CRAVES: Controlling Robotic Arm with a Vision-based Economic System
Supplementary Materials

Yiming Zuo[1]*, Weichao Qiu[2]*, Lingxi Xie[2,4], Fangwei Zhong[3], Yizhou Wang[3,5,6], Alan L. Yuille[2]
[1]Tsinghua University [2]Johns Hopkins University [3]Peking University
[4]Noah's Ark Lab, Huawei Inc. [5]Peng Cheng Laboratory [6]DeepWise AI Lab
{zuoyiming17,qiuwch,198808xc,zfw1226,alan.l.yuille}@gmail.com, yizhou.wang@pku.edu.cn

1. Implementation Details of Reinforcement Learning

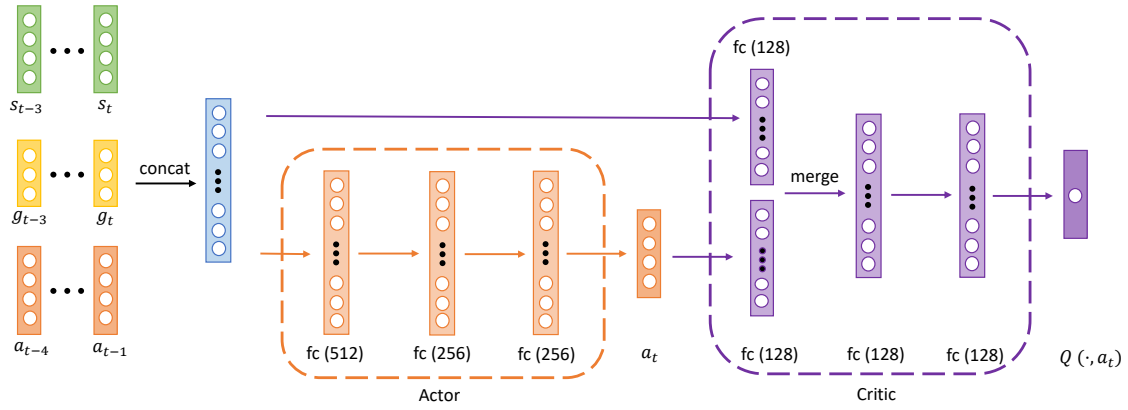

Figure 1. Details of the DDPG network for reaching a point.

We develop an RL-based agent to control the arm to reach a target point without collision. This section introduces the implementation details of the training process, including the reward function, action, network, *etc*.

- **Reward Shaping.** Our goal is to train an agent to reach a target point as fast as possible and without collision. So, the environment should provide a large positive reward when the goal is achieved, and a large negative reward (*a.k.a.* penalty) when collision happens. At other steps, the definition of reward should encourage the grip to get closer to the goal and reach the point using as few steps as possible. Considering the above requirements, the reward function is shaped as below:

$$r^{\beta} = \begin{cases} 10, & \text{if } d_t < D; \\ -10, & \text{if collision detected;} \\ 0.05(d_{t-1} - d_t) - 0.1, & \text{otherwise.} \end{cases} \quad (1)$$

  Here, d_t is the distance between the grip and the target point at time t. Note that an episode is done when the target is reached, a collision is detected, or the maximum number of steps (200 in our experiments) is achieved.

- **Action.** An action $\mathbf{a}_t$ is a vector in a continuous space, with its values in each dimension corresponding to the expected change of each angle. The value of each dimension is truncated by $[-5°, 5°]$.

- **Input.** A state $\mathbf{s}_t$ is a 4-dimensional vector which describes the pose of the arm, *i.e.*, each dimension indicates the angle of one joint. The goal is a 3-dimensional vector $\mathbf{g}_t$ representing the coordinate of the target point in the 3D space. The

*This work was done when the first author was an intern at the Johns Hopkins University. The first two authors contributed equally.

	Synthetic	Lab	YouTube
Source	UE4	Webcam	YouTube
Total Videos	N/A	20	109
Total Frames	$+\infty$	21,720	503,622[1]
Training Labels	4,500	0	0
Testing Labels	500	428	275
Annotation Type	3D Pose	3D Pose	2D Keypoints

Table 1. Data statistics of different datasets. Note that, with 3D pose annotated, one can compute 2D keypoints with simple geometry.

input of the agent network is composed of three parts, namely, the state vector $\mathbf{s}_t$, the goal vector $\mathbf{g}_t$, and the action taken at the previous step $\mathbf{a}_{t-1}$. For providing richer temporal information, we stack the above input vectors from the 4 most recent frames.

- **Network.** The agent is an actor-critic network (see Figure 1), in which the actor is a policy function $\mathbf{a}_t = \pi(\cdot; \boldsymbol{\xi})$, mapping the input to an action $\mathbf{a}_t$, and the citric predicts the expected discounted future reward, $Q(\cdot, \mathbf{a}_t)$, which is used for training the actor.

- **Training.** The network is trained with the DDPG [1] algorithm. To accelerate the learning process, we use a prioritized experience replay with a buffer size of 100K. The learning rate is 0.001 for the critic and 0.0001 for the actor. To encourage the actor to explore, we add Ornstein-Uhlenbeck noise to the output action. The factor of the noise decays linearly from 1 to 0.1 in the first 100K steps and remains a constant afterwards. The entire training process contains 400K steps. Note that we use the ground-truth arm pose as the state input during training. At the beginning of each episode, we randomly reset the target coordinate and the initial arm pose.

2. Video Demonstrations

2.1. Reach above a Target Point

In reach_point.mp4, we demonstrate a video on the task for quantitative evaluation, *i.e.*, controlling the arm to reach above target points. The video is shown in the first-person view, which is exactly what the vision algorithm takes as input. The video contains 18 clips, including 14 success cases and 4 failure cases produced by our vision-based robotic arm.

2.2. Move Dices into a Box

In move_dices.mp4, we show a video demo for moving a stack of dices into a box. To simplify deployment, we directly feed the ground-truth locations of the dices as well as the box into the controller, instead of applying another vision-based module to estimating these variables.

The successful rate of this task is not high, because it requires highly accurate pose control in both horizontal and vertical directions. We have also provided some interesting failure cases at the end of the video. At the current point, this demo reveals the potential of our vision-based control system, as well as advocates more advanced vision algorithms to be designed to improve its performance.

3. Dataset Statistics

Detailed statistics of our datasets are shown in Table 1.

4. Example Images for Domain Adaptation

Finally, we show some images we used in the paper for domain adaptation. Figure 2 shows some close-ups of the images from different domains. Figure 3 shows sample images generated by the CycleGAN [2] algorithm, including successful and failure cases. Figure 4 shows some real images generated by background augmentation.

[1] Only part of these frames are usable. Some of them are introductions or about assembly, etc.

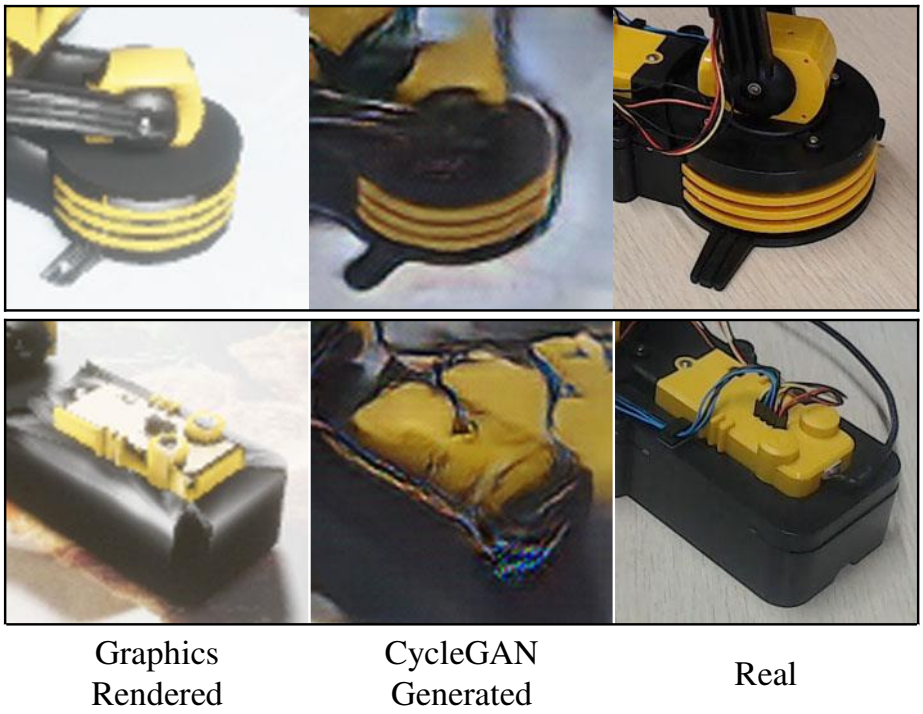

Figure 2. Comparison between close-ups of two graphics rendered images, two CycleGAN-generated images, and two real images. The CycleGAN algorithm is able to learn some fine-scaled features (*e.g.*, screws and wires) from real training images.

Figure 3. Sample images generated by CycleGAN. Each image pair consists of a source image (left) and a stylized image (right). Both successful cases (top two rows) and failure cases (bottom row) are shown.

Figure 4. Real training samples after background augmentation. On an image captured in the lab environment, we segment the arm with a pixel-wise color-based filter, and then place it onto a random image from the MS-COCO dataset.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference for Learning Representations*, 2016. 2

[2] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, 2017. 2