Deep reinforcement learning for calibration of 2D-3D registration

Eungjune Shim[1], Hannah Kim[1], Laehyun Kim[2], Youngjun Kim[1,*]
[1]Division of Bio-Medical Science & Technology, KIST School,
Korea University of Science and Technology, Seoul 02792, Republic of Korea
[2]Center for Bionics, Korea Institute of Science and Technology, Seoul 02792, Republic of Korea

2017-05

Abstract

The Deep Q-Learning method is a novel approach to approximate the value functions of reinforcement learning. It has been successfully applied to solve problems in robot control, elevator scheduling, and telecommunication networks. We applied this method to a simplified 2D-3D registration problem, a point-based visual servoing simulator. The simulator environment was virtually organized in 3D space: four 3D target point vectors, a 2D correct point vector, and a virtual camera are defined. For each time step, the 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 errors between target and correct point vectors. The Q-network takes the states of the current time step, selects a action, receives rewards, and updates weights. The actions are defined in six directions: the camera moves forward, backward, up, down, right, left. The state is defined as four 2D error vectors. As learning progresses, the network moves the camera to increase to probability of reducing errors. There are several advantages to using a well-trained network compared to conventional methods such as random search algorithms or Jacobian matrix estimation. While conventional methods require the computation of error variations or inverse matrix in every time step, our proposed method only requires simple network forwarding to find its solution. However, since this method used an oversimplified registration environment and camera actions, its performance was a bit awkward; but there still are a lot to improve from this approach. First, we can define each step's state with much more complex and sophisticated forms by replacing the neural network. Many convolutional neural networks have been proposed to handle 2D images. The state can be defined by simply using the virtual camera's rendered image instead of eight-digit 2D point error vectors. Second, the camera actions can be more natural and efficient by using the whole output possibility of the network and combining actions. As our proposed method shows considerable advantages over conventional methods, future work will eventually make it applicable to real 2D-3D registration processes.

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

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

1. Introduction

There are various methods that use 2D features to perform in 3D space. As image processing and 3D technologies evolve and computing power increases, many related works using this approach have become possible. Some methods utilize a monoscopic camera for 3D scanning[1], camera calibration[2], motion capture[3], or visual servoing[4], and all are using 2D features to somehow perform in 3D space. In this paper, we proposed an approach to solve 2D-3D registration problems. 2D-3D registration also uses 2D features to conduct registration in 3D. This method is largely used in the medical field. Registering 3D dicom volume obtained from computed tomography or magnetic resonance imaging to a 2D X-ray image can produce valuable information in diagnosing patient conditions. For example, when diagnosing an orthopedic patient's condition, well-aligned 3D data can be used to visualize 3D data with information that can only be obtained from X-ray images [5, 6].
Many optimization approaches have been proposed and developed to apply to this registration task. When 2D features are simple as points, lines, or binary or faces, the Jacobian matrix estimation[7] method is known to be robust and is frequently used. This method extracts meaningful features from the target image, defines and calculates the error between the target and correct 2D features, and performs calibration. This method requires a considerable amount of 2D data preprocessing and feature matching in every time step, as well as large inverse matrix operations. Thus, this method is not applicable to most 2D-3D registration processes that use various 2D images such as x-ray radiographs. Many image comparison methods have been proposed, such as edge-filtering[8], to preprocess image data for this task. Too much simplified image feature can possibly lose most of information, it might loss most of valuable information of image resulting inefficiency. For these reasons, the Jacobian matrix estimation method is unsuitable for most 2D-3D registration processes. There are also plenty of methods to define errors between 2D features [9] and optimize errors. Optimization is normally conducted using iterative methods, such as simulated annealing algorithms[8]. However, this optimization task also has limitations such as local minima or unnecessary searches.
In recent years, deep learning methods have attracted interest for solving optimization problems in image or natural language processing, Most state-of-the-art methods are based on deep learning[10, 11]. For problems with very a large search space such as game playing or 2D-3D registration, deep reinforcement learning methods are often used. The current state-of-art reinforcement learning method is Deep Q-Network (DQN)[12] learning proposed by *DeepMind*. DQN has been successfully applied to learn and perform complex tasks such as playing Atari games[13]. We also have chosen the DQN method to conduct 2D-3D registration. Since our method is an approach to determine whether DQN can be applied to registration problems, we made our models and problems as simple as possible. We successfully applied DQN to operate a four-point-based visual servoing simulator, recorded learning outcomes and effects, and proposed a strong future research plan based on this method.

2. Methods

The virtual point-based servoing simulator is composed of a camera and a plane containing four circles in a virtual 3D space. The plane normal and camera view vectors are parallel to the y-axis of the world coordinate system. Since rotation is not possible under the conditions of this simulator, the camera view vector will not be changed. In visual servoing, four feature points are normally extracted from the rendered image using image processing algorithms. Since our purpose is to test our DQN-based 2D-3D registration algorithm, we decided to skip this process. Instead, we pre-defined the 3D positions of the four feature points and extracted these to the 2D rendering viewport by unprojecting them; this has the same effect as feature detection. The simulator runs on a web browser and used a JavaScript-based *WebGL* library named *Three.js*.

DQN training is based on a Q-learning[14] algorithm that uses a Q-network for approximating the Q-table. Experience replay and target Q are also applied to our method to prevent divergence and stabilize the process. For every time step t, the environment ε, composed of states x_t, actions a_t, and rewards r_t are defined. The DQN agent has a Q-network (**??**) that predicts the rewards for each action value(Equation 1) so that user-defined rewards can backpropagate and update weights inside the network[13]. In our proposed method, we simply set the state as four error vectors of 2D target points and the corresponding ground truth points. The reward is given according to variations in error vector size. If the total value of the error decreases in x_{t+1}, the agent receives a reward valued at 1.0; if the error increases, the reward is -1.0. The action a moves in six directions: forward, backward, up, down, right, left. For each action, all

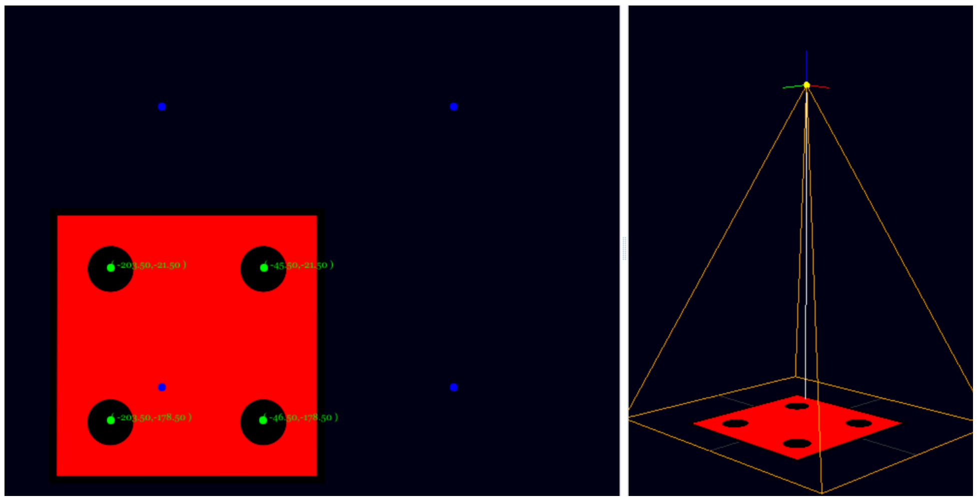

Figure 1: 3D servoing environment: renderer image(left) and 3D objects(right)

the distance the camera move is set to 1. The agent interacts with the simulator by selecting actions in a way that maximizes future rewards. The Q-network is composed of a simple four-layer neural network: an input layer, two fully-connected layers with 50 neurons, and a regression output layer Fig. 2.

$$Q(x, a; \theta) \Rightarrow r \tag{1}$$

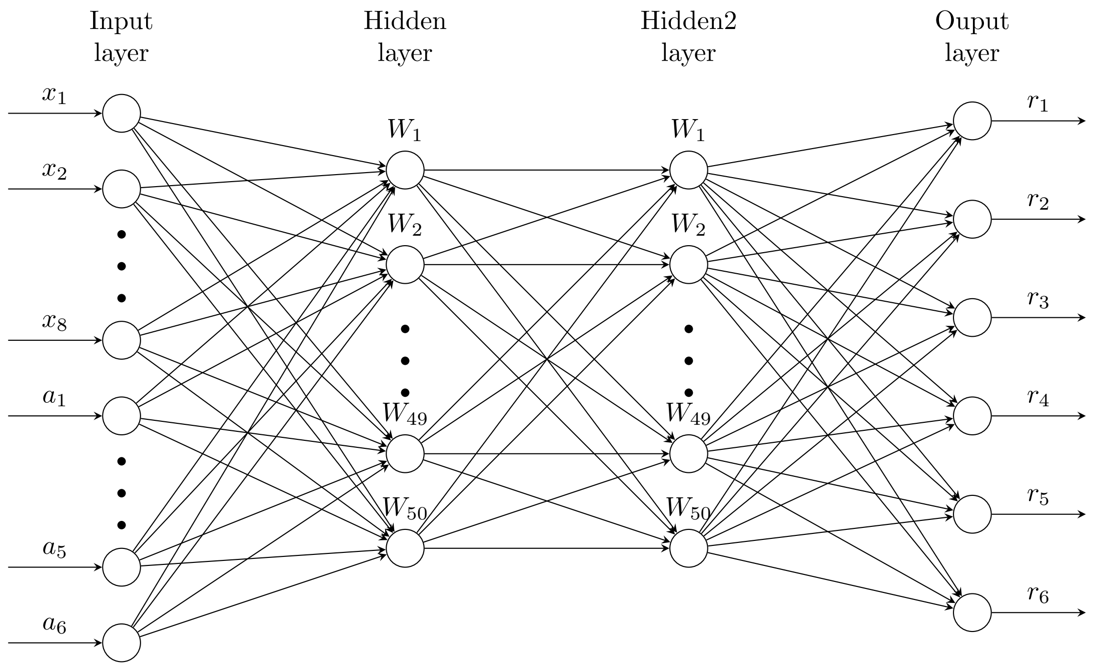

Figure 2: Q-network in the agent. The input layer is composed of states s and actions a. The output layer is composed of rewards

In the training process, the 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 decides on the action according to the DQN network policy. As learning progresses, the exploration epsilon ϵ slightly decreases, and this value causes the agent to increasingly rely on the Q-network instead of random selection when choosing the action. We set the minimum ϵ_{min} to 0.005 and when (eq??), the learning task is done. Setting ϵ_{min} to zero can make our process easily fall into the local minima. Thus, we opted to use this value to choose a random action even when running a calibration task using the trained network. 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 the reward and updates the weights of the Q-network ($agent.Backward(r_t)$). The epoch of the training session is set from the state when the camera is repositioned randomly until the optimal transformation that sets error values to less than 10.0 is reached.

$$\epsilon \leq \epsilon_{min} \tag{2}$$

Algorithm 1 DQN training process for point-based visual servoing

```
 1: camera.RandomPosition()
 2: for $t$ in $T$ do
 3:     $a_t$ = argmax(Q(s, a;$\theta$))
 4:     $cam_{pos} = cam_{pos} + a_t$
 5:     $E_t$ = Error($x_{t+1}$) - Error($x_t$)
 6:     if E $\geq$ 0 then
 7:         $r_t$ = 1.0
 8:     else
 9:         $r_t$ = -1.0
        agent.Backward($r_t$)
10:     if $r_t$ leq 10.0 then $cam_{pos} = random$
```

3. Results

The duration of each epoch during training is plotted on Fig. 3. A total of 545 epochs and 199,108 steps were conducted to complete the training, where equation 2 was satisfied. The average Q-learning loss was 0.7580. Fig. 3 (a) shows that it took 492.17 s to finish epoch 0. From the second epoch, nothing exceeded 100.0 s to reach the optimal state. From epoch 76, the simulator reached the optimal states within approximately 10.0 s (Fig. 3, (b)). Fig. 4 (a) shows the optimization time distribution of 30 epochs using the well-trained DQN model. It took an average 2.28 s to finish a single epoch starting from a random camera position. Each epoch's error rate variation is plotted on Fig. 4 (b). Fig. 5 shows the decreasing trend of the error rate in each epoch. As training progressed, the model not only reached its optimal state faster, but also showed less unnecessary movements, as seen in the error rate variation graphs that appeared softer in higher epochs.

4. Discussion

Our proposed DQN model was trained using 2D feature vector error as the state x and selected the correct action $a = argmax(Q(s, a; \theta))$ for the visual servoing camera in 3D space. Under our DQN policy, the model finished training after 545 epochs. The well-trained model can conduct registration from a random position to the optimal state within an average of 2.28 s. The visualized camera moved smoothly with almost no unnecessary movement. Compared to conventional methods such as random search tree algorithms or Jacobian matrix estimation, our proposed method has several advantages. First, the amount of calculations is greatly reduced. Both conventional methods require the calculation of error variations in every time step. Furthermore, the Jacobian matrix estimation method requires calculation of the inverse (or pseudoinverse) matrix every time. Random tree search algorithms such as simulated annealing can easily to fall into the local minima, and there are also excessive unnecessary movements during registration. The DQN method requires such calculations only during training. Once training is completed, it no longer requires complex computations 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, difficult tasks are eliminated except waiting for the agent to be trained.

However, there are currently several limitations that need to be addressed. First, camera movement is somewhat inefficient because the direction and range of movement is fixed to six actions. The Jacobian

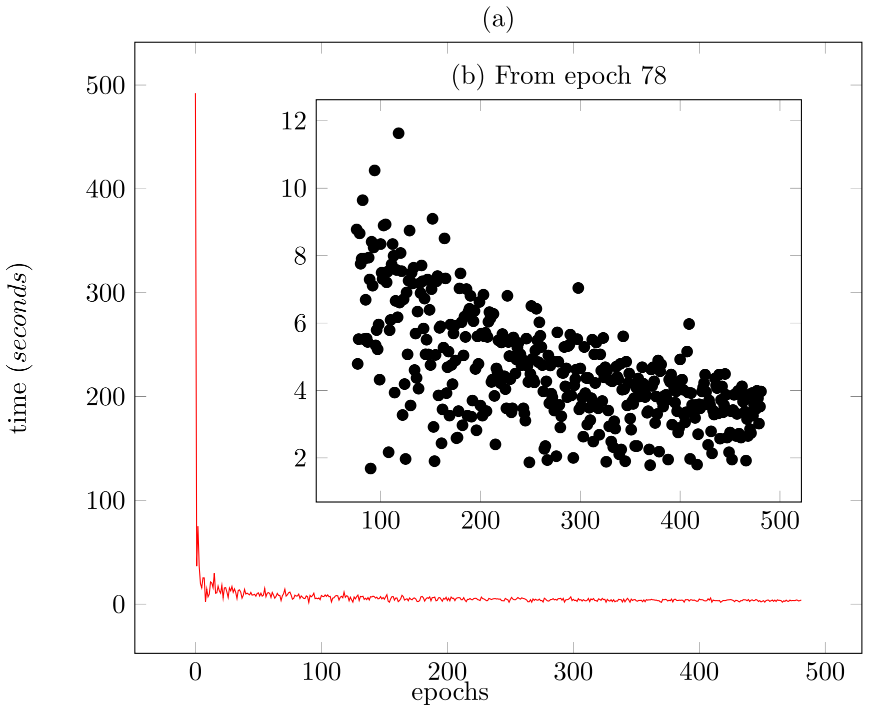

Figure 3: Searching time during learning progress

matrix estimation method can generate an all-round 3D translation and rotation vector, and currently has much smoother motion during registration. Since all the distance the camera move is 1, it is also inefficient in terms of utilization of error sizes. Second, state x_t is too simple to apply to real 2D-3D registrations. Although our current method defined x_t as four 2D point vector errors, real 2D-3D registration has larger and more complex features that need to be used to define current state.

Most of these limitations can 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 the output layer of the Q-network, and add the action vectors multiplied by the corresponding predicted reward values. This can provide a higher degree of freedom to camera motion. The addition of rotational action is also necessary to perform complicated 2D-3D registration. A more sophisticated form of state x can also be easily defined. This is the greatest advantage of using DQN rather than other methods. Compared to the Jacobian matrix estimation method, as the features of the input state become more complicated, the computation and complexity of deciding the proper action becomes exponentially more difficult. This method can use a convolutional neural network (CNN) that can self-study the feature information of input images. This method is also extremely easy to implement and can provide good results. Many DQN methods have been successfully applied to playing games that use rendered game images as the input state and deep CNN as its Q-networks[13]. Setting stochastic rewards can also efficiently utilize the size of error variations, and reduce registration time. To summarize, our future DQN model uses a deep CNN for its value network. It takes the rendered image of each time step as the input state, predicts actions, and performs registration by combining the whole output Q-network layer. The reward for this registration method could be the root mean square error between the pre-defined optimal transformation and current transformation. This model is expected to perform very well compared to most conventional methods using rigid single-view 2D-3D registration

5. Conclusion

We have proposed and developed a DQN method for 2D-3D registration. This method was successfully applied to optimize and solve a point-based visual servoing simulator. Since this simulator takes 2D feature states to perform in 3D, it can be regarded as a very simplified 2D-3D registration simulator. Although this is a simplified problem, the strength of the proposed approach is its ability to solve similar

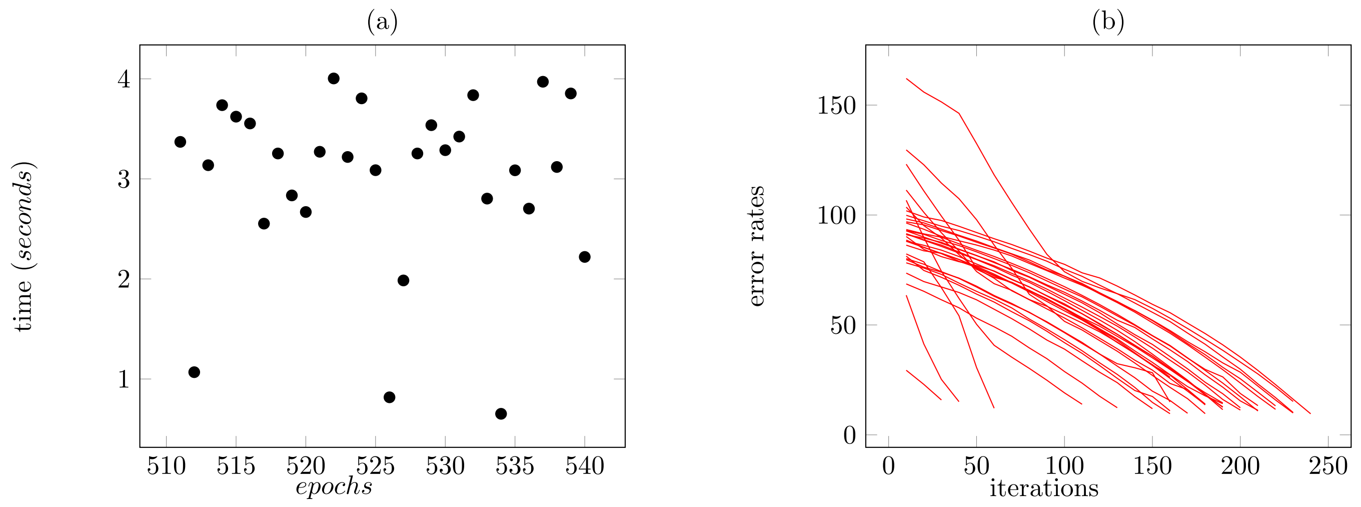

Figure 4: Well-trained network calibration error variation

but more complex problems compared to conventional registration methods. For example, the Jacobian matrix estimation method requires exponential computations and high difficulty as the problem becomes more complicated. Most search algorithms such as simulated annealing and heel climbing methods for optimizing 2D-3D registration require calculation of the similarity between two states. As the complexity of two states increases, the amount of computations also increases. Although the well-trained DQN model only forwards input states to the network to determine proper actions, its inherent advantages will enable the resolution of most limitations in the near future, when this approach is applied to higher-order 2D-3D registration problems.

Acknowledgement

This research was supported by the KIST institutional program (2E26880, 2V05430).

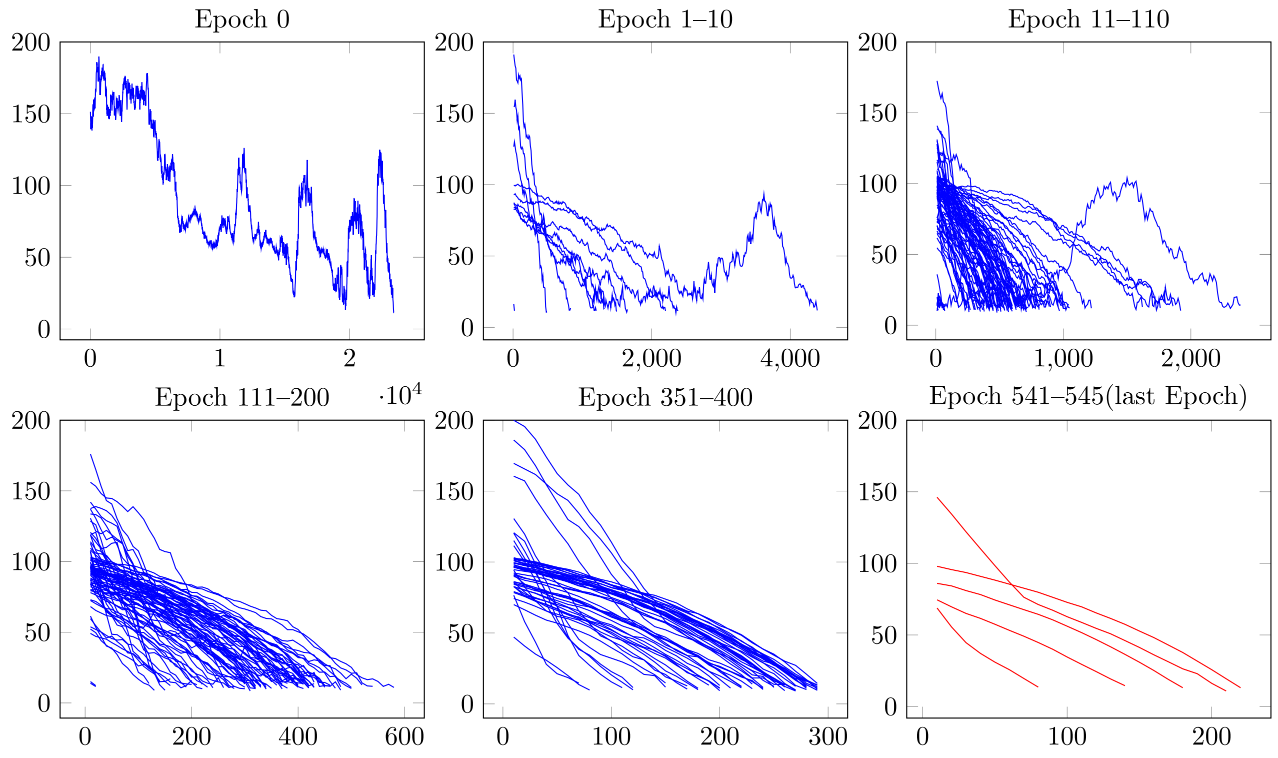

Figure 5: Error rate variation. Error rate(y-axis), and iterations(x-axis)

References

[1] W. Niem, J. Wingbermuhle, Automatic reconstruction of 3d objects using a mobile monoscopic camera, in: 3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in, IEEE, 1997, pp. 173–180.

[2] D. Li, J. Tian, An accurate calibration method for a camera with telecentric lenses, Optics and Lasers in Engineering 51 (5) (2013) 538–541.

[3] H. Wang, H.-T. Li, S. Manjunath, Real-time capturing and generating stereo images and videos with a monoscopic low power mobile device, uS Patent 8,970,680 (Mar. 3 2015).

[4] F. Chaumette, S. Hutchinson, P. Corke, Visual servoing, in: Springer Handbook of Robotics, Springer, 2016, pp. 841–866.

[5] J.-M. Kim, S.-H. Hong, B.-S. Lee, D.-E. Kim, K.-A. Kim, S.-I. Bin, Femoral shaft bowing in the coronal plane has more significant effect on the coronal alignment of tka than proximal or distal variations of femoral shape, Knee Surgery, Sports Traumatology, Arthroscopy 23 (7) (2015) 1936–1942.

[6] K. Radtke, C. Becher, Y. Noll, S. Ostermeier, Effect of limb rotation on radiographic alignment in total knee arthroplasties, Archives of orthopaedic and trauma surgery 130 (4) (2010) 451–457.

[7] D. Kosmopoulos, Robust jacobian matrix estimation for image-based visual servoing, Robotics and Computer-Integrated Manufacturing 27 (1) (2011) 82–87.

[8] Y. Kim, K.-I. Kim, J. hyeok Choi, K. Lee, Novel methods for 3d postoperative analysis of total knee arthroplasty using 2d–3d image registration, Clinical Biomechanics 26 (4) (2011) 384–391.

[9] C. Gendrin, P. Markelj, S. A. Pawiro, J. Spoerk, C. Bloch, C. Weber, M. Figl, H. Bergmann, W. Birkfellner, B. Likar, et al., Validation for 2d/3d registration ii: The comparison of intensity-and gradient-based merit functions using a new gold standard data set, Medical physics 38 (3) (2011) 1491–1502.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Transactions on Image Processing.

[11] Y. Goldberg, Neural network methods for natural language processing, Synthesis Lectures on Human Language Technologies 10 (1) (2017) 1–309.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[13] 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.

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