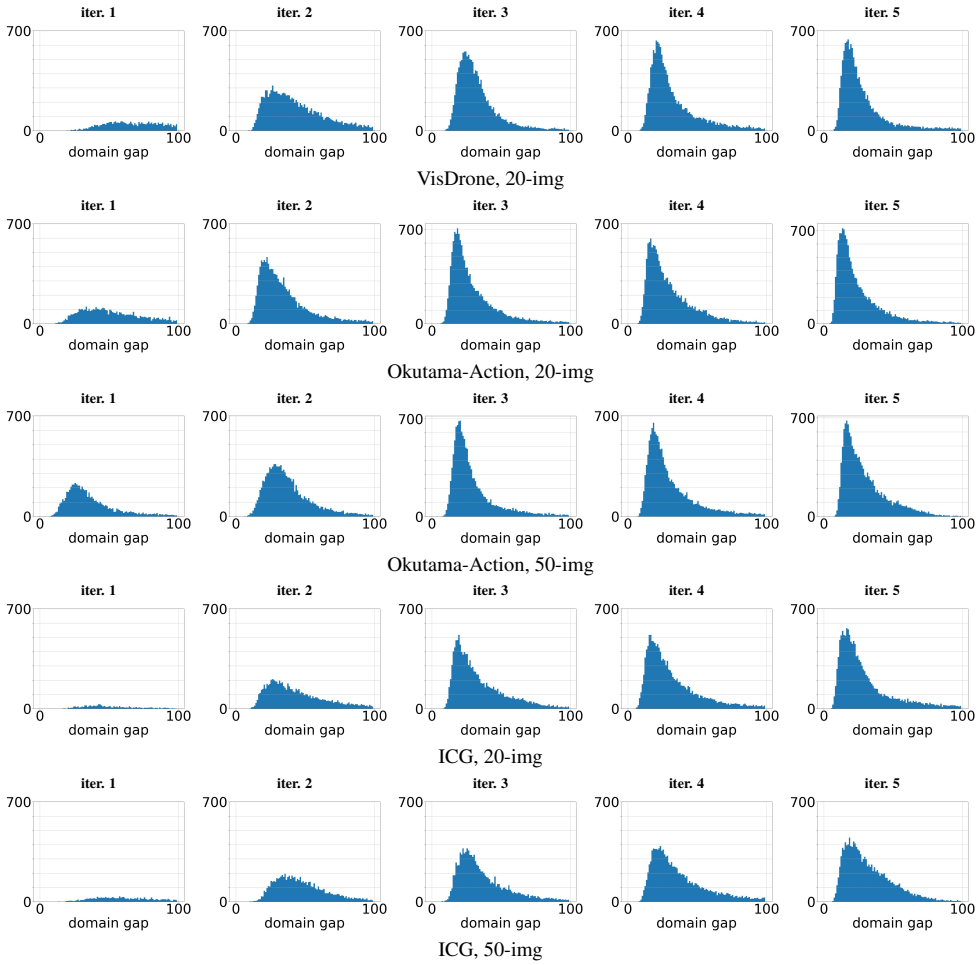


Figure 3. **Distributions of virtual images with respect to the domain gap in more setups.** This figure shows the domain gap distributions of virtual images in other five experimental setups except for the setup (i.e., VisDrone, 50-img) shown in Figure 3 of the main manuscript. The x axis represents the domain gap and the y axis represents the corresponding number of virtual images.

respect to camera locations at the first PTL iteration as a virtual image with a smaller domain gap is selected with a higher probability. Accordingly, we speculate that humans in the real training set were captured from various camera locations in the Okutama-action dataset. In contrast, in the other two datasets, most of them were taken at some similar ranges. In addition, in the ICG dataset, the camera locations where most images were taken are in the close range, which might be the reason why diversifying camera locations through PTL does not significantly improve accuracy in the cross-domain setup.

For the distributions of virtual images with respect to the domain gap (Figure 3), the observation mentioned in the main manuscript that the distribution becomes narrower and smaller as PTL progresses is still perceived in these settings. However, the speed of this distribution change is particularly slow for the ICG dataset compared to the other datasets, as shown in Figure 3. This observation also implies that the ICG dataset has very different characteristics compared to the other two datasets.

Qualitative analysis of transformation. Figures 4, 5, 6, 7, 8, and 9 show several samples of transformed virtual images included in the training set using methods with virtual2real transformation (i.e., PTL and 'naive merge w/ transformation') in the six experimental setups. Since the trends seen in these examples are similar to Figure 5 in the main manuscript, a qualitative analysis of the superiority of PTL over 'naive merge w/ transform' and our claim that the domain gap between virtual images and real images should be considered when training the virtual2real transformation generator can also be applied to these experimental setups.

Naive merge w/ transform PTL

Figure 4. **Sample Virtual2Real transformation output (VisDrone, 20-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

Naive merge w/ transform PTL

Figure 5. **Sample Virtual2Real transformation output (VisDrone, 50-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

References

[1] Aerial semantic segmentation drone dataset. `http://dronedataset.icg.tugraz.at`.

[2] Cyclegan and pix2pix in pytorch. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

[3] Mohammadamin Barekatain, Miquel Martí, Hsueh-Fu Shih, Samuel Murray, Kotaro Nakayama, Yutaka Matsuo, and Helmut Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. In *Proc. CVPR Workshop*, 2017.

[4] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proc. NeurIPS*, 2018.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. ICCV*, 2017.

[6] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, 2017.

Naive merge w/ transform PTL

Figure 6. **Sample Virtual2Real transformation output (Okutama-Action, 20-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

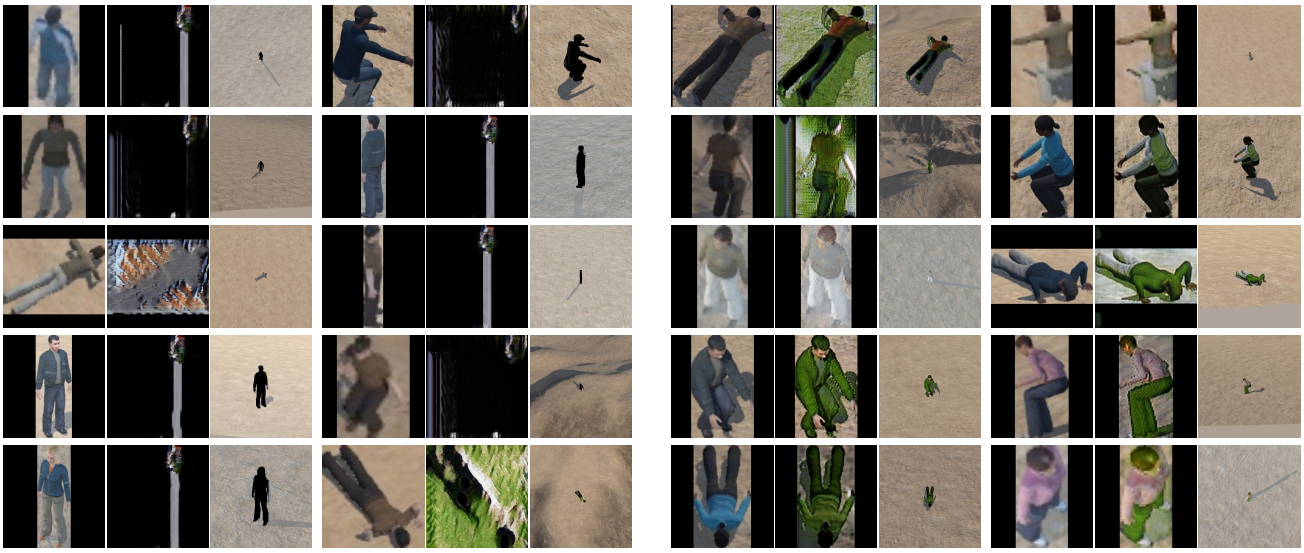
Naive merge w/ transform PTL

Figure 7. **Sample Virtual2Real transformation output (Okutama-Action, 50-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

[8] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):7380–7399, Nov 2022.

Naive merge w/ transform PTL

Figure 8. **Sample Virtual2Real transformation output (ICG, 20-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



Naive merge w/ transform PTL

Figure 9. **Sample Virtual2Real transformation output (ICG, 50-shot).** Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).