CONTINUOUS ACTIONS CONTROL OF ROBOTIC MEDICAL DEVICES USING DEEP REINFORCEMENT LEARNING:
A LAPAROSCOPIC SURGERY CASE STUDY.

BY: HOANG NGOC SON (SEAN)

DATE: JULY 2020

1

AGENDA

2

1. INTRODUCTION

- Automation of healthcare through machine learning
  - Medical image diagnosis
  - Drug discovery
- Self-controlling robotics
  - Self-driving car
  - Autonomous robotics (manufacturing robot arms)

➢ Autonomous controlling of robotic medical devices

➢ Case study: controlling of laparoscopic surgery apparatus (robot arm)

3

2. LITERATURE REVIEW

- Machine learning in medical sector
- Laparoscopic surgery
- Reinforcement learning
- Reinforcement learning applied in (robotic) laparoscopic surgery

4

2.LITERATURE REVIEW

Machine Learning in Medical Sector

- Diagnosis
  - Medical Radiology
  - Symptom-based Decision System
- Prognosis
  - Drug Discovery
  - Genome Sequencing
- Healthcare
  - Virtual Companion (chat-bot)
  - Vital Sign Monitoring

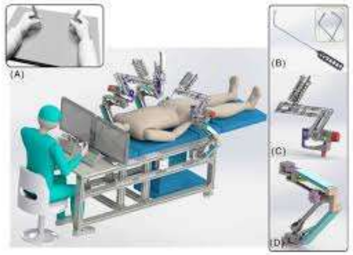
A more direct potential application: Laparoscopic surgery

5

2. LITERATURE REVIEW

Laparoscopic Surgery

Laparoscopy (keyhole surgery)	Abdomen and pelvis areas
	Small, precise incisions
Laparoscope	Thin, long tube
	High intensity light
	High resolution camera
Characteristics	Video operation
	Shorter hospitalization
	Less bleeding and scarring



https://ieeexplore.ieee.org/abstract/document/8441801/

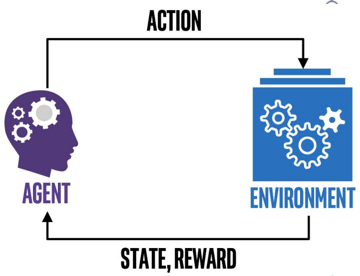
6

2. LITERATURE REVIEW

Reinforcement Learning

- Policy → Action
- Current State + Action → Next State + Reward
- Objective: Maximize Expected Returns
- How: Improve Policy
- May or may not know environment dynamics

Explicit Policy	Implicit Policy
• Learn environment representation • Action based on current representation • PPO, TRPO	• Learn environment interaction • Action based on perceived benefits • DQN



https://nervanasystems.github.io/coach/

7

2. LITERATURE REVIEW

Reinforcement Learning

- Returns:

$$G = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})}}\left[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,s_{t+1})\right]$$

- Action value:

$$Q^{\pi}(s,a) = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})},\, a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,a_t,s_{t+1})\middle| s_t = s, a_t = a\right]$$

- State value:

$$V^{\pi}(s) = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})},\, a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,a_t,s_{t+1})\middle| s_t = s\right]$$

- Advantage:

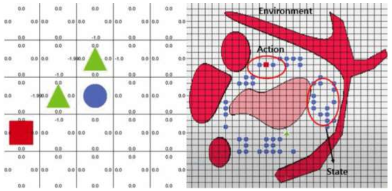
$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

8

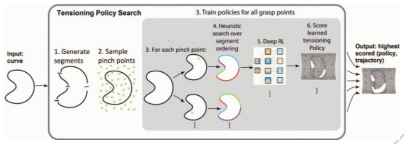
2. LITERATURE REVIEW

Reinforcement Learning Applied in Laparoscopic Surgery

- Positioning of instruments in preoperative procedures (robot arm, camera)
- Training/collaborating with human operators
- Path-planning of surgical robot



https://ieeexplore.ieee.org/abstract/document/8441801

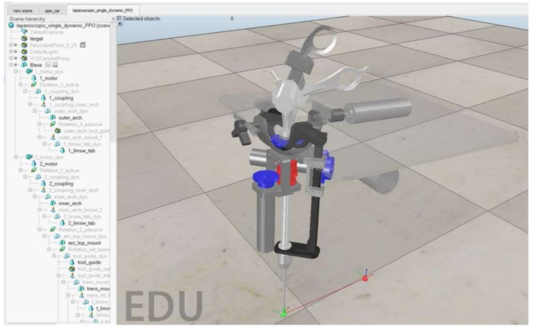


https://ieeexplore.ieee.org/abstract/document/7989275

9

3. V-REP ENVIRONMENTS

Laparoscopic Simulation



10

3. V-REP ENVIRONMENTS

Laparoscopic Simulation

- State vector:

$$\begin{bmatrix} [Previous\ action] \\ [Tip\ position] \\ [Target\ position] \end{bmatrix} \equiv \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_9 \end{bmatrix}$$

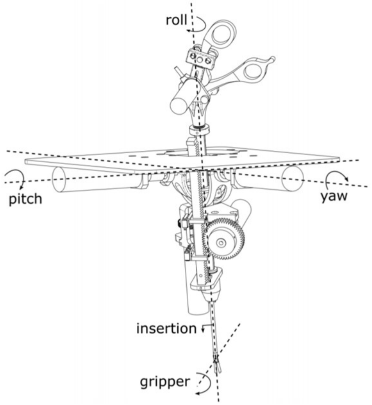
- Action vector:

$$\begin{bmatrix} pitch \\ yaw \\ insertion \end{bmatrix} \equiv \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- Reward:

$$r_t = \begin{cases} 0.05 + 0.1e^{-d_t} & if\ d_t \leq d_{t-1} \\ -0.05 & if\ d_t > d_{t-1} \\ 200 & if\ d_t \leq 0.015 \end{cases}$$

d_t *is the distance between the tip and the target position*



11

3. V-REP ENVIRONMENTS

Car Simulation

- State vector:

$$\begin{bmatrix} [Collision\ infos] \\ [Previous\ actions] \\ [Obstacles\ positions] \\ [Car\ absolute\ orientation] \\ [Direction\ to\ target] \\ [Car\ position] \\ [Target\ position] \end{bmatrix} \equiv \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{22} \end{bmatrix}$$

- Action vector:

$$\begin{bmatrix} Left\ joint \\ Right\ joint \end{bmatrix} \equiv \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

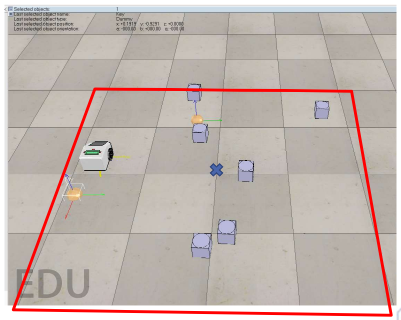
- Reward:

If early termination:

$$r_t = -50$$

else:

$$r_t = -0.5(Col) + \begin{cases} 0.05(2.7 - d_t) & if\ d_t \leq d_{t-1} \\ -0.04 & if\ d_t > d_{t-1} \\ 80 & if\ d_t \leq 0.05 \end{cases}$$



12

4. LEARNING ALGORITHM

Policy Gradient

Returns: $G = E_{s_{t+1}\sim P_{a(s_t,s_{t+1})}}[\sum_{t=0}^{\infty}\gamma^t R_{a_t}(s_t,s_{t+1})] = E_\tau[R(\tau)]$

- Directly evaluating G as an objective function
- Represent policy as a function approximator (e.g. neural network): $\pi(a|s)$
- Use gradient ascent and backpropagation to improve $\pi(a|s)$

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G\nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

13

4. LEARNING ALGORITHM

Proximal Policy Optimization (PPO)

- Limit the deviation of new policy
  - ➢Avoid destructive updates
- Use Advantage instead of Monte-Carlo return
  - ➢Reduce gradient variance
- Can monotonically improve policy

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

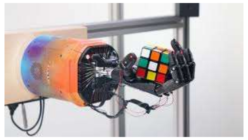
  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

$$\boxed{r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}}$$

end for

http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_13_advanced_pg.pdf



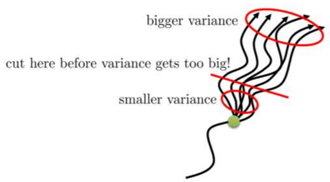
14

4. LEARNING ALGORITHM

Generalized Advantage Estimates (GAE)

$$\hat{A}_t^{GAE} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V$$
$$\delta_t^V = r_t + \gamma V_{s_{t+1}} - V_{s_t}$$

- Similar approach with TD(λ)
- Bias-Variance trade-off
  - λ = 0: 1-step return
  - λ = 1: Monte-Carlo return with baseline
- Only estimate value function (V)



http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-6.pdf

15

4. LEARNING ALGORITHM

Proposed Modifications

- Avoid bias

$$\hat{A}'_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^A$$

- Use advantage definition

$$\delta_t^A = Q_{s_t} - V_{s_t}$$

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right)\\ &= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \ldots)\\ &= (1-\lambda)(\delta_t^V(1+\lambda+\lambda^2+\ldots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\lambda^3+\ldots)\\ &\quad + \gamma^2\delta_{t+2}^V(\lambda^2+\lambda^3+\lambda^4+\ldots) + \ldots)\\ &= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \ldots\right)\\ &= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V\end{aligned}$$

➢ Use Q-function to estimate action values

16

4. LEARNING ALGORITHM

Q-function Approximation (Theory)

- Represent Q-function as a neural network Q_ϕ
- Action values recursive relation
  - SARSA(0)

$$Q^\pi(s,a) = r + \gamma Q^\pi(s',a')$$

- Minimizes Time-Difference Error

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\left(Q_\phi(s,a) - \left(r + \gamma Q_\phi(s',a')\right)\right)$$

17

4. LEARNING ALGORITHM

Q-function Approximation (Practice)

- Represent 2 Q-functions as neural networks:
  - Learning function Q_ϕ
  - Target function $Q_{\phi_{targ}}$
- Uses replay buffer to store and sample training data
  - (s,a,r,s') tuples
- Minimizes Mean Squared Bellman Error (MSBE):

$$\frac{1}{N}\sum_{i=0}^{N}\left(Q_\phi(s_i,a_i) - y(r_i,s_i',d_i)\right)^2$$

where $y(.)$ *is the regession target*

Deep Q-Network

Deep Deterministic Policy Gradient

Soft Actor-Critic

18

4. LEARNING ALGORITHM

Target Q-Values

$$y(r_i,s_i',d_i) = r_i + \gamma(1-d_i)Q_{\phi_{targ}}(s_i',\tilde{a}')$$

where:

d_i *indicates whether or not the state is terminal,*

$\tilde{a}'$ *is the action sampled from the lastest policy for state* s_i'.

- Uses target Q-function, $Q_{\phi_{targ}}$ to calculate regression targets
- Samples training data from replay buffer
- $\tilde{a}'$ ensures that Q_ϕ is on-policy
- Update $Q_{\phi_{targ}}$ by polyak averaging

$$\phi_{targ}' = (1-\rho)\phi + \rho\phi_{targ}$$

19

4. LEARNING ALGORITHM

Model-based Reinforcement Learning

- Definition:
  - Suppose there exists a function $\mathcal{T}$ such that: $\mathcal{T}(s,a) = (s')$
  - Approximate $\mathcal{T}$
- Motivations:
  - Physical interactions are expensive
  - Hard or unable to simulate environment
- Advantages:
  - Sample-efficient
  - Accelerate learning if model is accurate

20

4. LEARNING ALGORITHM

Learning Transition Dynamics

- Model $\mathcal{T}$ with a function approximator
- Collect state transitions (s, a, s') or sample from replay buffer
- Solve regression problem

$$\mathcal{L}_{dyn} = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{T}_\omega(s_i, a_i) - s_i')^2$$

where:
$\mathcal{T}_\omega(.)$ *is the dynamic model to be trained with parameters* ω,
N *is the number of sampled data from* $\mathcal{D}$.

4. LEARNING ALGORITHM

Overall Algorithm

Algorithm 1: Q-PPO

1: Input:
- Initial policy parameters θ_0
- Initial value function parameters η_0
- Initial Q-function parameters ϕ_0
- Initial dynamic model parameters ω_0

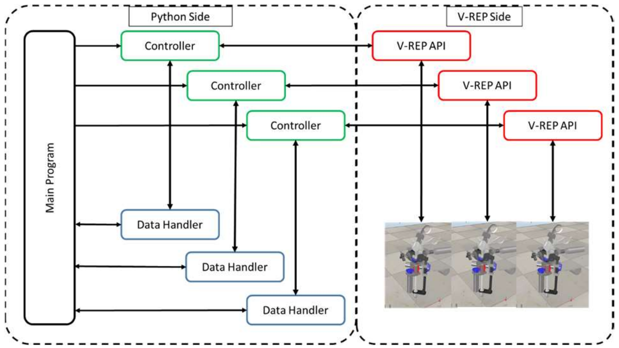
2: Set target Q-function parameters $\phi_{targ} \leftarrow \phi_0$
3: Create empty repay buffer $\mathcal{D}$.
4: **for** k = 1, 2, 3, … **do**
5: Collect set of trajectories $\{\tau_k\}$ by running policy π_{θ_k} in environment.
6: Generate imaginary rollouts using dynamic model $\mathcal{T}_{\omega_k}$ and add them to $\{\tau_k\}$.
7: Add all tuples $(s, a, r, s^{'})$ from $\{\tau_k\}$ to $\mathcal{D}$.
8: Compute $\hat{A}^{'}$ for all collected states s with Q_{ϕ_k} and V_{η_k}.
9: Update policy parameter θ_k via stochastic gradient ascent.
10: Update value function parameters η_k by minimizing the loss:

$$\mathcal{L}(\eta, \tau_k) = \frac{1}{N}\sum_{i=1}^{N}\Big(V_\eta(s_i) - \big(r_i + \gamma V_\eta(s_i^{'})\big)\Big)^2$$

11: Sample training set $\{\mathcal{R}_k\}$ from $\mathcal{D}$.
12: Update Q-function parameters ϕ_k using samples from $\{\mathcal{R}_k\}$.
13: Update ϕ_{targ} .
14: Update dynamic model parameters ω_k using samples from $\{\mathcal{R}_k\}$.
15: **end for**

4. LEARNING ALGORITHM

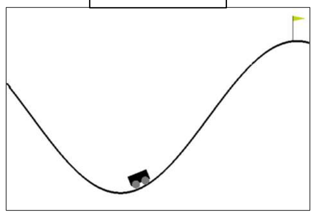
Implementation



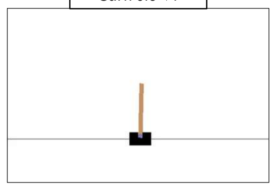
5. EXPERIMENTS AND RESULTS

Test Environments (OpenAI Gym)

MountainCar-v0



CartPole-v1



5. EXPERIMENTS AND RESULTS

Car Simulation

https://www.youtube.com/watch?v=dBp4lveVllk

25

5. EXPERIMENTS AND RESULTS

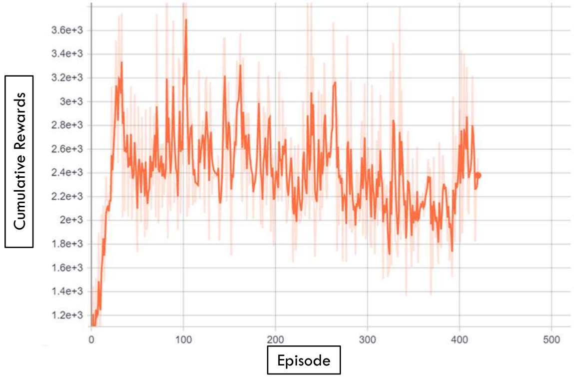
Laparoscopic Simulation

https://www.youtube.com/watch?v=FBDMQz5KcCM

26

5. EXPERIMENTS AND RESULTS

Cumulative Rewards



27

6. CONCLUSION & FUTURE WORK

- A modified PPO algorithm is proposed.
- The proposed algorithm was tested in multiple environments
- Demonstrated the potential for autonomous robotic control of the laparoscope

- ○ Extensive testing against other algorithms
- ○ Hyper-parameters tuning for better performance

28

Thank you for listening!

29

Q&A

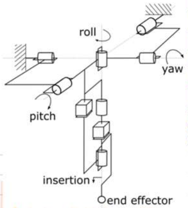
- Email: e0442113@u.nus.edu

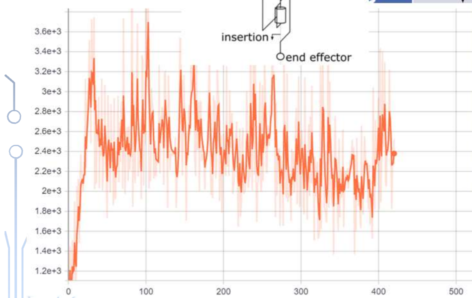
30

APPENDIX

Hyper-parameters	Value
Policy network	Dense(256) - Dense(256) - Dense(256) - Hyperbolic Tangent
Value network	Dense(256) - Dense(256) - Dense(256) - Linear
Q-network	Dense(128) - Dense(128) - Dense(128) - Linear
Dynamic model	Dense(128) - Dense(128) - Dense(128) - Linear
Policy learning rate	0.001 then decrease linearly with iterations
Value function learning rate	Same as policy network
Dynamic model learning rate	0.001
Q-function learning rate	0.01
Discount factor γ	0.99
GAE factor λ	0.95
Clip ratio	0.1 then decrease linearly with iterations
Polyak factor ρ	0.995
Number of trajectory per episode	5
Maximum interactions per trajectory	1400
Number imaginary trajectories	20
Maximum interactions per imaginary trajectory	8
Number of training iteration	1000
Policy training epochs per iteration	30
Value function training epochs per iteration	30
Dynamic model training epochs per iteration	15
Q-function training epochs per iteration	5

31





- Introduction
- Literature Review
- V-REP Environments
- earning Algorithm
- xperiments and Results
- onclusion and Future Works

32