An approach to 2D/3D registration using deep reinforcement learning (ACDDE 2017)

Eungjune Shim,
Center for Bionics, Korea Institute of Science and Technology,
Department of Biomedical Engineering, University of Science and Technology, Seoul, Korea

Youngjun Kim*,
Center for Bionics, Korea Institute of Science and Technology, Seoul, Korea

2017-05

Abstract

Deep Q Learning method is a novel approach to approximate value functions of reinforcement learning. This has been successfully applied to solve problems such as robot control, elevator scheduling, telecommunication networks. We applied this method to a simplified 2D-3D registration problem; Point-based visual servoing simulator. The simulator environment has been virtually organized in three-dimensional space: four three-dimensional target point vectors and two-dimensional correct point vector, and a virtual camera are defined. For each timesteps, camera moves according to the output of a neural network(Q-network), and the 3D target points are projected onto the viewport. The purpose of this simulator is to reduce 2D vector error between target and correct point vectors. The Q-network takes states of current timestep, decide a action, receive rewards and update weights. The actins are defined in six: camera moves forward, backward, top, bottom, right, left. The state is defined as four 2D error vectors. As learning processes, the network moves camera with higher possibility of reducing errors. When using well-trained network, there are several benefits compare to conventional methods, such as random searching algorithms or jacobian matrix estimation. While conventional method requires computation of error variance or matrix inverse in every timestep, our proposed method only requires simple network forwarding to find its solution. Since this method used too much simplified registration environment and camera actions, the performance looks a little bit awkward, but there still are a lot to improve from this approach. Firstly, we can define each step's state with much more complex and sophisticated form by replacing the neural network. There are many Convolutional Neural Networks(CNNs) proposed to handle 2D images, the state can be defined simply using virtual camera's rendered image, not eight-digit 2D point error vectors. Secondly, the camera actions can be more natural and efficient by using the whole output possibility of the network, and combining action. As our proposed method shows considerable benefit over the conventional method, the future work from this approach can be expected to be applicable to real 2D-3D registration works.

Key words: Deep Q Learning, Reinforcement Learning, 2D-3D Registration, Visual Servoing

1. Introduction

Deep Q Learning(DQN) method is a novel approach that has been successfully applied to solve problems such as robot control, elevator scheduling, telecomunication networks.

(Some Related Works)

We applied this method in a simple point-based virtual visual servoing simulator.

In the simulator, we use four 2D vector errors to find 3D camera transformation, This is a basic concept of 2D-3D registration, so this approach shows possibility to extend solving 2D-3D registration problems using DQN.

*Corresponding author email: junekim@kist.re.kr

2. Methods

Virtual point-based servoing simulator is composed of a camera and a plane containing four circles on it in a virtual 3D space. The plane normal and the camera view vectors are parallel to world coordinate's Y-axis. Since no rotation is possible in the condition of this simulator, camera view vector will not be changed. Normally in visual servoing, four point features are extracted from the rendered image by using image processing algorithms. Since our purpose is to test our DQN-based 2D-3D registration algorithm, we decided to skip this process. Instead of that, we pre-defined the 3D positions of four feature points, extract to the 2D rendering viewport by unproject them. This has the same effect as feature detection. This simulator runs on web browser, used javascript-based *WebGL* library named *Three.js*.

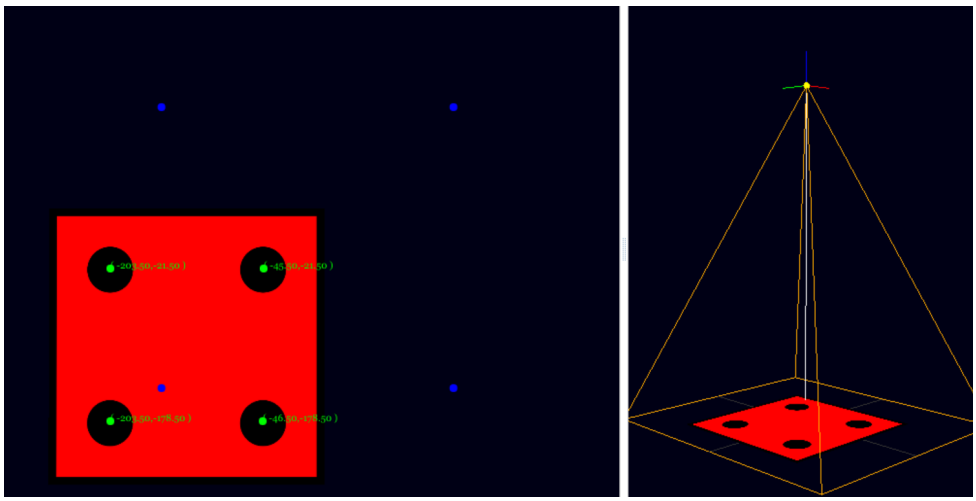

Figure 1. 3D Servoing Environment, renderer image(left), and 3D objects(right)

DQN training is based on Q-learning[1] algorithm, using Q-network for approximating Q-table. Experience replay and target Q are also applied to our method to prevent divergence and stablize the process. For every timestep t, environment ε, composed of statess_t actions a_t, and rewards r_t are defined. The agent of DQN has Q-network 1, predicts rewards of each action values, so user-defined reward can back-propagate and update weight inside the network[2]. In our proposed method, we simply set the state as four error vectors of 2D target points and corresponding ground-truth points. The reward is given according to the variance of error vector size. If the total amount of the error is decreased in x_{t+1}, the agent receives reward of value 1.0, and if the error is increased, the reward is -1.0. The total experience size of sequence is set to 30000 and the training task proceeds when $size(\epsilon) \geq 100$. The action a are consist of six actions as mentioned: forward, backward, up, down, right, left. For each actions, all the distance the camera move is set to 1. The agent interacts with the simulator by selecting actions in a way that maximises future rewards. The Q-network is composed of a simple four-layered neural network: input layer, two fully-connected layers with 50 neurons, and regression output layer Fig. 2.

$$Q(s, a) \Rightarrow r \tag{1}$$

In the training process, camera position is initialized randomly where all four target feature points are visible. If the error value variation E_t between x_{t+1} and x_t is less than 10.0 the camera is repositioned according to the camera position initialization function. After 100 experiences are stacked, the training starts. The agent takes the state x_t and decide action according to DQN network policy; As learning progresses, agent relies more on Q-network then randomly select in order to choose action. The camera performs the action specified by the agent ($camera.GoTo(a_t)$). The reward r_t is fixed according to E_t, and the agent takes reward and updates weights of Q-network ($agent.Backward(r_t)$).

3. Results

(Some Pictures)

(Graph during training)

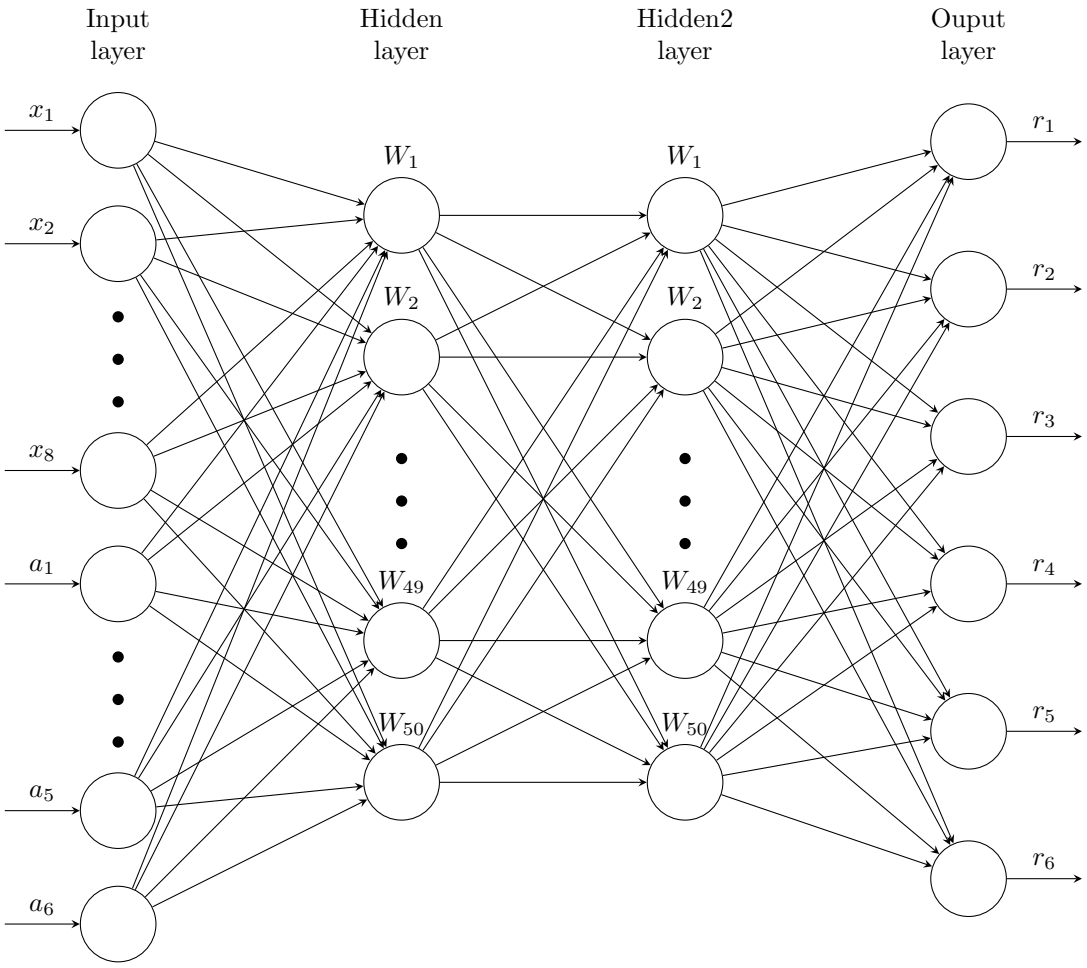

Figure 2. Q-network in the agent. Input layer is composed of states s and actions a. The output layer is rewards

4. Discussion

Our proposed DQN model is trained using 2D feature vector error as the environment ε, decided correct action a for servoing virtual camera in 3D space. Under our DQN policy, the model became totally dependent the Q-network deciding action after ??????? steps. From this point, the network is well-trained as far as possible, so further training is meaningless. This well-trained model can conduct registration from a random position to the optimal position within average of ??? seconds. The visualized camera moved smoothly with almost no unnecessary movement. Compare to conventional method, such as random tree searching algorithm or jacobian matrix estimation, our proposed method has several benefits. First, the amount of calculation is greatly reduced while conventional method requires large amount of computations. Both methods requires calculation of error varation in every timestep. Furthermore, jacobian matrix estimation method requires to calculate inverse (or pseudoinverse) matrix everytime. In random searching algorithm, such as simulated annealing can easily to fall into local minima, and also, there are too much unnecessary movement durign registration. The DQN method requires such calculation only when it is being trained. Once training task is done, this does not require any complex computation, and shows efficient servoing movement. Second, the proposed method is very easy to implement. Once the form of the network, environment, actions, rewards and some parameters are configured, no more difficult tasks are needed except waiting the agent to be trained.

For now, there are several limitations that need to be improved. Firstly, Camera movement is somewhat inefficient like staircase. Because the direction and range in which camera can move is fixed in six actions. The jacobian matrix estimation method can put out all-round 3D translation and rotation vector, this method currently shows much smoother motion during registration. Since all the distance

Algorithm 1 DQN training process for point-based visual servoing

camera.RandomPosition()
for t in T **do**
    a_t = argmax(Q(s, a;θ))
    camera.GoTo(a_t)
    E_t = Error(x_{t+1}) - Error(x_t)
    if E $\geq$ 0 **then**
        r_t = 1.0
    else
        r_t = -1.0
     agent.Backward(r_t)
    if r_t *leq* 10.0 **then** camera.RandomPosition()

the camera move is 1, it is also inefficient in terms of utilization of error sizes. Secondly, states x_t is too simple to apply to real 2D-3D registrations. Our current method defined x_t as four 2D point vector errors, but real 2D-3D registration, larger and much more complex features need to be used to define current state.

Most of this limitations can soon be resolved in the future works. The camera movement problem can be solved by combining actions. Instead of choosing a single action predicted by $argmax(Q(s, a; \theta))$, we can use the whole values in output layer of Q-network, and add the action vectors multiplied by the corresponding predicted reward values. This can give a much higher degree of freedom(DOF) to camera motion. Also, adding rotation action is necessary to perform complicated 2D-3D registration. More sophisticated form of stats x can also be easily defined. This is the greatest potential and advantage of using DQN rather then other methods; Comparing jacobian matrix estimation method, as the feature of input state gets more complicated, the computation and complexity for deciding proper action gets exponentially more difficult. This method can use CNN that can self-study the feature information of input images, this method is extreamly easy to implement, and also, can expect good results. There already are many DQN methods succesfully applied to play games that uses rendered game images as input state, deep CNN as its Q-networks[2]. Setting stochastic reward can also efficiently utilizes the size of error variation, and reduces registraiton time. To sum it up, our future DQN model uses deep CNN for its value network. takes rendered image of each timestep as input state, predicts actions, and perform registration by combining whole output Q-network layer.

5. Conclusion

We have proposed and developed a DQN method for 2D-3D registartion. This method is successfuly applied to optimize and solve the point-based visual servoing simulator. This simulator takes 2D feature states to perform in 3D, this basically can be regarded as very simplified 2D-3D registration simulator. Although this is simplified problem, the proposed approach has its strength in terms of solving similar but more complex problems compare to conventional registration methods; For example, jacobian matrix estimation method requires exponential computation and high difficulty as the problem becomes more complicated. Also, most of searching algorithm such as simulated annealing, heel climbing methods for optimizing 2D-3D registration requires to calculate the similarity of two states, and the complexity of two states gets higher, the amount of computation also gets larger. While the DQN only forwards input states to the network, once it is well-trained.

Acknowledgement

This research was supported by the KIST institutional program (??????, ???????).

References

[1] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602.