Supplementary materials - Pose Augmentation: Class-agnostic Object Pose Transformation for Object Recognition

Yunhao Ge[1 [0000−0002−8110−9280]], Jiaping Zhao[2], and Laurent Itti[1 [0000−0002−0168−2977]]

[1] University of Southern California, Los Angeles, USA {yunhaoge,itti}@usc.edu
[2] Google Research, Los Angeles, USA jiapingz@google.com
https://github.com/gyhandy/Pose-Augmentation

0.1 Network structure and Training details

Continue pose transformation demo at https://youtu.be/uZS5JKMBDZA

We mentioned the high level network architecture and training instructions in main paper Section 3.2. Here are the details.

Network structure OPT-Net consists of 3 modules: eliminate-add structure generator G (which consists of G_{elim} and G_{add}) (shown in Table 1), discriminator D (shown in Table 2), and pose-eliminate module P_{elim} (shown in Table 3). Here are some notations; n_p: number of predefined discrete poses; n_{3D}: channel dimension of added pose information mask representing yaw and pitch values; Conv-(): convolutional layer; DeConv-(): deconvolutional layer, n: number of convolution kernels, k: kernel size, s: stride size, p: padding size, In: instance normalization, Res: Residual Block.

Training details We train OPT-Net on iLab-20M (P-UB) dataset using Adam with β_1=0.5 and β_2=0.999, batch size 64, learning rate 0.0001 for the first 500 epochs and linear decay to 0 over the next 500 epochs.

Hyperparameters in the loss function: For Pose classification loss and Reconstruction loss, we use $\lambda_{cls} = 1$, $\lambda_{rec} = 10$ unmodified. For Pose-eliminate loss, as we described at the end of main paper 3.2, we set $\lambda_{pose} = 2$ at first 1/4 of all epochs (250 epochs), and then we linearly decay the λ_{pose} to 0 over the next 750 epochs.

During training, when we add new loss terms to GAN structure, we need to monitor the adversarial loss and make sure the new term will not break the balance. Moreover, for some loss to refine the quality or perform as regularization, like style consistency, we can monitor the performance of synthesized images and add them after the generator can synthesize good quality images.

0.2 Object pose significance on different object recognition tasks

As we described in main paper 4.4, after selecting categories that are more confused by the classifier: marker, comb, toothbrush, stapler, lightbulb, and sponge, we assign different fixed poses for each category to improve overall pose variance and form P-UB-1, P-UB-2, and P-UB-3 pose-unbalanced datasets. Table 4 shows the pose category details in different unbalanced datasets.

Table 1. Eliminate-add structure generator G network architecture

Part	Input → Output Shape	Layer Information
G_{elim}	(h, w, 3) → (h, w, 64)	Conv-(n64, k7x7, s1, p3), In, Relu
	(h, w, 64) → ($\frac{h}{2}$, $\frac{w}{2}$, 128)	Conv-(n128, k4x4, s2, p1), In, Relu
	($\frac{h}{2}$, $\frac{w}{2}$, 128) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Conv-(n256, k4x4, s2, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256)→($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256)→($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256)→($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
G_{add}	($\frac{h}{4}$, $\frac{w}{4}$, 256 + $\mathbf{n_{3D}}$) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256) → ($\frac{h}{4}$, $\frac{w}{4}$, 256)	Res: Conv-(n256, k3x3, s1, p1), In, Relu
	($\frac{h}{4}$, $\frac{w}{4}$, 256)→($\frac{h}{2}$, $\frac{w}{2}$, 128)	DeConv-(n128, k4x4, s2, p1), In, Relu
	$\frac{h}{2}$, $\frac{w}{2}$, 128)→(h, w, 64)	DeConv-(n64, k4x4, s2, p1), In, Relu
	(h, w, 64)→(h, w, 3)	Conv-(n3, k7x7, s1, p3), In, Tanh

Table 2. Discriminator D network architecture

Layer	Input → Output Shape	Layer Information
Input Layer	(h, w, 3) → ($\frac{h}{2}$, $\frac{w}{2}$, 64)	Conv-(n64, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{2}$, $\frac{w}{2}$, 64) → ($\frac{h}{4}$, $\frac{w}{4}$, 128)	Conv-(n128, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{4}$, $\frac{w}{4}$, 128) → ($\frac{h}{8}$, $\frac{w}{8}$, 256)	Conv-(n256, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{8}$, $\frac{w}{8}$, 256)→($\frac{h}{16}$, $\frac{w}{16}$, 512)	Conv-(512, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{16}$, $\frac{w}{16}$, 512)→($\frac{h}{32}$, $\frac{w}{32}$, 1024)	Conv-(1024, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{32}$, $\frac{w}{32}$, 1024)→($\frac{h}{64}$, $\frac{w}{64}$, 2048)	Conv-(n2048, k4x4, s2, p1), Leaky ReLU
Output Layer(D_{src})	($\frac{h}{64}$, $\frac{w}{64}$, 2048)→($\frac{h}{64}$, $\frac{w}{64}$, 1)	Conv-(n1, k3x3, s1, p1)
Output Layer(D_{cls})	($\frac{h}{64}$, $\frac{w}{64}$, 2048)→(1, 1, n_p)	Conv-($n(n_p)$, k $\frac{h}{64}$ x $\frac{h}{64}$, s1, p0)

0.3 Generalization to Imagenet

ImageNet image pose transform without finetune As we described in main paper 4.5, we directly use the pretrained OPT-Net on iLab-20M to synthesize different pose images for images from ImageNet without finetuning (Fig.1).

Although the general shape and pose structure are reasonable, the image quality is not as good as the synthesized images in iLab-20M. We interpret this result for two reasons. First, the object pose, background, and camera view of images in ImageNet are different from that in iLab-20M, which may cause domain gap and inferior quality. Second, we do not finetune OPT-Net on ImageNet, which may degrade the generation quality as well. It might be improved using domain adaptation or fine tuning in future work.

OPT-Net performance on boost object recognition in ImageNet As we described in main paper 4.5, we replace real images by OPT-Net synthesized

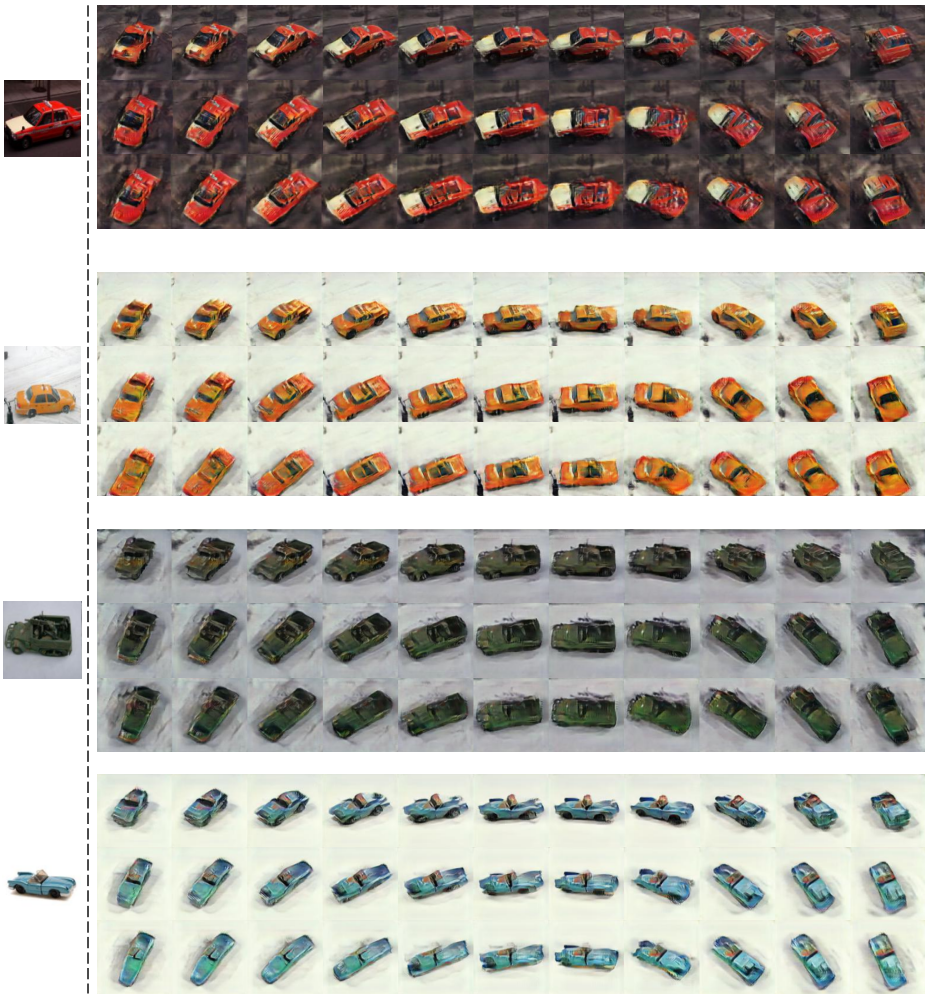

Fig. 1. Directly synthesized different-posed ImageNet images by OPT-Net without finetune.The first column shows input images from ImageNet, and the remaining columns show target pose images transformed by OPT-Net. Each row demonstrates the pose change along yaw axis, each column shows pose changes along pitch axis.

Table 3. Pose-eliminate module P_{elim} network architecture

Layer	Input → Output Shape	Layer Information
Input Layer	(h, w, 3) → ($\frac{h}{2}$, $\frac{w}{2}$, 64)	Conv-(n64, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{2}$, $\frac{w}{2}$, 64) → ($\frac{h}{4}$, $\frac{w}{4}$, 128)	Conv-(n128, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{4}$, $\frac{w}{4}$, 128) → ($\frac{h}{8}$, $\frac{w}{8}$, 256)	Conv-(n256, k4x4, s2, p1), Leaky ReLU
Residual Block	($\frac{h}{8}$, $\frac{w}{8}$, 256)→($\frac{h}{8}$, $\frac{w}{8}$, 256)	Res: Conv-(256, k3x3, s1, p1), Leaky ReLU
Hidden Layer	($\frac{h}{8}$, $\frac{w}{8}$, 256)→($\frac{h}{16}$, $\frac{w}{16}$, 512)	Conv-(512, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{16}$, $\frac{w}{16}$, 512)→($\frac{h}{32}$, $\frac{w}{32}$, 1024)	Conv-(1024, k4x4, s2, p1), Leaky ReLU
Hidden Layer	($\frac{h}{32}$, $\frac{w}{32}$, 1024)→($\frac{h}{64}$, $\frac{w}{64}$, 2048)	Conv-(n2048, k4x4, s2, p1), Leaky ReLU
Output Layer	($\frac{h}{64}$, $\frac{w}{64}$, 2048)→(1, 1, n_p)	Conv-($n(n_p)$, k $\frac{h}{64}$ x $\frac{h}{64}$, s1, p0)

category	Pose in P-UB-1	Pose in P-UB-2	Pose in P-UB-3
comb	1	1,2	1,2,3
toothpaste	2	2,3	2,3,4
flashlight	3	3,4	3,4,5
lightbulb	4	4,5	4,5,6
marker	5	5,6	1,5,6
sponge	6	1,6	1,2,6

Table 4. Available pose categories in different unbalanced datasets

images in S-P-B (main paper 4.2) and form a S-P-B (OPT-Net) dataset (all synthesized images). Similarly, we use StarGAN synthesized images form S-P-B (StarGAN). We use a resnet18 10-class vehicles classifier pretrained with this two synthesized datasets and predict 4 classes of vehicles in ImageNet which have similar meanings as iLab-20M, with good results on some classes like car (Shown in Table 5.).

Table 5 demonstrate two main points, first, a pre-trained OPT-Net learns some knowledge from iLab-20M, which can be directly used to boost classification tasks on ImageNet with no finetune (by synthesizing new images). Second, the synthesized performance of OPT-Net is better than other methods.

Generalization from iLab-20M to ImageNet As we described in main paper 4.5, to further explore generalization, we pretrian an AlexNet on S-P-B which synthesized pose images by StarGAN and OPT-Net respectively and then finetune it on ImageNet with limited images per class (randomly choose 10, 20, and 40 images for each class). Results in Table 6 show better accuracy compared to training from scratch when using only a small number of images per class, demonstrating generalization from iLab-20M to ImageNet.

Table 5. Generalization to ImageNet (percent correct)

Pretrain dataset	boat	car	mil	tank
No pretrain	0	0	0	0
S-P-B (StarGAN)	16.33	41.84	17.20	6.21
S-P-B (OPT-Net)	**31.14**	**76.20**	**30.47**	**12.86**

Table 6. Top-5 recognition accuracy (%) on ILSVRC-2010 test set

Number of images per class	10	20	40
Alexnet (scratch)	4.32	17.01	25.35
Alexnet (pretrain on S-P-B (StarGAN))	7.57	18.91	27.14
Alexnet (pretrain on S-P-B (OPT-Net))	**10.94**	**19.75**	**27.82**

0.4 Additional Qualitative Pose Transformation Results of OPT-Net

Figs. 2, 3, and 4 show additional images with 128×128 resolutions pose transformation results generated by OPT-Net. All images were generated by a eliminate-add structure generator trained on iLab-20M datasets. The first column shows input images from the test dataset, and the remaining columns show target pose images transformed by OPT-Net. Poses framed in red are defined in the training dataset, while all other poses are new poses, which shows OPT-Net can achieve continuous pose transformation.

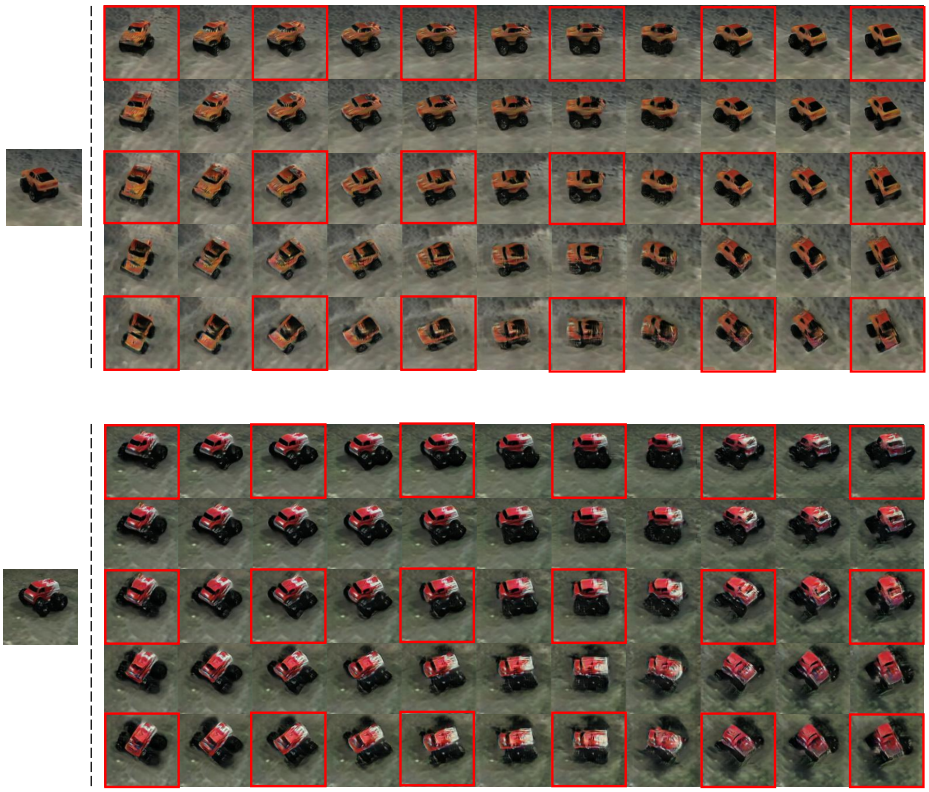

Fig. 2. OPT-Net pose transformation results on monster vehicle, The first column shows input images from the test dataset, and the remaining columns show target pose images transformed by OPT-Net. Poses framed in red are defined in the training dataset, while all other poses are new poses, which shows OPT-Net can achieve continuous pose transformation

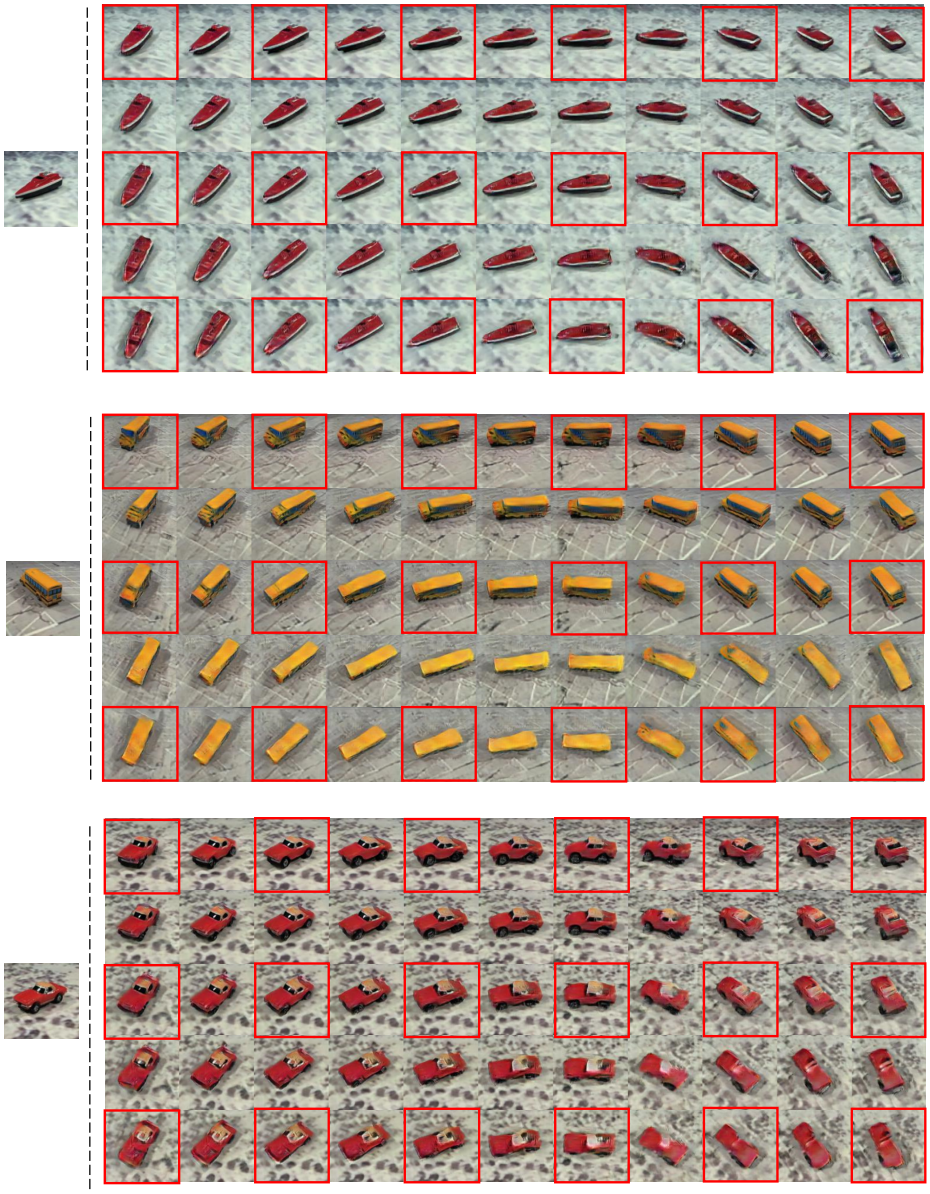

Fig. 3. OPT-Net pose transformation results on boat, bus and car, The first column shows input images from the test dataset, and the remaining columns show target pose images transformed by OPT-Net. Poses framed in red are defined in the training dataset, while all other poses are new poses, which shows OPT-Net can achieve continuous pose transformation

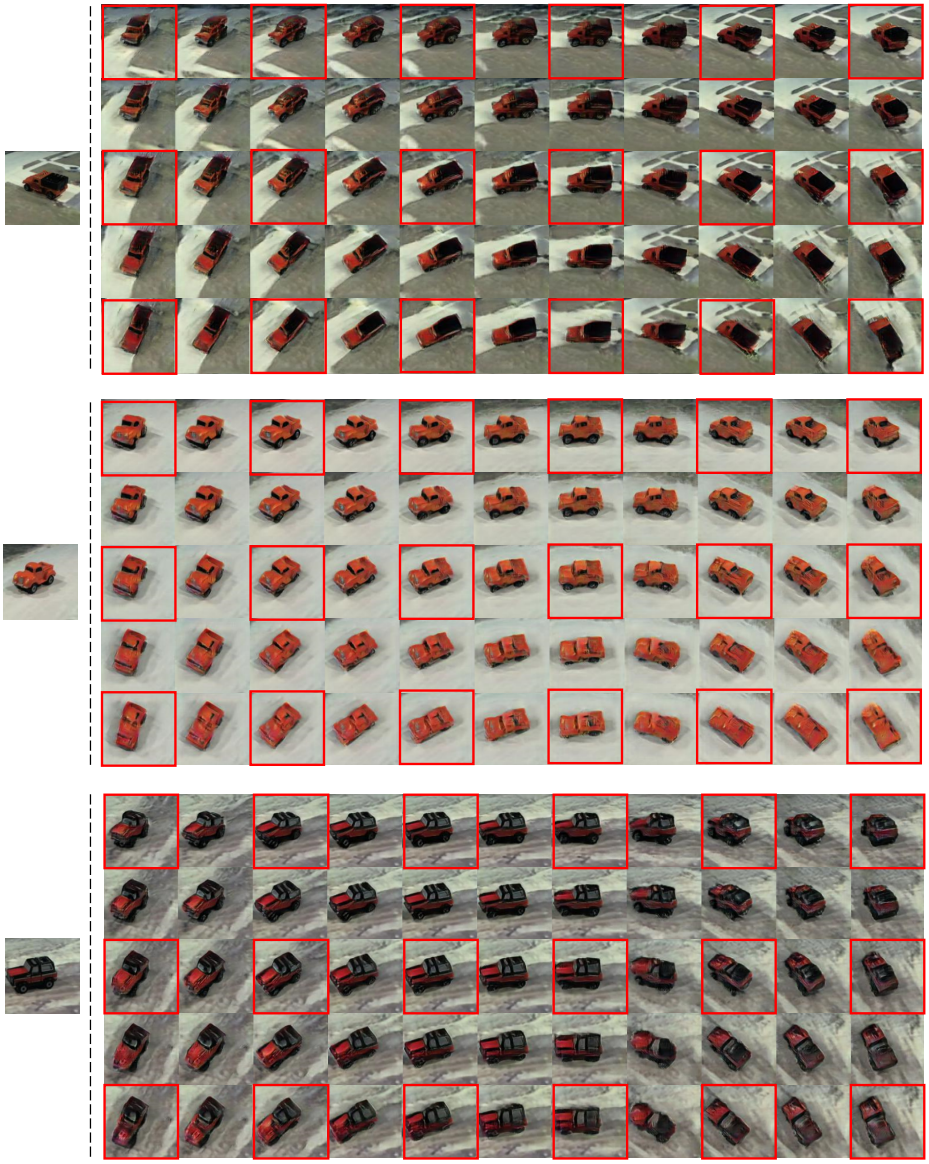

Fig. 4. OPT-Net pose transformation results on pick-up vehicle, The first column shows input images from the test dataset, and the remaining columns show target pose images transformed by OPT-Net. Poses framed in red are defined in the training dataset, while all other poses are new poses, which shows OPT-Net can achieve continuous pose transformation