Visual Servoing With Deep Models

Xuchong Qiu

Yang Xiao

Outline

- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Deep Reinforcement Learning

Outline

- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Reinforcement Learning

Eye-in-hand and eye-to-hand configurations

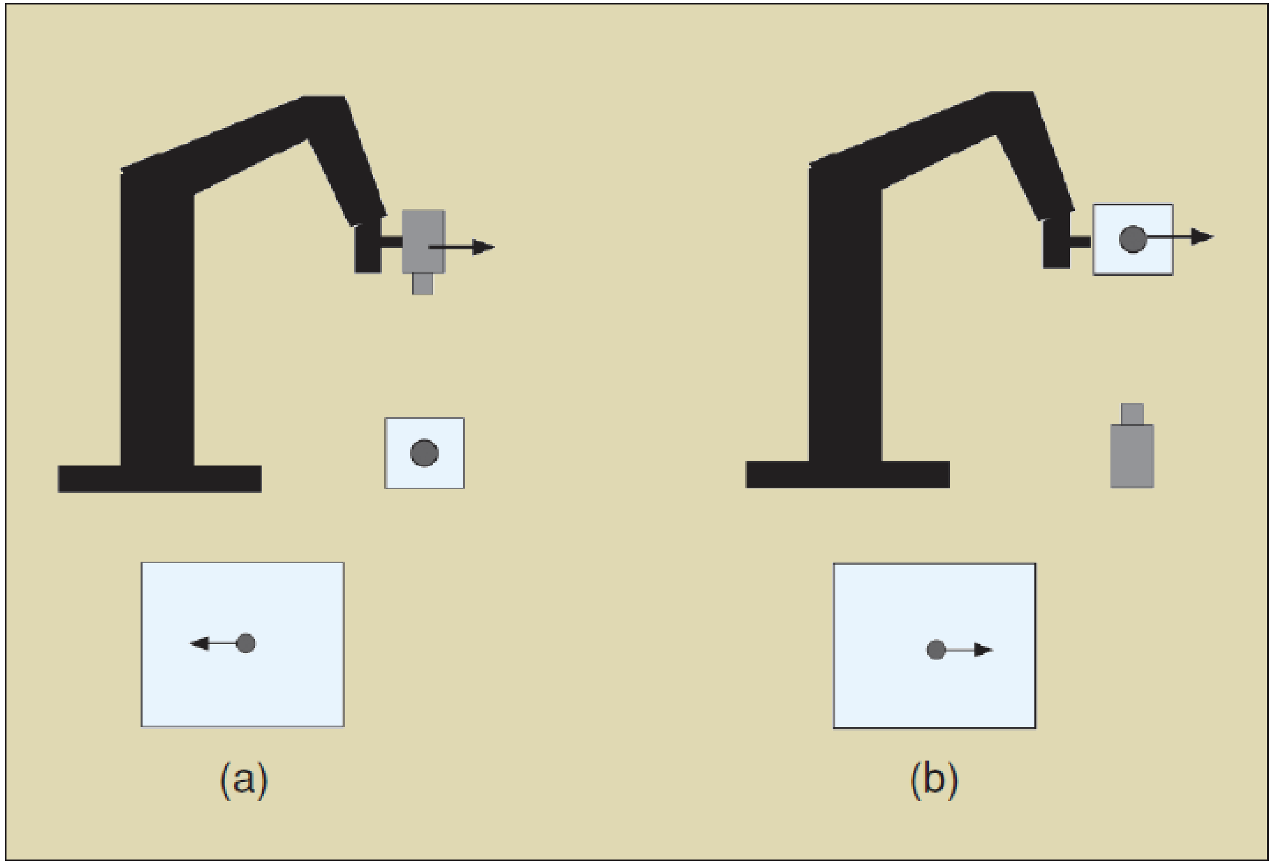


Figure 5. *Top: (a) Eye-in-hand system. (b) Eye-to-hand system. Bottom: Opposite image motion produced by the same robot motion.*

[François Chaumette et al, 2007]

Basic components of visual servoing

Goal:

Give controller motion commands to minimize the error according visual information

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

e(t) Error of feature vector to minimize

s(**m**(t), **a**) measured k-dim feature vector according to visual information

s* Desired feature vector

m(t) Visual information, e.g. image coordinates of interest point

a Additional information about system, e.g. camera intrinsic

Example: camera motion controller

- Static desired camera pose s* and image I of a motionless target
- Eye-in-hand system where camera is fixed at end-effector

$$\mathbf{v}_c = (v_c, \boldsymbol{\omega}_c)$$ Motion command

Define: $$\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v}_c$$ $\mathbf{L_s} \in \mathbb{R}^{k\times 6}$

$$\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$$ where $\mathbf{L_e} = \mathbf{L_s}$

$$\dot{\mathbf{e}} = -\lambda\mathbf{e}$$ Desired exponential decreasing

$$\mathbf{v}_c = -\lambda\mathbf{L_e^+}\mathbf{e}$$ $\mathbf{L_e^+} = (\mathbf{L_e}^\top\mathbf{L_e})^{-1}\mathbf{L_e}^\top$

$$\mathbf{v}_c = -\lambda\widehat{\mathbf{L_e^+}}\mathbf{e}$$

Keypoints:

- Definition of feature vector **s,** therefore **L**e

- Estimation of **L**e 's pseudo inverse or inverse(if k = 6 and **L**e nonsingular)

Basic variants depending on **s**

- Image-Based Visual Servoing (IBVS)

$$\mathbf{s} = \mathbf{x} = (x,\ y)$$

- Pose-Based Visual Servoing (PBVS)

use the pose of the camera **p** with respect to some reference frame

To define **s,** (**t**, **R**) relative to reference frame F_0 or desired camera frame $^*F$

- Direct visual servoing (DVS, where **p** is camera pose)

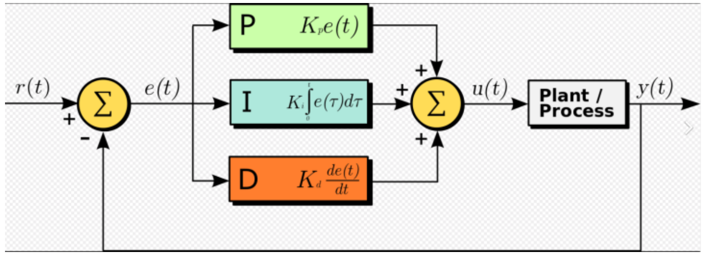
$$\mathbf{s}(\mathbf{p}) = \mathbf{I}_{\mathbf{x}}(\mathbf{p}) = (\mathbf{I}_{1\bullet}, \mathbf{I}_{2\bullet}, \ldots, \mathbf{I}_{N\bullet})$$

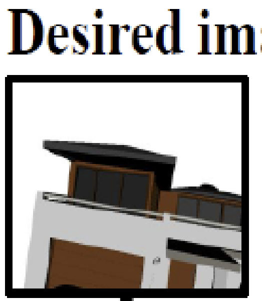
Outline

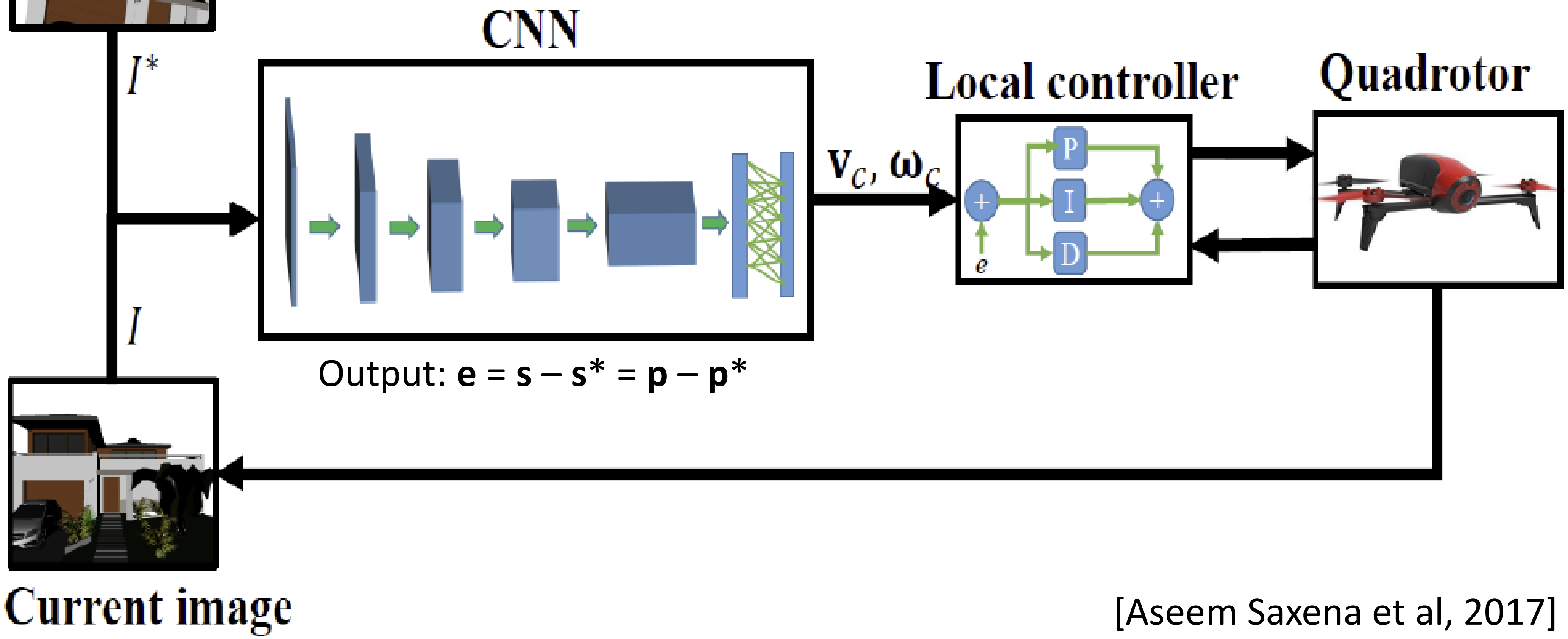
- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Reinforcement Learning

Exploring Convolutional Networks for End-to-End Visual Servoing

PID controller







Output: $\mathbf{e} = \mathbf{s} - \mathbf{s}^* = \mathbf{p} - \mathbf{p}^*$

[Aseem Saxena et al, 2017]

Exploring Convolutional Networks for End-to-End Visual Servoing

Our network takes in two monocular images I, I^* and outputs a pose vector $\mathbf{p}$ comprising of a relative (I^* with respect to I) translation $\mathbf{x}$ and rotation $\mathbf{q}$ in quaternion form.

$$\mathbf{p} = [\mathbf{x}, \mathbf{q}] \tag{1}$$

To regress relative pose, we consider the following objective loss function similar to [12].

$$loss(I, I^*) = \|\hat{x} - x\|_2 + \beta \left\| \hat{q} - \frac{q}{\|q\|} \right\|_2 \tag{2}$$

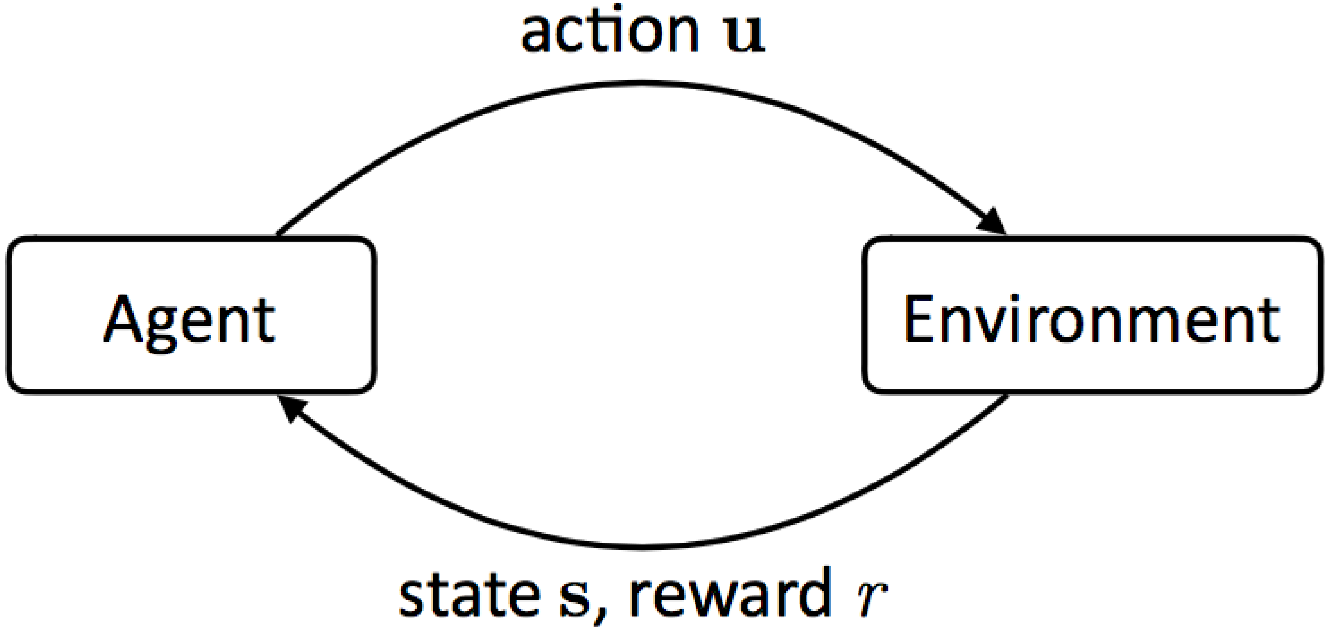
β is chosen so as to keep the expected value of translation and rotation errors to be equal. We found β as $500,000$ to be optimal for training. The motive behind deploying

[Aseem Saxena et al, 2017]

Outline

- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Reinforcement Learning

What is Reinforcement Learning

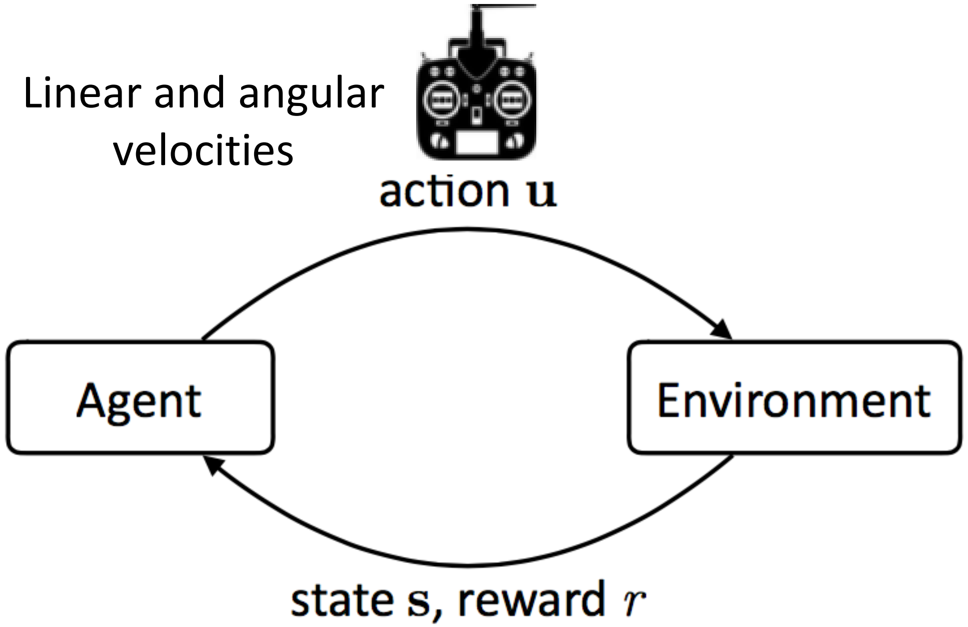


In robotics, the state can be camera images or joint angles of the robot, and the action can be the robot commands.

[Alex et al, 2017 ICLR]

Learning visual servoing with RL

Linear and angular velocities

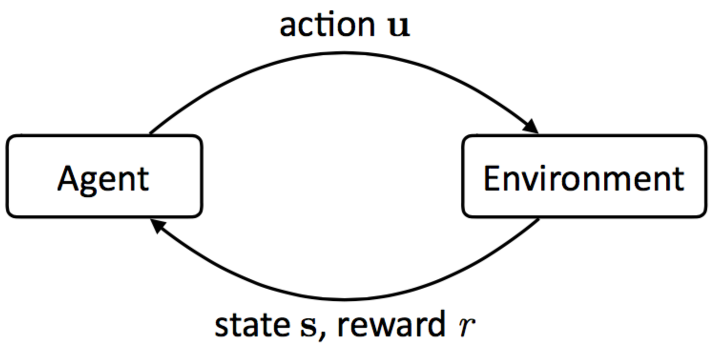


Current and goal image observation

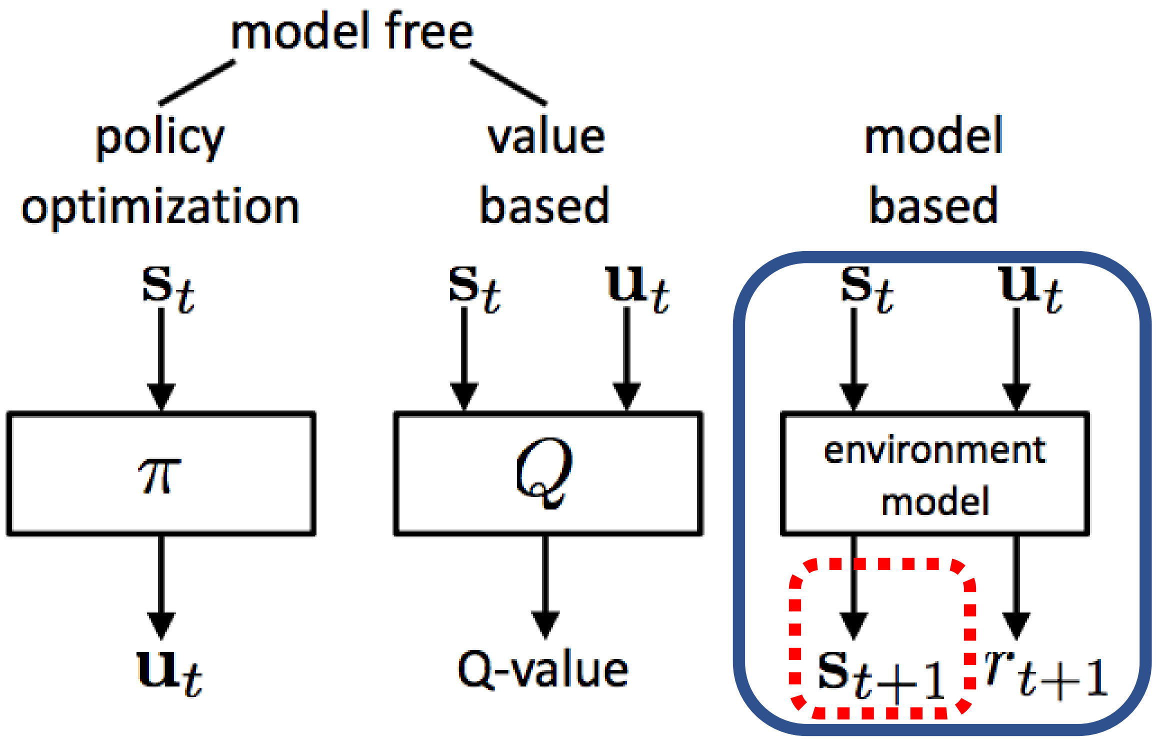
Distance from the drone to desired position

[Alex et al, 2017 ICLR]

Reinforcement Learning Approaches



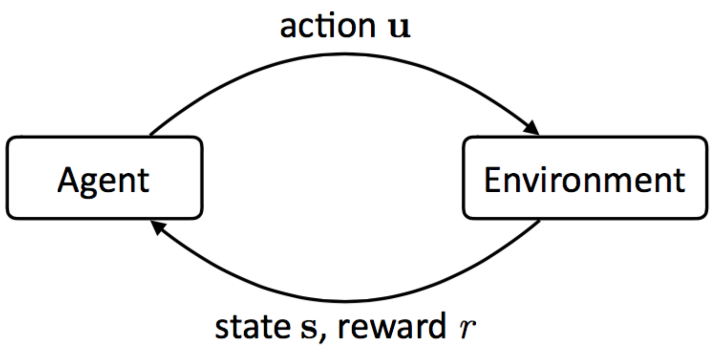
$$s_{t+1}, r_{t+1} = f(s_t, u_t)$$
$$\hat{u} = \arg\min d(s_{t+1}, s^*)$$

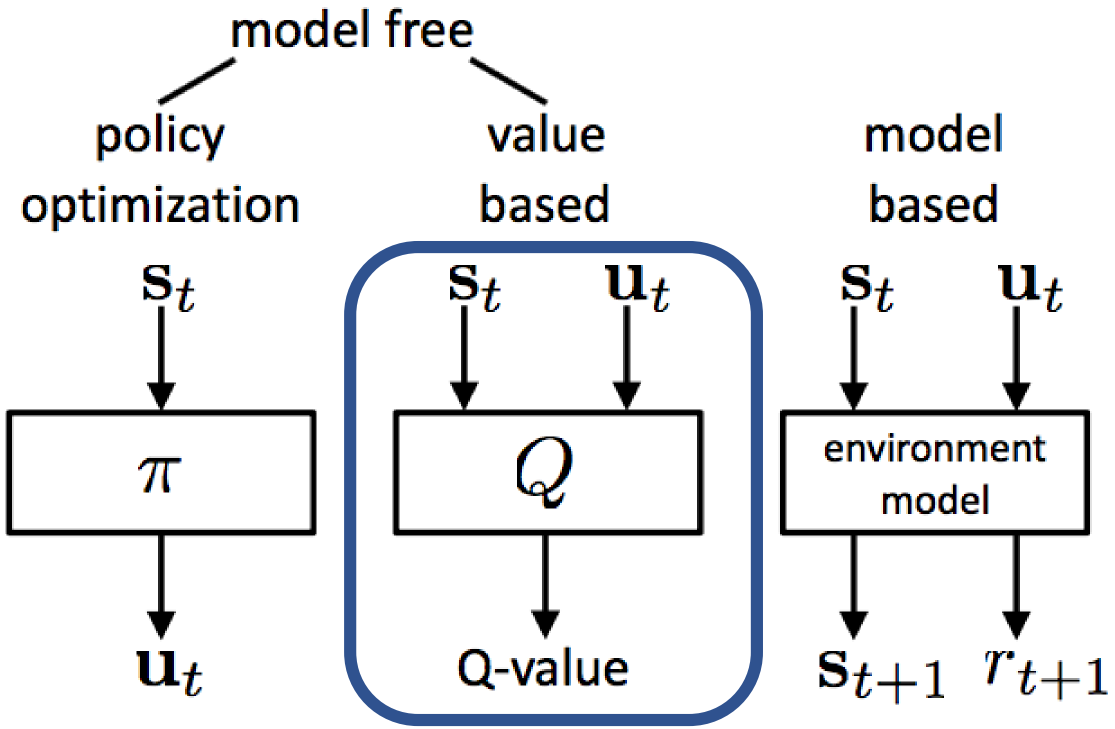


If we have access to the environment model

[Alex et al, 2017 ICLR]

Reinforcement Learning Approaches



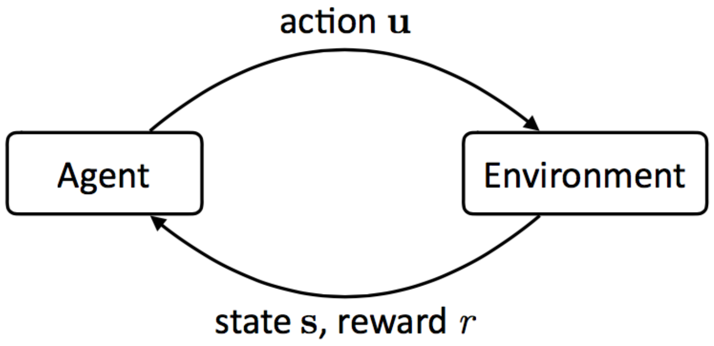


$$Q_t = Q(s_t, u_t)$$
$$\hat{u} = \arg\max E[Q_t]$$

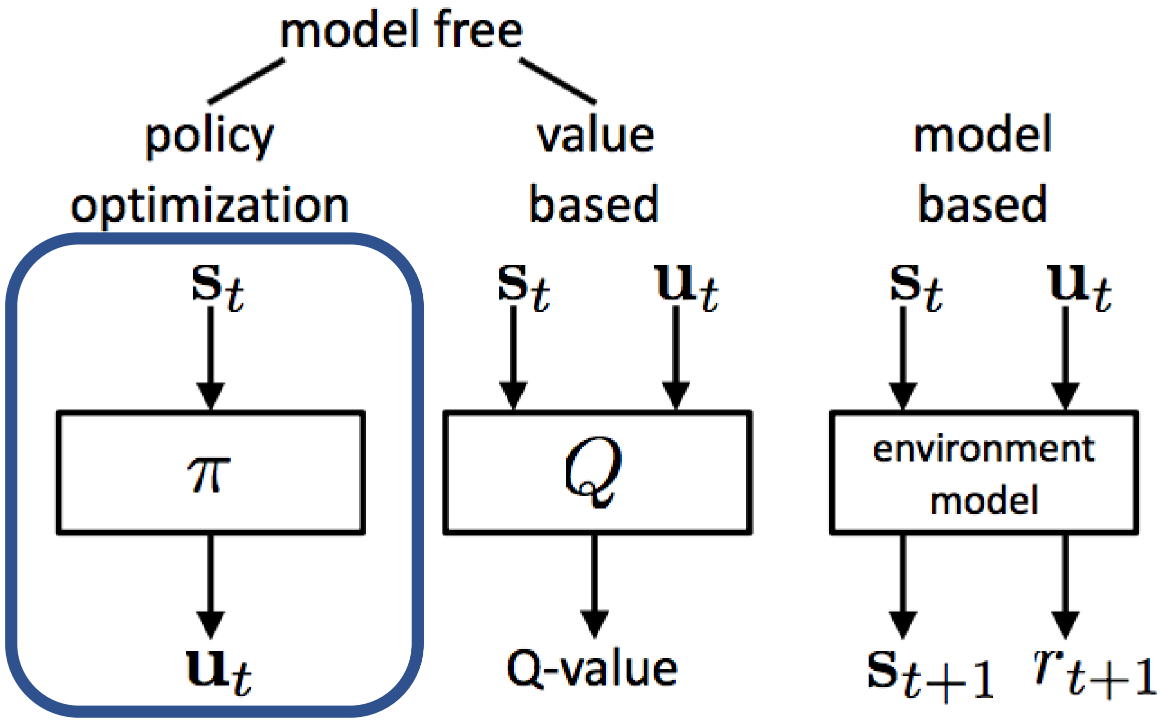
Learn a value-function

[Alex et al, 2017 ICLR]

Reinforcement Learning Approaches



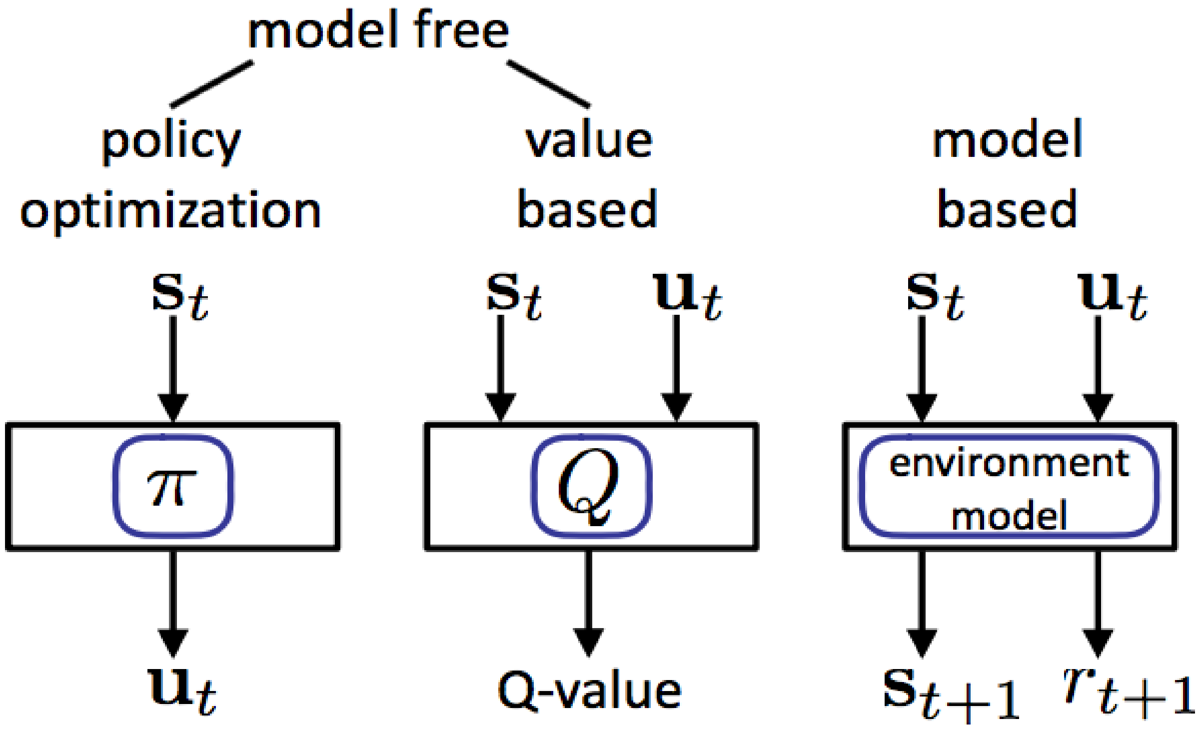
$$u_t = \pi(s_t)$$
$$\hat{u} = \arg\min d(u_t, u^*)$$



Simple policy optimization requires supervision on optimal action

[Alex et al, 2017 ICLR]

Deep Reinforcement Learning



Deep neural networks can be used to approximate complex functions.

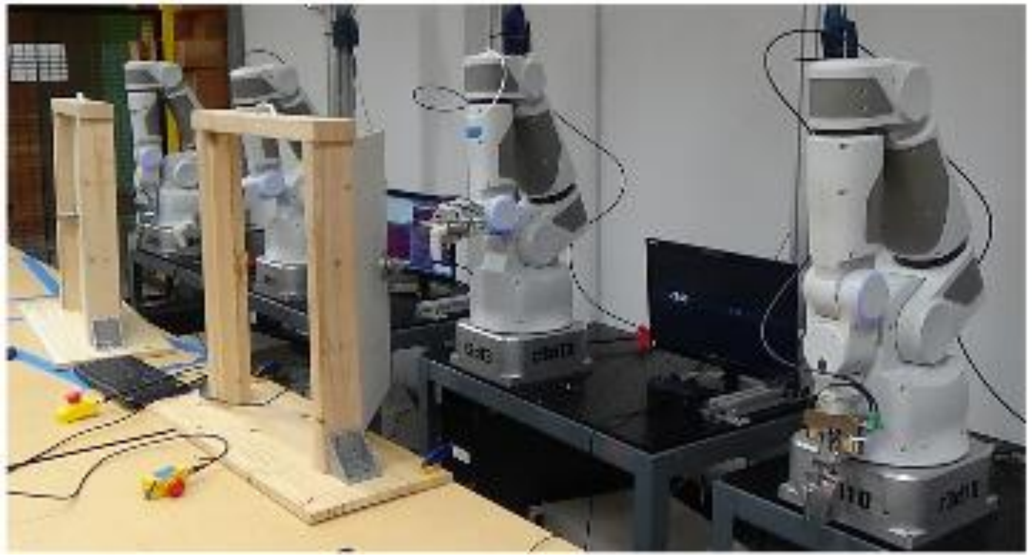
[Alex et al, 2017 ICLR]

Examples of Deep Reinforcement Learning

[Mnih et al, 2015]

[OpenAI, 2017]

Examples of Deep Reinforcement Learning



[Gu*, Holly* et al, 2017 ICRA]

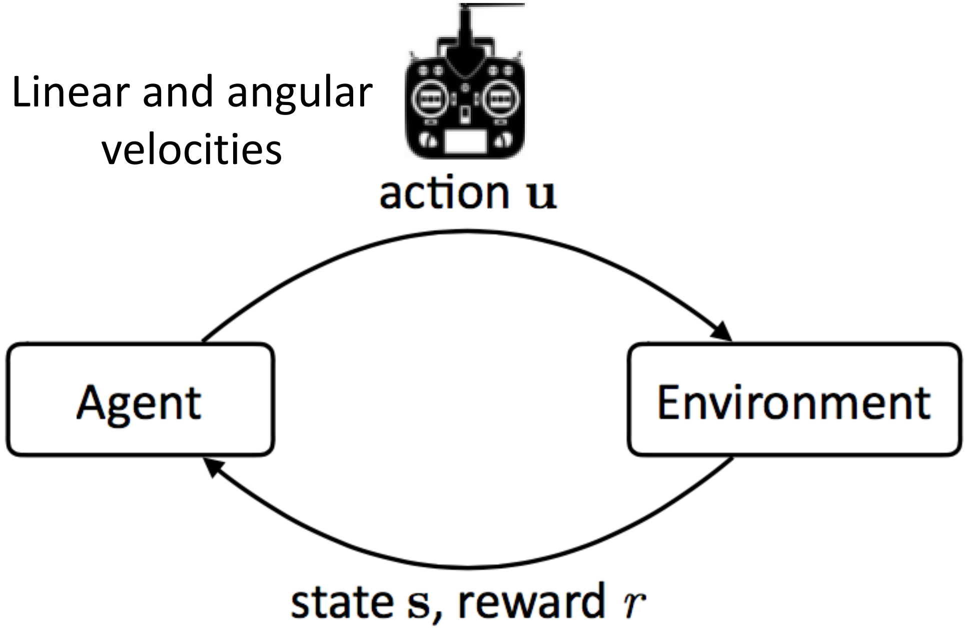
[Levine*, Finn* et al, 2017 JMLR]

Outline

- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Reinforcement Learning
  - Learning visual servoing with deep features and fitted Q-iteration
  - Sim2Real View Invariant Visual Servoing by Recurrent Control

Learning visual servoing with RL

Linear and angular velocities



Current and goal image observation

Distance from the drone to desired pose relative to car

[Alex et al, 2017 ICLR]

Combining Q-Value and Model Based RL

State-action value based RL:

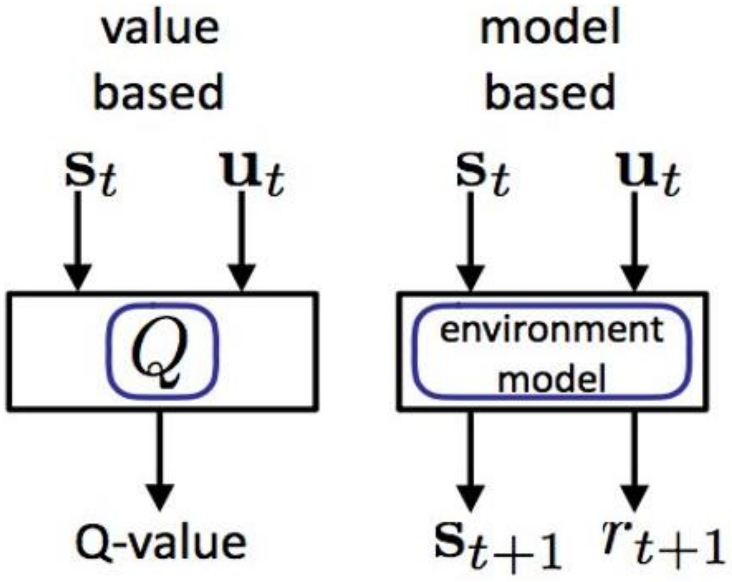
$$\pi(\mathbf{s}_t) = \arg\min_{\mathbf{u}} -Q(\mathbf{s}_t, \mathbf{u})$$

Visual Servoing:

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

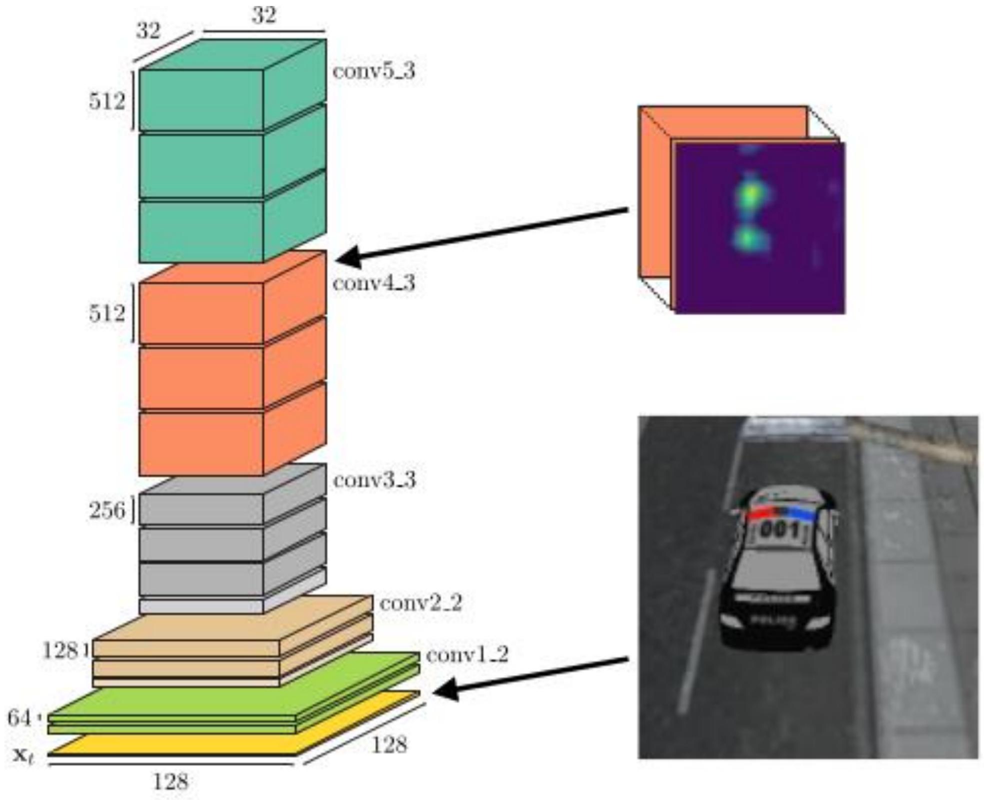
dynamics function

$$\pi(\mathbf{s}_t) = \arg\min_{\mathbf{u}} \underbrace{||\mathbf{x}_* - f(\mathbf{x}_t, \mathbf{u}_t)||^2}_{-Q(\mathbf{s}_t, \mathbf{u})}$$



$\hat{x}_{t+1} = f(x_t, u_t)$ where x_t is the current image observation, $x_t \epsilon s_t$

Representing the state as network activations



$$\pi(\mathbf{x}_t, \mathbf{x}_*) = \arg\min_{\mathbf{u}} \|\mathbf{y}_* - f(\mathbf{y}_t, \mathbf{u})\|^2 .$$

$$y_t = Net(x_t)$$

dynamics function

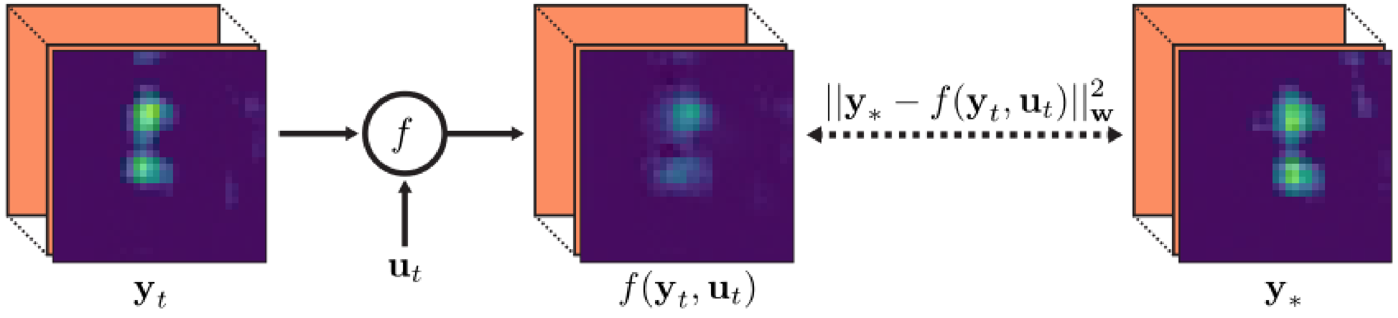
$$\pi(\mathbf{s}_t) = \arg\min_{\mathbf{u}} \underbrace{\|\mathbf{x}_* - f(\mathbf{x}_t, \mathbf{u}_t)\|^2}_{-Q(\mathbf{s}_t, \mathbf{u})}$$

[VGG-16, 2015 ICLR]

Bilinear Dynamics

$$\hat{\mathbf{y}}^{(l)}_{t+1,c} = \mathbf{y}^{(l)}_{t,c} + \sum_j \left(\boldsymbol{W}^{(l)}_{c,j} * \mathbf{y}^{(l)}_{t,c} + \boldsymbol{B}^{(l)}_{c,j}\right)\mathbf{u}_{t,j} + \left(\boldsymbol{W}^{(l)}_{c,0} * \mathbf{y}^{(l)}_{t,c} + \boldsymbol{B}^{(l)}_{c,0}\right)$$

$$\pi(\mathbf{x}_t, \mathbf{x}_*) = \arg\min_{\mathbf{u}} \underbrace{||\mathbf{y}_* - f(\mathbf{y}_t, \mathbf{u}_t)||^2_{\mathbf{w}}}_{-Q_{\mathbf{w}}(\mathbf{s}_t, \mathbf{u})}$$



current feature

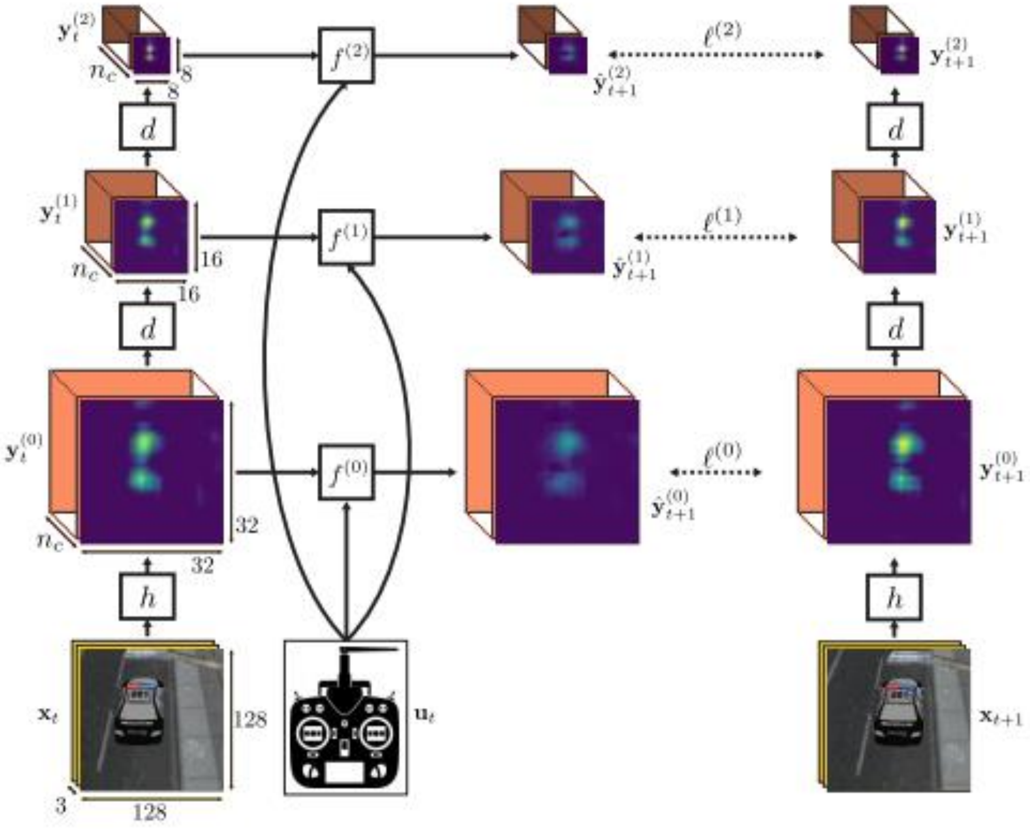
predicted feature

goal feature

[Alex et al, 2017 ICLR]

Servoing with Visual Dynamics Model

$$\pi(\mathbf{x}_t, \mathbf{x}_*) = \arg\min_{\mathbf{u}} \sum_c \sum_{l=0}^{L} \frac{\mathbf{w}_c^{(l)}}{|\mathbf{y}_{\cdot,c}^{(l)}|} \left\| \mathbf{y}_{*,c}^{(l)} - f_c^{(l)}\left(\mathbf{y}_{t,c}^{(l)}, \mathbf{u}\right) \right\|_2^2 + \sum_j \lambda_j \mathbf{u}_j^2$$



Good results with only 20 trajectories while model-free approach needs 20k trajectories

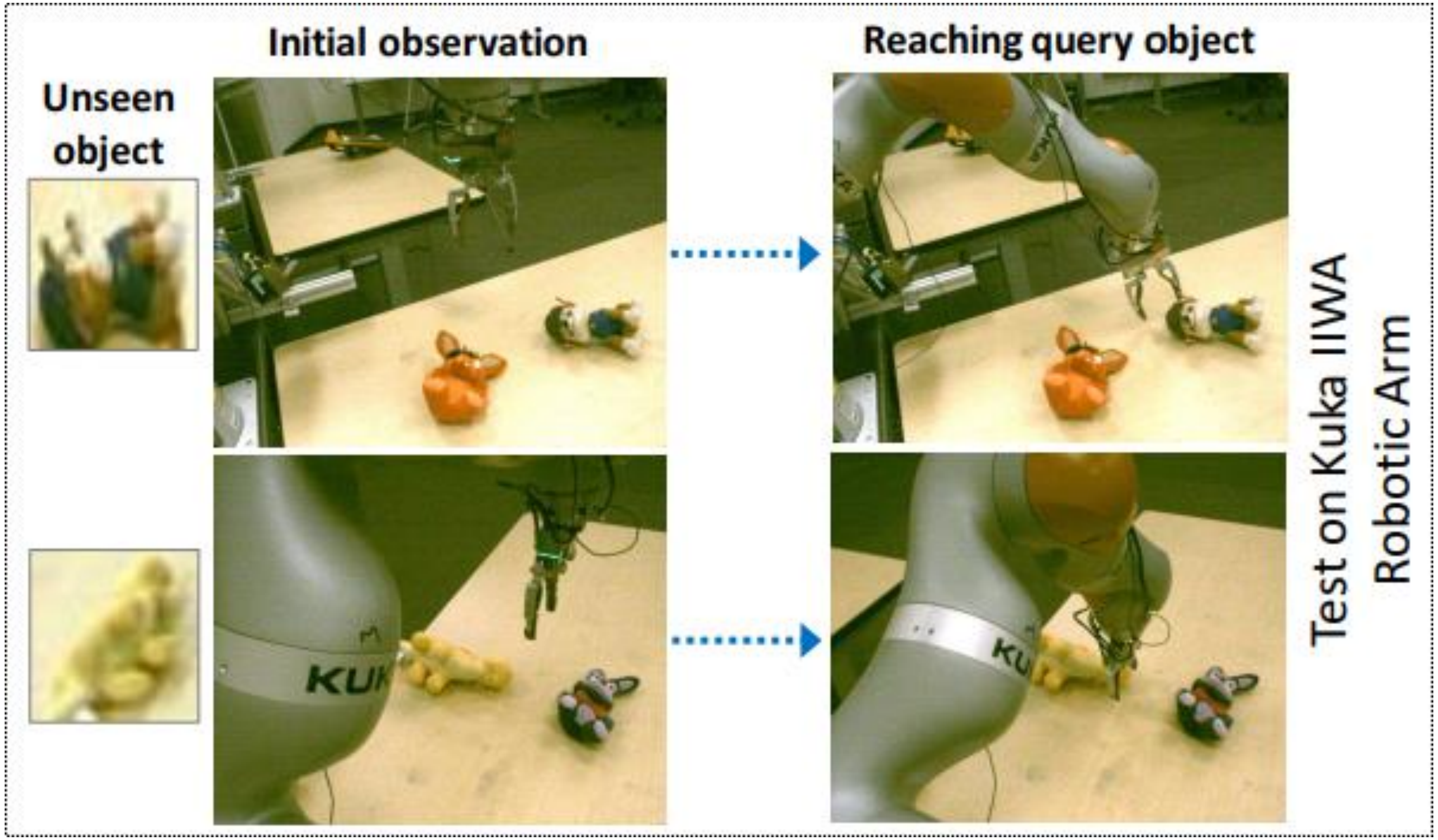
[Alex et al, 2017 ICLR]

Outline

- Visual servoing
- Visual servoing with deep features
- Reinforcement Learning
- Visual servoing with Reinforcement Learning
    - Learning visual servoing with deep features and fitted Q-iteration
    - Sim2Real View Invariant Visual Servoing by Recurrent Control

Reach objects from various viewpoints

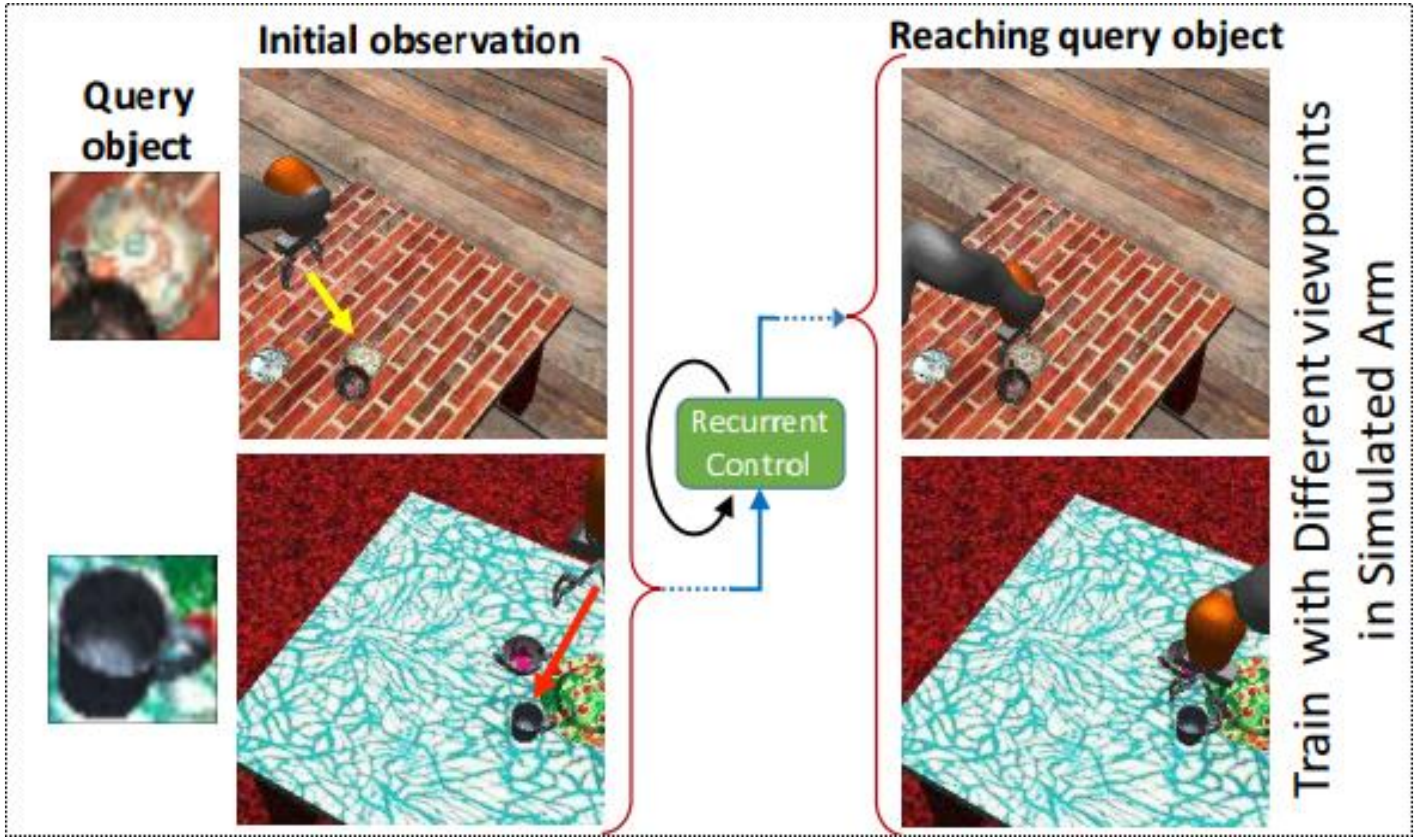
- Uncalibrated camera
- Novel objects
- Unseen viewpoints



[Sadeghi et al, 2018 CVPR]

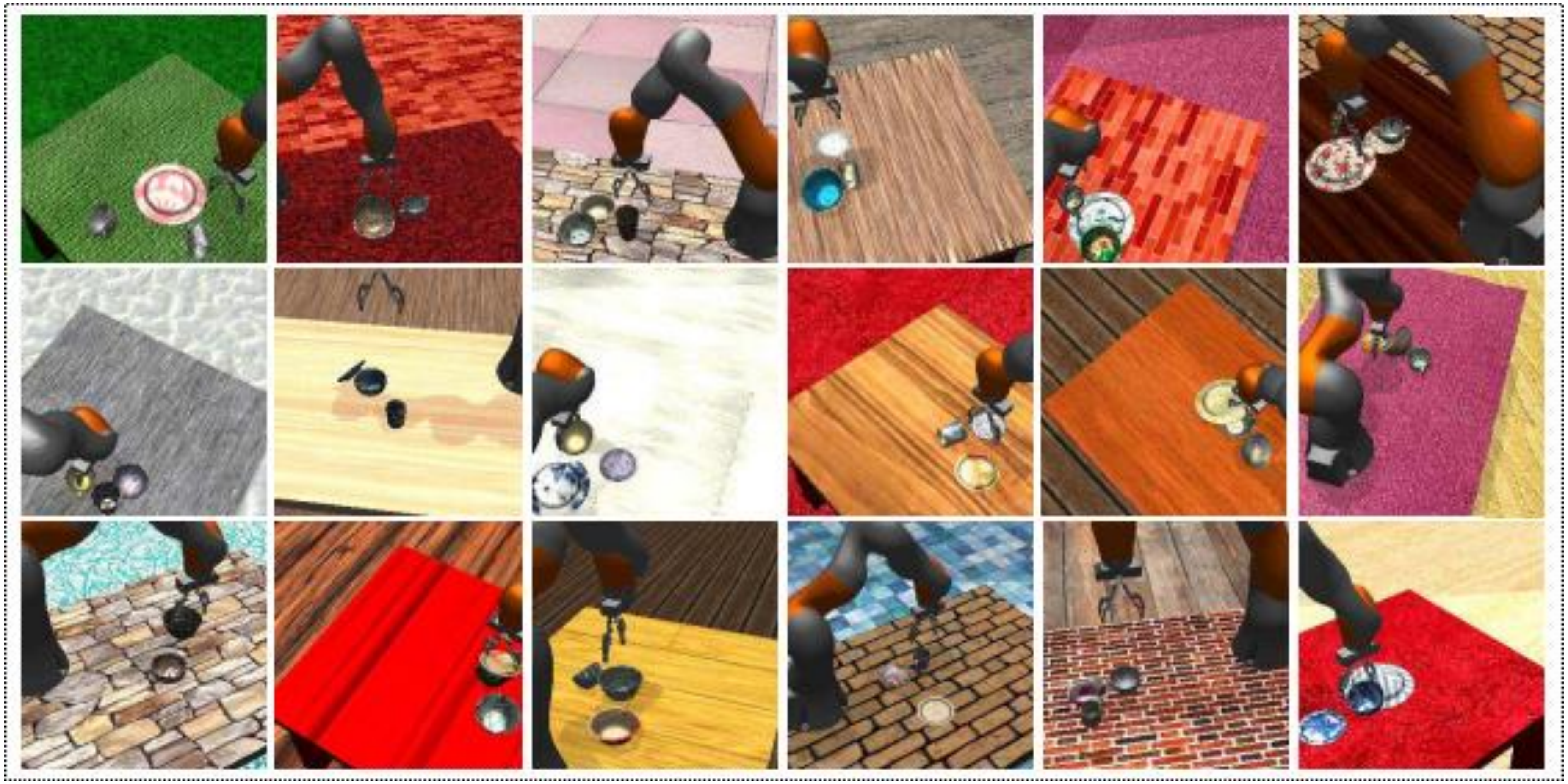
Network training in simulation

- Simulation environment
- Texture variation
- Recurrent control

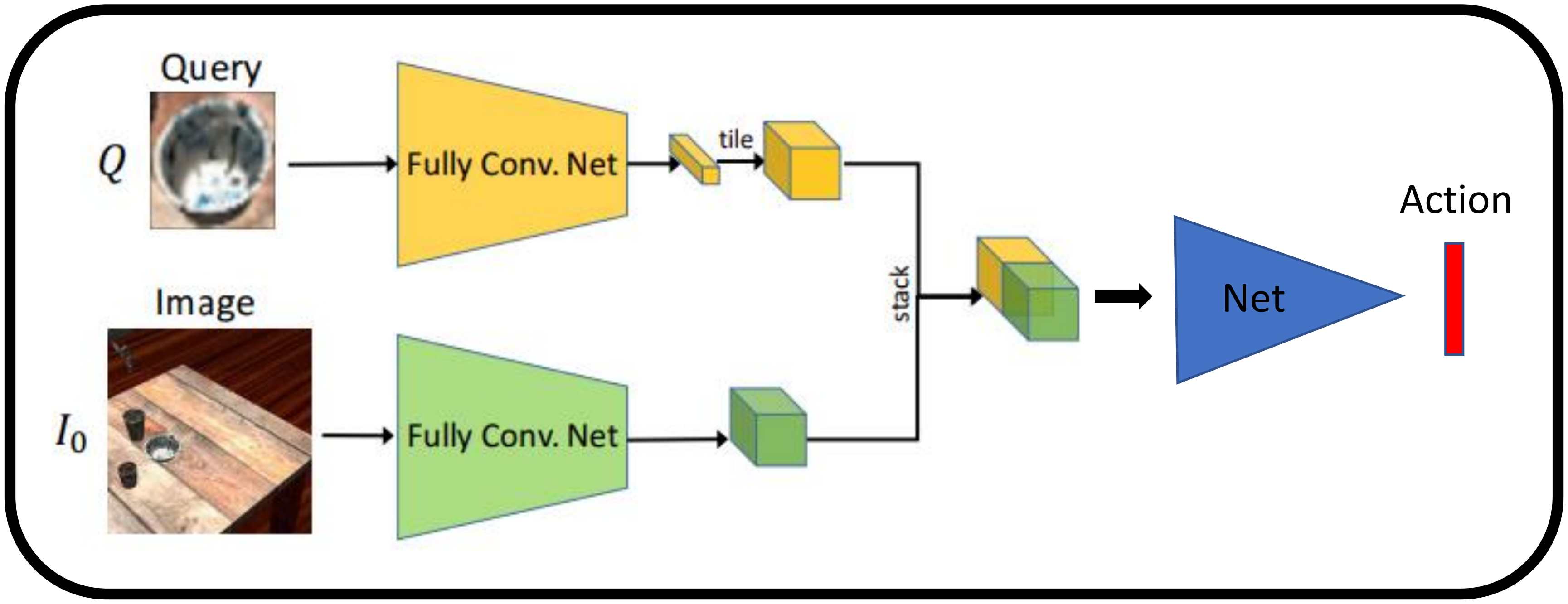


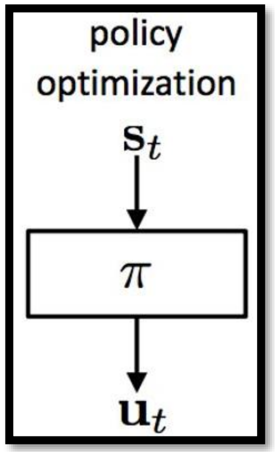
[Sadeghi et al, 2018 CVPR]

Domain randomization



Model free policy learning

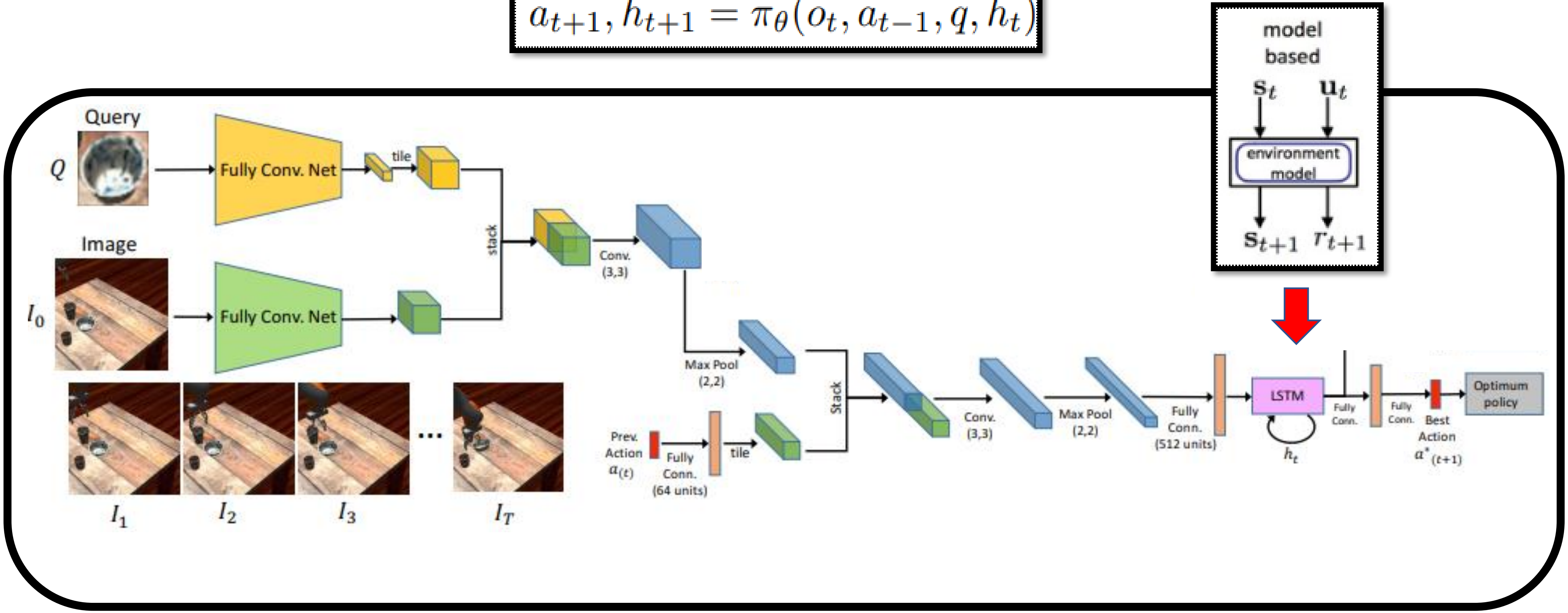




The action is the end-effector movement in Cartesian space

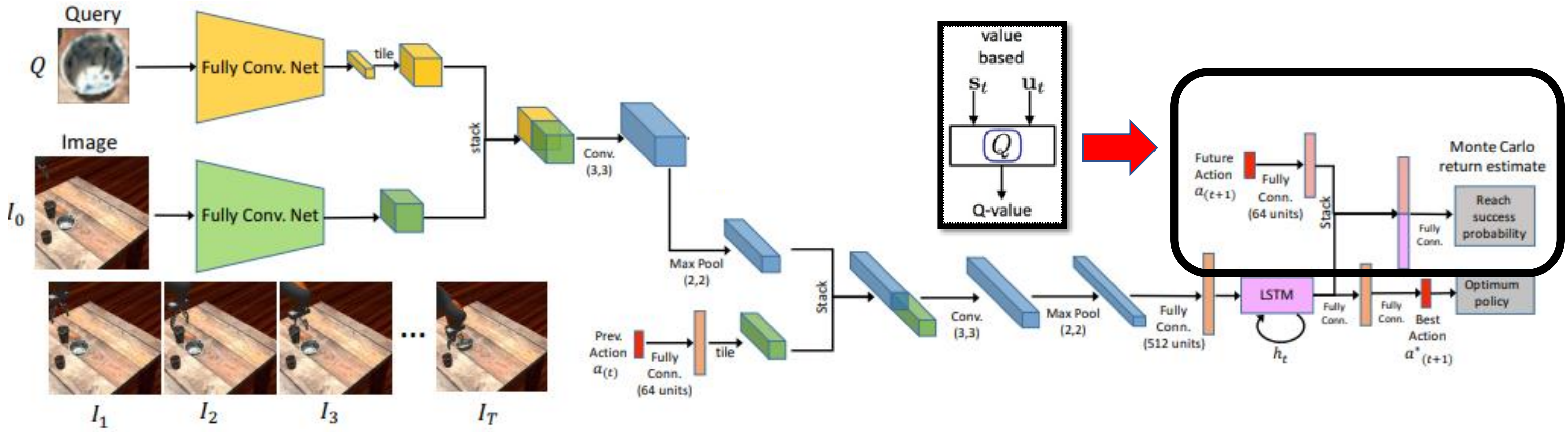
Implicitly learn the environment model

$$\boxed{a_{t+1}, h_{t+1} = \pi_\theta(o_t, a_{t-1}, q, h_t)}$$



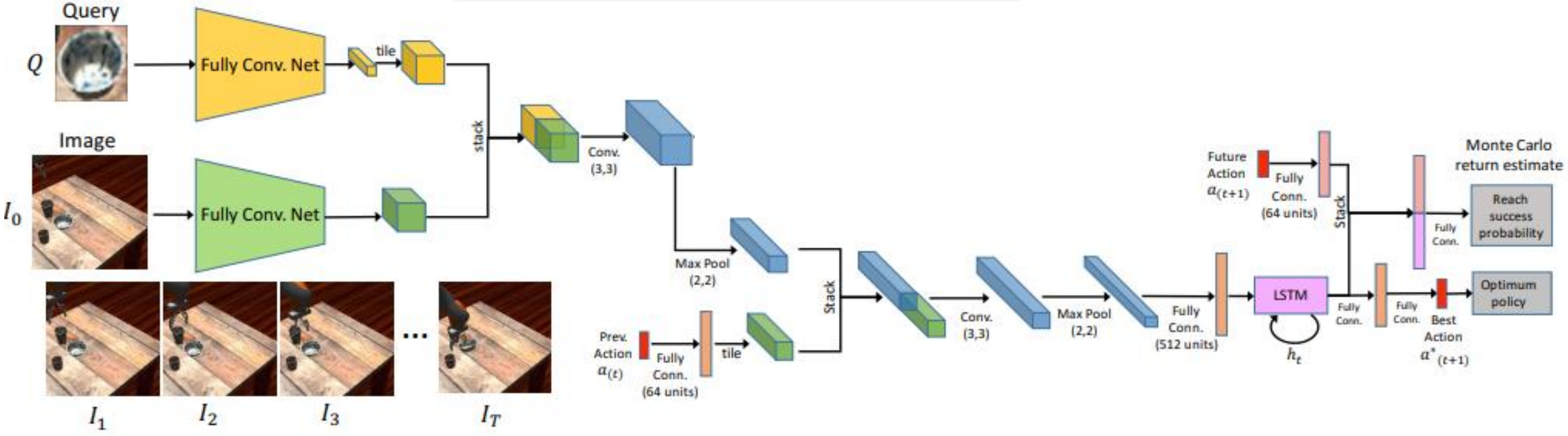
Incorporating value-function learning

$$Q(s_t, a_t) = r(s_t, a_t) + E_{\tau \sim \pi_\theta}\left[\sum_{t'=t+1}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})\right]$$



Learning from Synthetic Demonstration

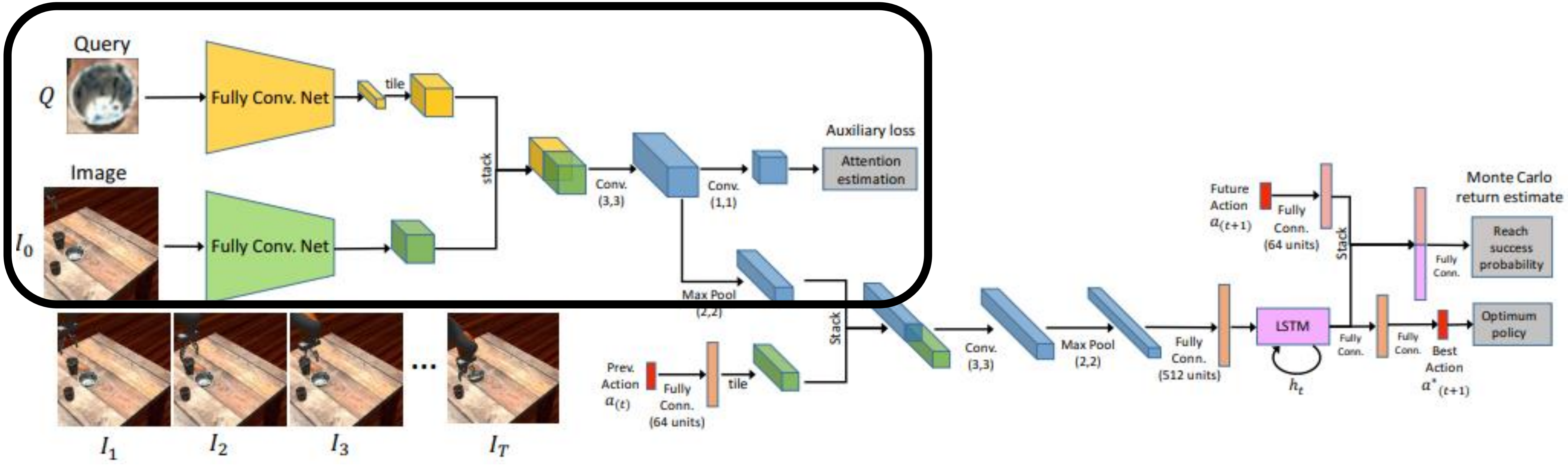
$$Loss = \sum_{t=1}^{T} || \frac{y - x_t}{||y - x_t||} - a_t ||^2$$



Demonstration in simulation environment

Domain adaptation

Finetune on a small amount of real data



Recap

- Visual Servoing can be combined directly with deep features or deep Reinforcement Learning

- A network should implicitly learn the environment model when there is no direct access to it

- The network training should need only a few real robot hours

Reference

- François Chaumette, S. Hutchinson. Visual servo control. In IEEE Robotics and Automation Magazine, 2007
- Quentin Bateux, Eric Marchand, Jürgen Leitner, François Chaumetter, Peter Corke. Training Deep Neural Networks for Visual Servoing. In ICRA, 2018
- Aseem Saxena, Harit Pandya, Gourav Kumar, Ayush Gaud, K. Madhava Krisshna. In ICRA, 2017
- Alex X.Lee, Sergey Levine and Pieter Abbeel. Learning visual servoing with deep features and fitted Q-Iteration. In ICLR, 2017.
- Fereshte Sadeghi, Alexander Toshev, Eric Jang, Sergey Levine. Sim2Real View Invariant Visual Servoing by Recurrent Control. In CVPR 2018