EasyChair Preprint

№ 8178

Reinforcement Learning for Path Generation for Surgical Robot Maneuver

Junhong Chen, Zeyu Wang, Ruiqi Zhu, Ruiyang Zhang,
Weibang Bai and Benny Lo

June 1, 2022

Reinforcement Learning for Path Generation for Surgical Robot Maneuver

Junhong Chen[1], Zeyu Wang[1], Ruiqi Zhu[2], Ruiyang Zhang[1], Weibang Bai[1], and Benny Lo, *Senior Member, IEEE*[1]

[1]*Hamlyn Centre for Robotic Surgery* [2]*King's College London*
junhong.chen16@imperial.ac.uk

INTRODUCTION

In the last decades, surgical robots have been widely used in Robot-Assisted Minimally Invasive Surgery (RAMIS), which benefits surgeons by reducing their burdens and leads to safer operations. However, RAMIS tasks are still mainly relied on surgeon's control, thus the performance of a task mostly depends on the level and proficiency of a surgeon, and prone to human errors due to fatigue. However, despite their extensive experience, operators often make small mistakes and corrections during the tasks. Therefore, their kinematics data usually contains small differences to an ideal trajectory. To make full use of Learning from Demonstration(LfD) as well as reduce the dependence on kinematics data, a novel path generation method based on reinforcement learning(RL) is proposed in this paper. The da Vinci Research Kit, as the open-source robotic platform, is used to validate the model. In addition simulation is used for model training and early validation.

The contributions in this work: a) a quick path generation method for transferring tasks via RL; b) a new training strategy for surgical tasks based on surgeme[1]; c) the validation results of the proposed through transferring the learnt model from simulation to real robots.

MATERIALS AND METHODS

The overview of the proposed path generation for automatic transferring is shown in Fig.1. In most surgical tasks, transferring objects or purely moving tools are inevitable. Peg transfer is one of the standard tasks that are suitable for transferring training. In order to train, develop and test the peg transfer task, Asynchronous Multi-Body Framework (AMBF)[2] is used as the simulated environment based on our previous work[3], and then validate on the da Vinci Robot. Fig.2 shows the simulation environment (a) and da Vinci Robot (b) used in validation of the proposed. In Peg transfer task, the transferring process happens when the gripper is carrying the peg or moving tools to reach the peg. Therefore, the transferring would started from a random pillar with the peg to another random pillar as the target. The starting and targeting points are defined as:

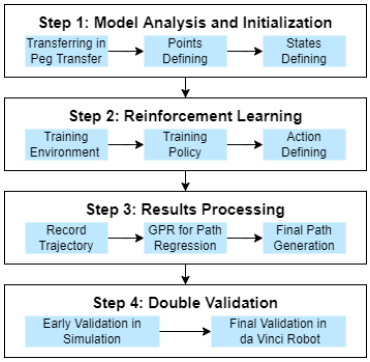

Fig. 1 Framework of the proposed path generation method

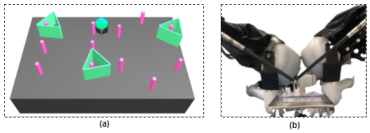

Fig. 2 Overview of the training environment (a) and the validation environment (b)

$$\begin{cases} P_0 = [x_0, y_0, z_0] \\ P_e = [x_e, y_e, z_e] \end{cases} \tag{1}$$

The gripper state contains its position as well as its orientation. Using quaternion to represent its orientation, and its state in time t is defined as:

$$S(t) = \begin{matrix} [p_x(t), p_y(t), p_z(t), \\ o_x(t), o_y(t), o_z(t), o_w(t)] \end{matrix} \tag{2}$$

Both the points and gripper state are in the global frame for easier processing. And the boundary for training depends on the size of pegboard. Here we have: $x \in [-0.72, 0.56], y \in [-0.72, 12], z \in [-0.5, 0.5]$. For training, the starting points are randomly picked from the range$[-0.1, 0.1]$ with equal probability in all x,y,z axes, while the targeting points are chosen randomly based on the pillar's position, a fixed coordinate throughout the training. Then three targeting points are chosen based on different distances between starting and targeting points, from close to far. For practical consideration, both gripper and peg are objects with volume, therefore, a round shape with radius $r = 0.08$ is defined as the peg. After initializing the environment, the action ΔS is defined as the variation of state of the gripper measured in the global frame. The action of the gripper can be represented as:

TABLE I Metrics in Simulation and Real robot

	S-Left	S-Mid	S-Right	R-Left	R-Mid	R-Right
M(m)	0.636	0.290	1.063	0.0440	0.0298	0.0327
t(s)	0.53	0.2	0.6	1.17	0.87	0.97
A(m/s)	1.200	1.446	1.772	0.0376	0.0342	0.0337

$$\Delta S(t) = S(t) - S(t-1) \tag{3}$$

This action will be the output command from the training model to control the gripper. Since the output control command is a vector with continuous values, thus the training policy is designed with the algorithm, Deep Deterministic Policy Gradients (DDPG)[4], which is widely used for continuous control[5]. The rewards are defined as:

$$R = \begin{cases} -1.0 & \text{if over boundary} \\ -\frac{d}{D} + k & \text{if moving away from the target} \\ -\frac{d}{D} & \text{if moving close to the target} \\ 1.0 & \text{if reaching the target} \end{cases} \tag{4}$$

where d is the distance between the current gripper position and the targeting point using Euclidean norm. D and k are constant for adjusting the rewards. The value of D is chosen empirically, otherwise, the gripper may collide or reach the boundary, and k must be a small negative, together with D to control the gripper. In our model, $D = 18$ and $k = -0.2$. Also, when t reaches maximum time steps, $R = -1.0$. Unlike other reward functions, like [5], except the success or failure cases, the other two rewards aim to guide the agent following the way of approaching the target, which efficiently increases the training and path generation.

During the training, success happens when the gripper reaches the targeting point without collision or out-of-bounds. Once the simulation consecutively succeeds 100 times, the model stops training and records the trajectory in the simulation. These trajectories are treated as demonstrations, each one of them indicates one path to the target. To extract their features and generate the final path, Gaussian Process Regression (GPR) is used. With uncertainty of the training results, GPR will compute the final path as required and reduce errors. After that, further processing can ensure the endpoint will reach the close proximity to the targeting point. Through this method, the path generation of other tasks, like needle passing and pattern cutting could follow the training strategy. From the framework view, the only difference would be the Step 1 model analysis. For example, needle passing could be divided into several targets while pattern cutting could adjust the rewards to follow the cutting curve.

RESULTS

The training curve is shown with average reward curve in Fig.3(a). The final success rate is close to 100%, while the average reward increases to its highest possible value. The final generated path is shown in Fig.3(b). In the figure, Red line indicates the final path, circle represents the size of a peg and all blue points are point path before GPR process. The generated path will certainly reach the allowing range of targeting point. Fig.4 shows the frames of gripper approaching

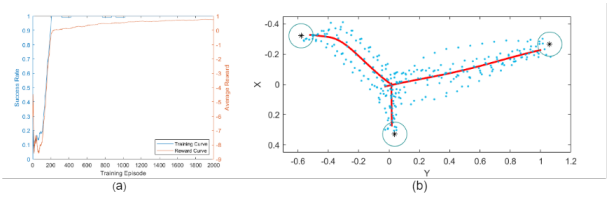

Fig. 3 a) Training and Reward curve, b) Generated path

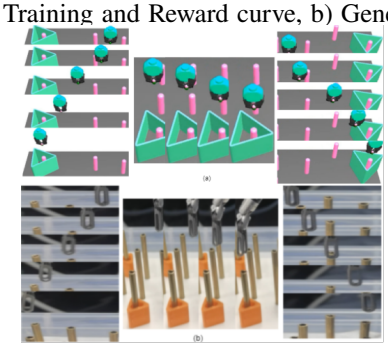

Fig. 4 a) Simulation validation frames, b) da Vinci validation frames

the target based on the similar generated path in this framework. In the real-world validation, the initial position of end-effector tools are randomized. Fig.4 shows the frames of gripper approaching the target pillar when tested with the real da Vinci robot. Table.I shows the metrics: i)path-length (M-m); ii) completion time (t-s); iii) average speed (A-m/s) during validations in both environments, simulation(S) and real(R).

DISCUSSION

In this paper, we present a novel method for path generation to conduct automatic transferring of pegs. The results of the generated path and their validation verified that the proposed path generation method could automate this repetitive surgical task. Unlike other research like [5], the method used after RL part aims for combining the strength of both LfD and RL. If integrated this work with our previous work on shared control[3], a surgeon will only need to pinpoint a few locations, the robot can then complete the whole peg transfer task.

REFERENCES

[1] H. C. Lin, I. Shafran, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, vol. 11, no. 5, pp. 220–230, 2006.

[2] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 1875–1882.

[3] J. Chen, D. Zhang, A. Munawar, R. Zhu, B. Lo, G. S. Fischer, and G.-Z. Yang, "Supervised semi-autonomous control for surgical robot based on banoian optimization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2943–2949.

[4] 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 *ICLR (Poster)*, 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[5] Z. Chiu, F. Richter, E. K. Funk, R. K. Orosco, and M. C. Yip, "Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning," *CoRR*, vol. abs/2011.04813, 2020. [Online]. Available: https://arxiv.org/abs/2011.04813