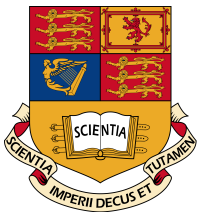
Imperial College London



Department of Bioengineering

Individual Project Report

Autonomous Agents for Virtual Surgery Simulators

Author:
Samradnyee Kolas

Supervisor:
Pr. Fernando Bello

Co-supervisor:
David Escobar-Castillejos, Ph.D.

Submitted in fulfilment of the requirements for the award of MSc in Biomedical Engineering from Imperial College London

September 11, 2020

Word Count: 5976

Contents

Abstract

The use of Virtual Reality Surgery Simulators to train surgeons on surgical skills has gained popularity in the last few years. Using Machine Learning algorithms alongside Virtual Reality Surgery Simulators has broadened the possibilities of classification and training. Current work investigates the use of Reinforcement Learning algorithms to train agents providing demonstration and feedback. This project investigated the success of Machine Learning agents trained on a Unity learning environment that simulated laparoscopic tasks. The agent was trained using Proximal Policy Optimization, Soft Actor-Critic, Generative Adversarial Imitation Learning and Behavioural Cloning, to autonomously perform localisation and peg transfer laparoscopic tasks. The agent successfully completes a basic localisation task using Reinforcement Learning algorithms, and can carry out a peg transfer task using Reinforcement Learning, with brain switching, and Imitation Learning, although not with the same precision or speed as an expert surgeon. Therefore, the potential of Reinforcement Learning and Imitation Learning has been shown in this project, but further work would be needed to optimise training.

Acknowledgements

I would like to begin by thanking my supervisor, Prof. Fernando Bello, for his continued support and advice throughout this project. I would also like to thank Dr. David Escobar-Castillejos, without whom this project would not have been successful. I am grateful for his constant willingness to assist me with any problems, as well as creating the laparoscopic environment used in my research. I would like to thank the SiMMS research team for always providing me with constructive feedback during our fortnightly meetings. I am especially thankful to my close friend and colleague, Solene Girardeau. With you, these last few months have been brighter and more bearable. Thank you for all your support, and I will miss you. Thank you to all my friends, because of whom I have learned distance is meaningless in the face of strong bonds. Finally, I would like to express my deepest gratitude to my parents and my brother, for encouraging me every step of the way. I am who I am today because of your love and support.

Samradnyee Kolas
September 11, 2020

Glossary

- API - Application Programming Interface
- BC - Behavioural Cloning
- GAIL - Generative Adversarial Imitation Learning
- IL - Imitation Learning
- ML - Machine Learning
- MIS - Minimally Invasive Surgery
- NN - Neural Network
- PPO - Proximal Policy Optimization
- RL - Reinforcement Learning
- SAC - Soft Actor Critic
- SiMMS - Simulation and Modelling in Medicine and Surgery
- VRSS - Virtual Reality Surgery Simulator
- TRPO - Trust Region Policy OptimiZation

1 Introduction

1.1 Minimally Invasive Surgery

Minimally Invasive Surgery is a surgical procedure that has been used as an alternative to traditional procedures for lung, heart, gynaecologic, general surgery etc. since the 1980s, as a result of advancements in surgical technology [1][2]. MIS provides patients with more benefits than conventional surgery, such as reduced pain and bleeding, reduced scarring, and faster return to normal functions post-operation [3][4]. However, compared to conventional open surgery where the procedure is conducted via one large incision, MIS uses numerous small incisions in the abdomen to access the surgical area [Fig. 1] [5]. This poses several challenges for surgeons, such as reducing the field of view, difficulty maneuvering tools, less depth perception and reduced haptic feedback. These challenges increase the difficulty of the operation, resulting in a longer learning curve for trainees [6]. Hence, an appropriate training method must be used to train surgeons quickly and accurately.

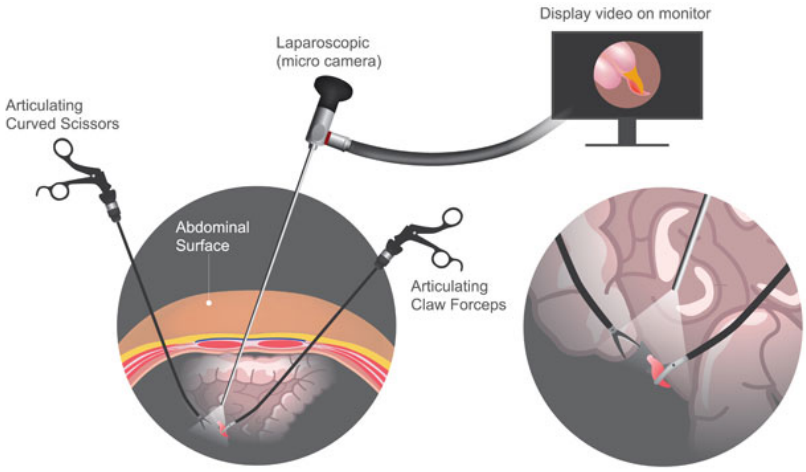


Figure 1: Laparoscopic surgery [7]

1.2 Virtual Reality Surgery Simulators

Simulation training for resident surgeons has become increasingly popular. Common simulators are benchtop models, such as box trainers, which are equipped with laparoscopic tools. However, current assessment techniques involve subjective evaluations by expert surgeons. Furthermore, changes to trainee working hours have led to a drastic decrease in "hands-on time" available for training [8][9].

Virtual Reality Surgery Simulators (VRSS) provide trainees the opportunity to perform tasks numerous times in a safe, repeatable environment until they have reached a sufficient level of competency [10][11]. VRSS provide a unique, comprehensive learning experience for trainees based on their individual advancement. VRSS provide

training for fundamental technical skills, such as hand-eye co-ordination, fine motor skills and spatial skills, as well as complication and scenario training for rare procedures/diseases [10]. Surgeons trained using VRSS have been shown to perform laparoscopic tasks better than surgeons that had not. Seymour et. al (2002) found that VR-trained trainees completed gallbladder dissections 29% faster than their non-VR trained counterparts [12]. Progress reports and real-time feedback can be used to objectively monitor trainee performance, and proficiency can be determined using a training curriculum with tasks of increasing difficulty [10][13].

Although VRSS have the potential to provide data-based assessments, most assessments are still based on checklists and rating scales monitored by experts, which can be influenced by human bias. The immense volume of data collected by VRSS must be processed suitably to discover patterns in the data that reveal information about the skills of the trainee [14]. Machine Learning (ML) methods lend themselves well to data processing for this purpose, especially deep learning algorithms that aim to discover patterns in data during training [14].

1.3 Machine Learning

Machine Learning (ML) is an overarching term that encompasses the use of artificial intelligence to learn patterns from data. ML can be divided into different types types of algorithms: supervised learning, unsupervised learning and reinforcement learning. Supervised and unsupervised learning aim to label input data based on features [15]. Previous research has explored the use of different ML algorithms to classify participants into discrete skill levels. Analysing raw data collected by the simulator using either supervised or unsupervised learning has been proven to identify hidden patterns in technical skills, while classifying expertise with greater precision [16][17].

On the other hand, reinforcement learning (RL) trains an agent to maximise a reward through its experiences in an environment [Fig. 2] [15]. RL uses a trial-and-error learning process, making it suitable to train an agent that mimics human behaviour [18].

There are five key components in RL:

- the **environment** in which the agent is being trained
- the **observations** the agent learns from the environment i.e. the sensory inputs it can measure
- the **actions** the agent can take in its current configuration
- the **policy** that links an observation to the actions it can take
- the **reward signal** the agent receives for each action it takes

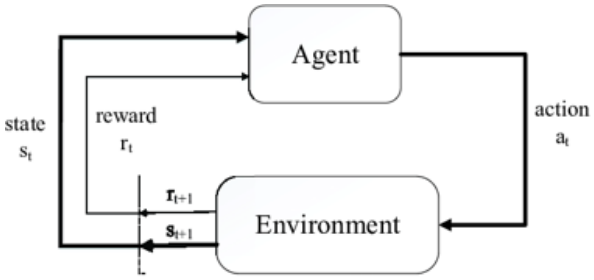


Figure 2: Reinforcement learning [24]

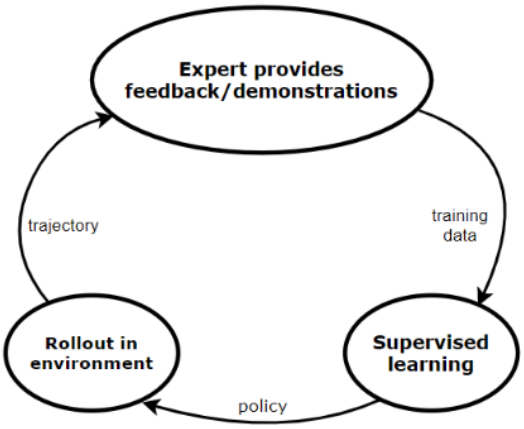


Figure 3: Imitation learning [22]

RL aims to maximise this reward signal by learning a policy that takes the action with the greates reward at each step [15]. Recent studies aim to use RL trained autonomous agents for demonstrations and feedback during training, rather than just classification. The use of autonomous agents for grading and feedback has already shown potential in medical training for traumatic emergencies [19].

Alternatively, agents can be trained using human demonstrations through imitation learning (IL). By modelling expert behaviour as Markov decision processes [20], supervised learning can be used to learn the relationship between the state of the agent and the optimal action it should take [Fig. 3] [21]. In particular, IL works effectively when demonstrating a policy for which defining an RL reward signal would be difficult [22]. Chui et. al (2011) investigated the use of an IL trained robot trainer for feedback and demonstration during surgical training. They suggested that the ability of the robot trainer to record performance could allow for comparison between experts and trainees [23].

Chui also suggested a combination of IL and RL could increase the robustness of the motion tracking performance [23]. Tan et. al (2019) investigated the use of ML agents trained using RL algorithms and human demonstration during training. The agent first learns an RL policy; this learned policy and expert demonstrations are used together to generate an IL policy. Using the IL policy, this agent can be used to demonstrate tasks and provide feedback to trainees [25].

1.4 Motivations and Aims

The Simulation and Modelling in Medicine and Surgery (SiMMS) research team has previously investigated the use of ML agents to train VRSS tools to autonomously complete laparoscopic tasks. Lerede (2019) investigated the suitability of the Unity framework as a learning environment for autonomous ML agents. Agents were trained to execute a simple localisation task and a complex needle handling task using an RL

algorithm. Although the framework showed potential, further work suggested exploring other algorithms for training [26]. Chen (2019) investigated the use of IL to train agents on a simple localisation task, as well as providing feedback to trainees. Demonstrations were first recorded using a laparoscopic device, after which an offline IL algorithm successfully used these demonstrations to train the agent to locate randomly positioned goals [27]. Both Lerede and Chen mention Generative Adversarial Imitation Learning, a new IL algorithm available with the latest release of Unity ML Agents Toolkit, as a starting point for further work.

Building upon Lerede and Chen's work within the SiMMS group, and considering studies discussed in Section 1.3, the aim of this project will be to train an agent to learn an RL policy and an IL policy for the same laparoscopic task. Both RL and IL policies will be compared using various metrics, to determine which worked better overall.

2 Methods

2.1 Unity and ML Agents

Unity is a 3D games development platform that can be used to make various interactive simulations. Unity combines sensory, physical, task logic and social complexity, which can be used to train ML Agents. Using the four key components of the Unity Engine, GameObjects, Scenes, Assets and the Project, high-fidelity surgical simulations can be recreated [28]. The ML Agents Toolkit provided by Unity enables ML research in simulated environments, by using the Unity Editor linked to a Python Application Programming Interface (API). Within a Unity learning environment, each GameObject that interacts with the environment is assigned an Agent component; all Agents receive three types of information every frame [29]:

- **Observations** - numerical and visual perceptions of the environment from the Agent's point of view
- **Actions** – the possible actions the Agent can take from that state
- **Reward Signals** – rewards given to the Agent when it takes an action that can be positive or negative depending on how desirable the action is

Using this information, each Agent can learn a policy, the optimal action to take from each possible state to maximise the episode reward. This policy is learned by playing several episodes, during which the Agent explores the environment and learns what rewards it can receive for different actions. This policy is called the Behaviour, or *brain*. Each Agent component has an associated Behaviour, which may be common between Agents. The Behaviour receives observations and rewards which determine the Agent's next action. There are three Behaviour types: Learning, one that is in the process of training, Inference, based on a Neural Network (NN) file, or Heuristic,

defined by hard-coded instructions. Depending on the type of Behaviour attached, the Agent changes how its actions are decided [29].

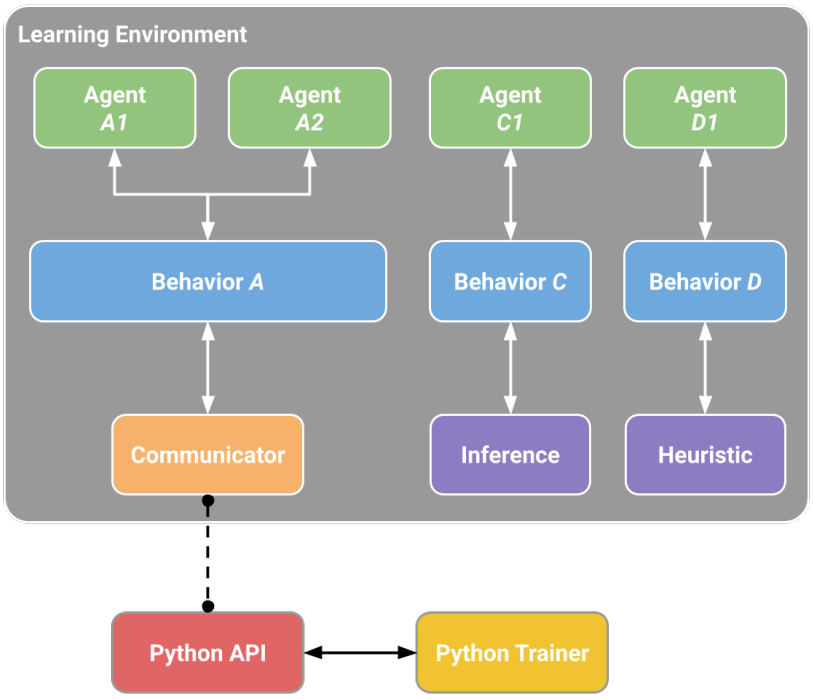


Figure 4: ML Agents hierarchy [29]

The environment can be used in either training or inference phase. During training, the Python API linked to the learning environment employs various RL and IL algorithms to train the Agent. All observations are processed by the API, which returns the actions the Agent should take. At the end of training, this learned policy is exported as an NN model. During inference, the Agent's observations are processed by the NN model, which returns the optimal action for the Agent to take [29]. In this project, the following four algorithms will be used to train the Agent:

- Proximal Policy Optimization (PPO): a type of deep RL algorithm which uses policy gradient methods. PPO is an on-policy algorithm, which aims to improve the current action selection policy [30]. PPO uses gradient descent on a cost function to search for the most optimal policy. It is comparatively stable, and is the most common RL algorithm [29][31].

- Soft Actor Critic (SAC): an off-policy deep RL algorithm which allows the agent to learn from past experiences, unlike PPO. Off-policy algorithms aim to improve a different policy than that currently being used for action selection [30]. SAC has many advantages that make it desirable when learning from real-world simulations, the most important ones being its sample rate and the hyperparameter tuning. All collected experiences are randomly replayed during training, drastically minimising the number of samples it requires to learn. As a maximum entropy algorithm, the requirement for hyperparameter tuning is reduced [29][32].

- Generative Adversarial IL (GAIL): an IL algorithm which learns from demonstrations of expert behaviour. This method is based on the idea of adversarial learning, where the Agent must learn to behave like a set of expert demonstrations; a separate discriminative model acts as the adversary, trying to distinguish between the learned distribution and demonstration distribution [Fig. 5]. The more the discriminator believes the Agent behaves like the demonstrations, the higher the reward is [29][33].

- Behavioural Cloning (BC): an IL algorithm that trains the agent to directly imitate the actions shown in a demonstration. It works best with GAIL, as it cannot generalise the agent's actions outside the demonstrations by itself and requires an exploration element that is rewarded [29].

A C# script attached to the Agent will be used alongside the API during training. The script will define the observations to be collected from the environment and the reward signals that can be received,

2.2 Laparoscopic Environment

The laparoscopic environment created by Dr. Escobar-Castillejos in Unity simulates a peg transfer task. This is a common task used during laparoscopic training to test ambidexterity, hand-eye co-ordination and fine motor skills [35]. The learning

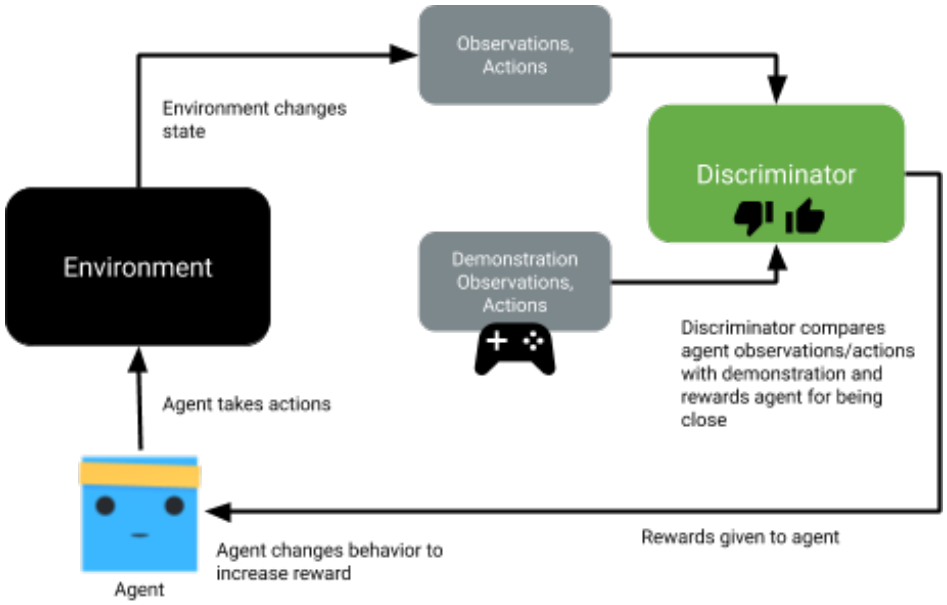


Figure 5: GAIL algorithm in Unity [34]

environment consists of an Agent, the laparoscopic tool, and the GameObjects, a ring and the two stacks between which the ring must be transferred. Henceforth, the ring will be referred to as the *goal*, the stack on which the ring initially rests as the *initial stack* and the stack on which the ring is to be transferred as the *target stack*. The goal is modelled as a ring of 10 sphere colliders, and the stacks are modelled as capsule colliders. The laparoscopic tool acts as an Agent to interact with the environment and learn a policy that maximises the reward it receives. The tool consists of a fulcrum, scope, tip and tail with four degrees of freedom: rotation in x, y, and z axes, and translation in the z-axis representing tool insertion. The GameObjects rest on a flat plane, mimicking the set up in a box trainer. The laparoscopic environment and laparoscopic tool are shown in Fig. 6 and 7, respectively.

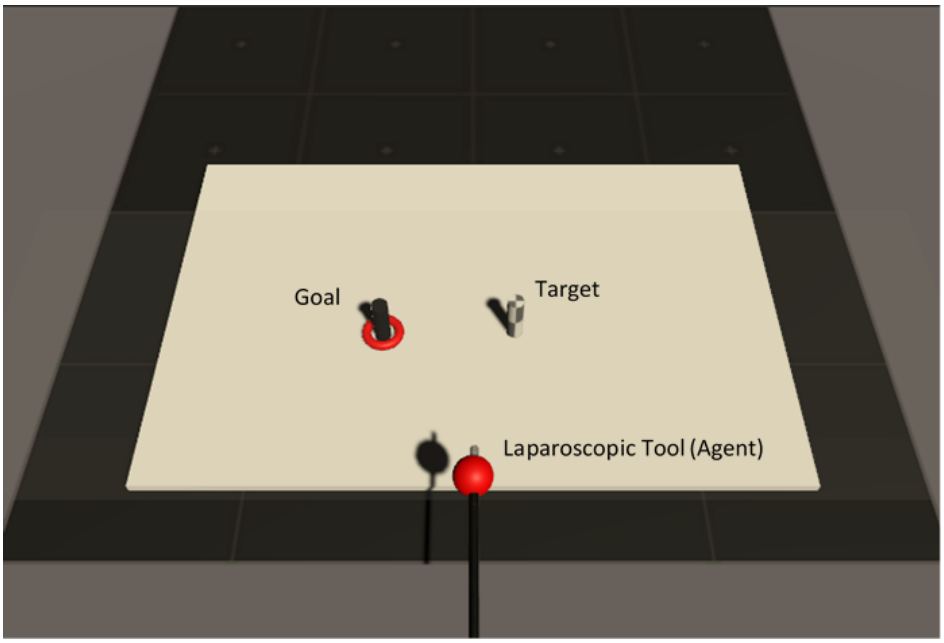


Figure 6: Laparoscopic environment

2.3 Tasks

Throughout the project, the Agent was trained on increasingly complex tasks. The specifics of the tasks are detailed below.

- Localisation: The Agent was first trained to localise the goal. It was trained to locate the goal and stay there for 50 frames, before the area was reset.
- Peg Transfer – One Brain: The Agent was then trained to complete the whole peg transfer task using one brain. This involved locating the goal on the initial stack, picking it up, and dropping it on the target stack.
- Peg Transfer – Brain Switching: The Agent was then trained to complete the peg transfer task while switching brains. The aim was to train one brain to locate the goal and another to locate the target stack. An Agent script would code for the agent to use the goal brain to locate the goal, switch to the target brain once it has grasped the goal, locate the target stack and drop the goal.
- Peg Transfer - IL: The simulation was heuristically completed 100 times by a user, to amass a set of demonstrations. The demonstrations were then used to teach the Agent how to transfer a goal from one stack to another, using GAIL and BC reward signals.

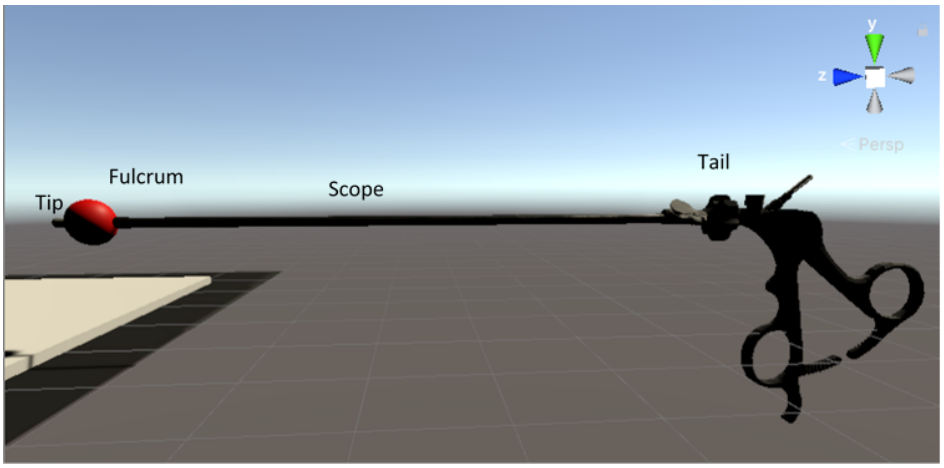


Figure 7: Laparoscopic tool

Distance Decrease	Distance Increase	Temporal Reward	Bounds Reward
+0.5	-0.2	-0.01	-3.0

Table 1: Rewards signal

2.4 Rewards Signal

To successfully train an agent, an appropriate rewards signal must be used. Success in both localisation and peg transfer tasks rely on reducing the distance between the tool tip and the goal, and the tool tip and the target. Agnew and Domingos investigated the use of human prior knowledge when modelling probability of object collision. One prior was that objects will only collide if the distance between the objects decreases [36]. Using this intuitive idea, a reward signal based on the change in relative distance between the Agent and the goal or target was determined. It positively rewarded a decrease in distance, and negatively rewarded an increase in distance. The Agent also received a negative temporal reward, encouraging the it to learn the quickest policy. To encourage the Agent to stay within the training environment, it also received a negative reward if it goes out of bounds. The proposed reward signal is shown in Table 2.4.

3 Results

3.1 Evaluation Metrics

Each task listed in Section 2.3 was trained using either PPO, SAC or IL (GAIL and BC). For each task, the aim was to measure how successfully the Agent completes the task using the learned policy. This was done using training and inference metrics.

During training, the ML Agent saves various statistics associated with the training session; these can be visualised using Tensorboard. The main training session metrics that were used for comparison between policies are [37]:

- **Cumulative Reward** – the accumulated reward of the Agent at a certain frame in the episode, which should increase over the training session
- **Episode Length** – the mean length of each episode in the environment
- **Entropy** – the randomness of the decisions taken by the model, which should decrease throughout training
- **Learning Rate** – the magnitude of the step an algorithm takes, which should decrease during training
- **Value Loss** – how well the model can predict each state value, which should initially increase but should decrease over time

Particularly, the entropy of training and cumulative reward were used for comparison. If possible actions of the Agent are modelled as a probability distribution, in deep RL, entropy is the randomness of those actions. As RL encourages exploration of the environment, the actions will be more random near the start of training. As an optimal policy is learned, the actions become less random. Decreasing entropy values over a training session indicate successful training [38]. As the Agent learns a policy that maximises the reward, the cumulative reward should increase over a training session. An increasing cumulative reward theoretically indicates a successful training session.

The evaluation metrics used during the inference phase were:

- **Average time** taken for the agent to complete an episode
- **Visual of the environment** showing successful transfer of the goal

It takes a expert surgeon an average of 48 seconds to successfully transfer six pegs between stacks [35]. Hence, the faster the Agent is able to successfully complete a task, the better the policy is assumed to be. Using visual feedback, if the Agent has managed to drop the goal on the target, the policy can be considered a success.

Unless explicitly stated, all agents were trained using ML Agents Ver. 0.14.0, the version for which the learning environment was designed. For each method listed in Section 2.3, both training and inference metrics were used to measure success of the learned policy. All graphs presented in Section results and Appendix 6.2 represent the statistics of the training session, visualised using Tensorflow. For each graph, there is a bold line and a faint line; the faint line represents the raw data and the bold line represents the smoothed data. Appendix 6.1 briefly describes the training parameters used.

3.2 Localisation

The Agent was trained using both ML Agents Ver. 0.14.0 and 0.19.0 (Release 6); henceforth, Ver. 0.14.0 will be referred to as Trial 1 and Ver 0.19.0 as Trial 2. The main difference between training was the reward signal given to the Agent. Both training sessions positively rewarded a decrease in relative distance between the tip and the goal; however, there was a slight difference between the reward signals. This difference is described in Figure 8.

ML Agents Version	Positive Reward	Negative Reward	Temporal Reward
0.14.0	+0.4f if magnitude of relative x and z positions decreased from previous step	-0.5f otherwise	-0.01f
0.19.0	+0.04f if magnitude of relative positions decreased from previous step	-0.05f otherwise	-0.01f

Figure 8: Localisation rewards during training

Whilst observing the tool locating the goal using the trained NN model, both models can successfully locate the ring repeatedly. The tool navigates to the goal, maintains its position for 50 simulation frames and then resets. However, the Trial 1 NN model cannot locate the goal as often as the Trial 2 model; the Trial 2 NN model also completes the task faster. Comparing both graphs for entropy, the Trial 2 model seemed to converge on a better optimal policy, as the entropy decrease is greater than that of the Trial 1 model, as seen in Figures 9 and 10.

Comparing cumulative reward, the training session for the Trial 2 NN model increased and then plateaued, suggesting a successful training session. On the other hand, the cumulative reward Trial 1 NN model increased, but did seem to plateau or reach the same value as Trial 2. Overall, the Trial 2 NN Model seemed to have a more successful training session.

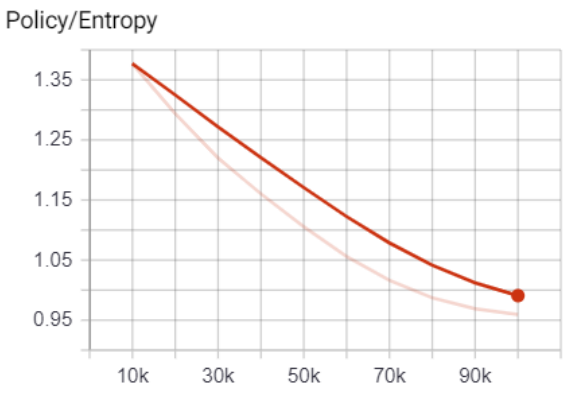


Figure 9: Trial 1

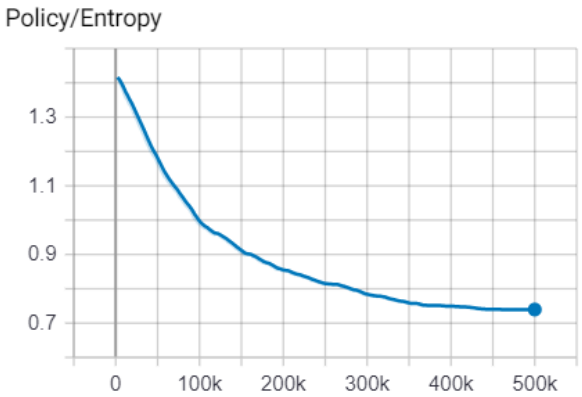


Figure 10: Trial 2

Entropy across training

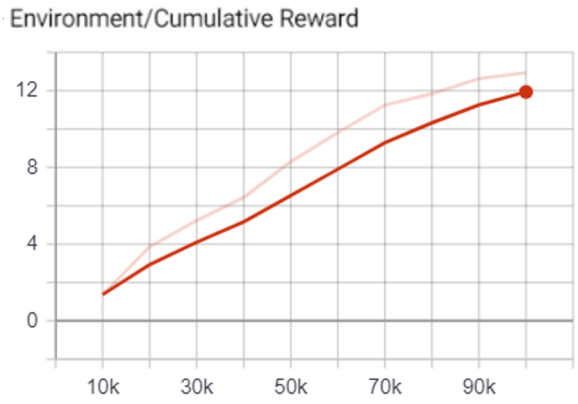


Figure 11: Trial 1

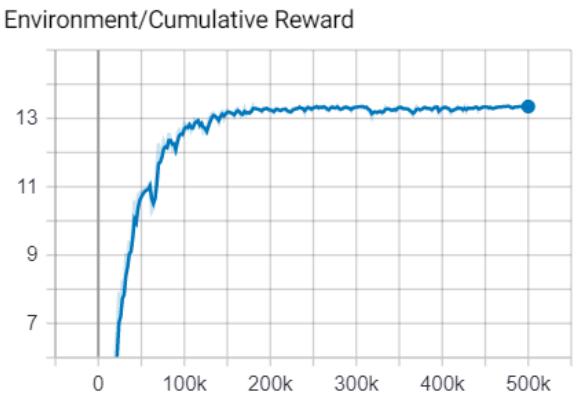


Figure 12: Trial 2

Cumulative reward across training

3.3 Peg Transfer - One Brain with No Constraints

The next task was to train one brain to pick up the goal and drop it onto the target. Using the localisation idea from the first task, the Agent had to locate the goal, grasp it using the Transform.parent function in Unity [39], and then locate the target and drop the goal, once it was within a certain distance. Like the localisation task, the Agent would be reward by decreasing the distance between the Agent and goal before lifting the goal, and decreasing the distance between the Agent and the target when the goal was grasped. Looking at the entropy and cumulative reward graphs (Figures 31 and 32), the Agent seems to have experienced a successful training session, since entropy decreases and cumulative reward increases. However, when the Agent is observed in inference mode, it fails to drop the goal on the target. Although it seems to drop it occasionally, this would suggest a random policy rather than a learned policy.

3.4 Peg Transfer - One Brain with Constraints

As training with one brain failed, it was decided that some constraints should be placed on the tool, to make the rewards system more specific to the current movement and to encourage the tool to act like a surgeon i.e. raise the goal off the initial stack, and then transfer it to the target stack. To do so, the task was split into six sub-sections. The first three sub-sections aim to locate the goal, by first moving the tool tip to a within a certain threshold of the z, then y, then x position of the goal. Similarly, when the tool grasps the goal, it does the same for the target stack, by first lifting the goal off the stack to a certain y height, then moving it in the z and x directions to within a certain threshold of the target, where it is then dropped. This is illustrated in Figure 13. Using these constraints, the agent was trained using both PPO and SAC, as SAC was thought to be more suitable for real-life situations.

As can be seen in Figures 15 and 14, SAC appears to have had more success

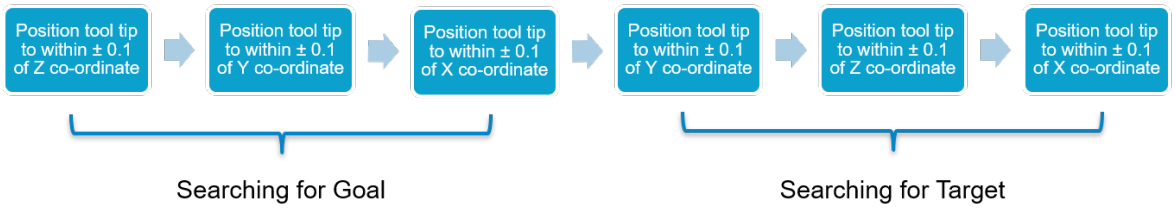


Figure 13: Constraints flowchart

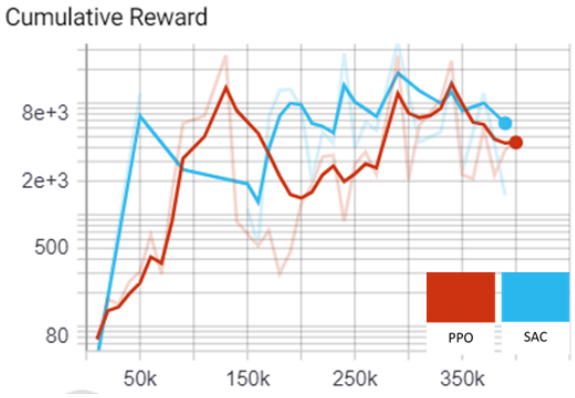


Figure 14: Cumulative Reward

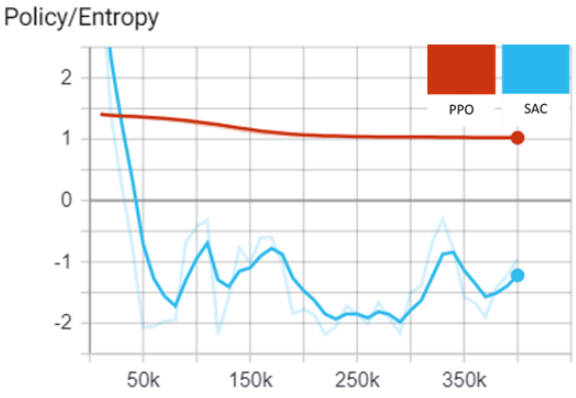


Figure 15: Entropy

PPO vs SAC training statistics

training the Agent than PPO, as SAC has a higher cumulative reward and lower entropy. However, observing the agent in inference mode shows that constraining the motion made no difference in training the Agent. Like the previous task, both NN Models trained using SAC and PPO fail often. Training one brain to complete the whole task seemed to fail regardless of whether or not constraints were put on the Agent, so concept of multi-brain switching was applied to the task.

3.5 Peg Transfer - Multi-Brain Switching

As mentioned in Section 2.3, the peg transfer task was to be split into two brains: one for goal localisation, the other for target localisation. Instead, following the singular brain method with constraints, seven brains were individually trained to position the tool tip within a threshold of the x,y,z co-ordinates of the goal or target, shown in Figure 16. Although six brains were to be trained initially, two brains were trained to localise the tool tip to the target stack's y co-ordinate, one to move the tip upwards and one to move the tip downwards. The brains were switched based on hard-coded constraints [Figure 17].

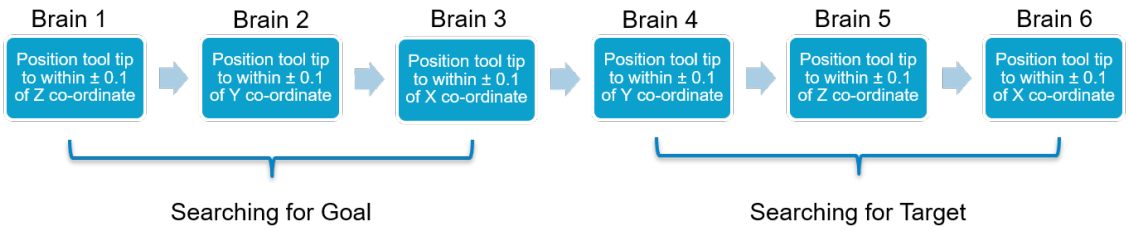


Figure 16: Multiple brains flowchart

Brain Name	Is grasping?	Conditions for Switching - *Distance (X, Y, Z)*	Action
Brain 1	X	For RGD: If |Z co-ordinate | > 0.2	Limit movement to insertion
Brain 2	X	For RGD: If |Z co-ordinate| < 0.2 & |Y co-ordinate| > 0.2	Limit movement to rotation around Y-axis
Brain 3	X	For RGD: If |Z co-ordinate| < 0.2 & |Y co-ordinate |< 0.2 & |X co-ordinate| > 0.1	Limit movement to rotation around X-axis
Brain 4	✓	For RTD: If Y co-ordinate > -0.25	Limit movement to rotation around Y-axis
Brain 5	✓	For RTD: If Y co-ordinate < -0.35	Limit movement to rotation around Y-axis
Brain 6	✓	For RTD: If Y co-ordinate > -0.35 & Y co-ordinate < -0.25 & |Z co-ordinate|> 0.075	Limit movement to insertion -axis
Brain 7	✓	For RTD: If Y co-ordinate > -0.35 & Y co-ordinate < -0.25 & |Z co-ordinate| < 0.075 & |X co-ordinate| > 0.075	Limit movement to rotation around X

Figure 17: Switching conditions, where RGD is Relative Goal Distance, distance between goal and tip, and RTD is Relative Target Distance, distance between target and tip

The training graphs for all brains followed the trends expected of a successful training session. As seen in Figures 19 and 20, cumulative reward increases and entropy decreases across the training session for all brains. When observed in inference mode, the Agent can locate the goal and drop it on the target with little error.

Although the brains were trained using one goal and target, the brains work just as well on multiple goals and targets in different locations. To simulate a real-life peg transfer training simulation, numerous goals and target stacks were set up in the environment (Figure 18. Using Fisher-Yates shuffling [40], a random order was generated for the goals to be picked up and dropped onto their corresponding targets. This aims to simulate a physical training simulation. Surgeons are often asked to pick up rings with one hand and drop them on the stacks on the opposite side of the board, and then pick them up with the other hand and drop them on the original stacks, to show their ambidexterity and fine motor skills [35].

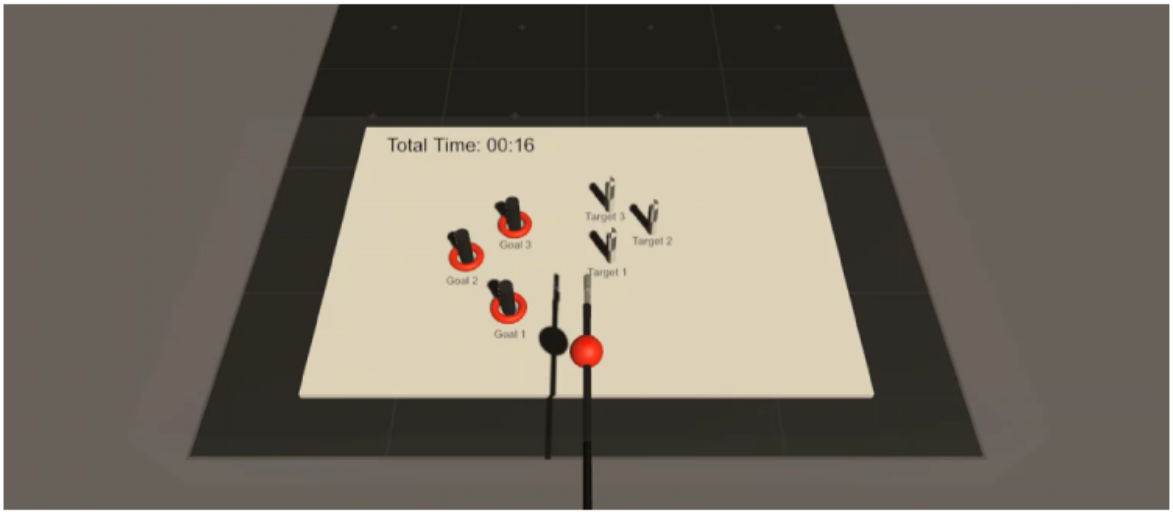


Figure 18: Multiple goals environment

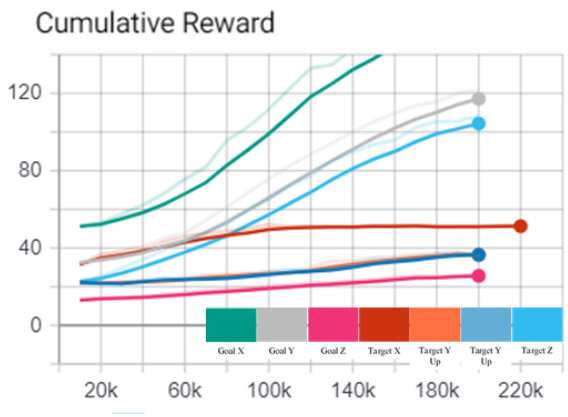


Figure 19: Cumulative Reward

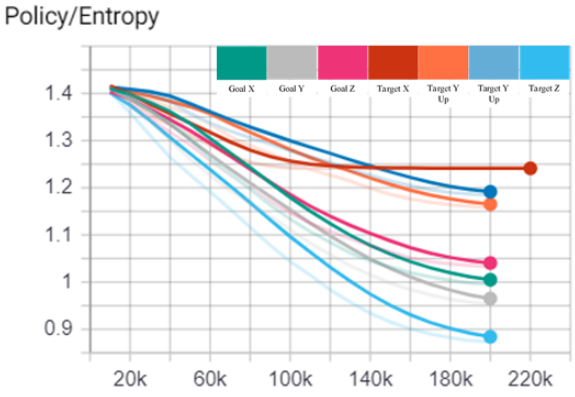


Figure 20: Entropy

Multi-brain training statistics

3.6 Peg Transfer - IL

Finally, using the heuristically recorded demonstrations, the agent was trained using GAIL and BC algorithms in ML Agents Ver. 0.19.0. Ver. 0.14.0 was incompatible with GAIL training, due to errors in the Toolkit; the project was then adapted to work with Ver. 0.19.0, the most recent release available. Unlike previous tasks using RL algorithms, this task did not take into account exploration reward signals from the Agent script. Instead, the configuration files provides parameters for the GAIL and BC reward signals. GAIL provides positive rewards if the Agent has similar observations and actions to the demonstrations. Another algorithm, known as the discriminator, classifies whether the observation-action pair comes from the Agent or the demonstration. If the Agent is able to convince the discriminator its observation-action pair came from a demonstration, it receives a positive reward. As the Agent gets better at behaving like the demonstrations, the discriminator gets better at differentiating between the Agent and demonstrations, increasing the difficulty over time [34]. Using this reward signal, the Agent was trained to complete the task.

As shown in Figures 21 and 22, the graphs for entropy and cumulative reward represent a successful training session. In inference mode, once the tool has grasped the goal, the Agent seems to search the environment before localising the tip within the threshold of the target's x-position. Although statistically, this method seems to have trained, the reality is slightly different.

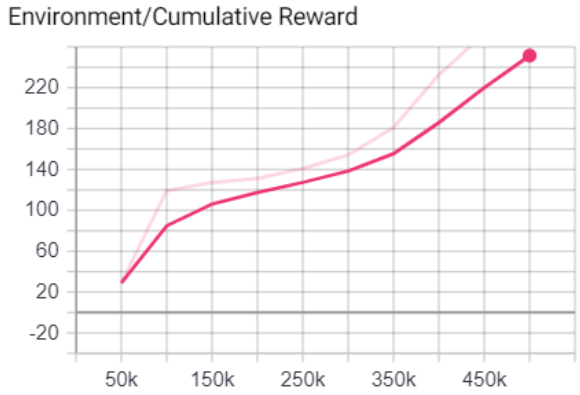


Figure 21: Cumulative Reward

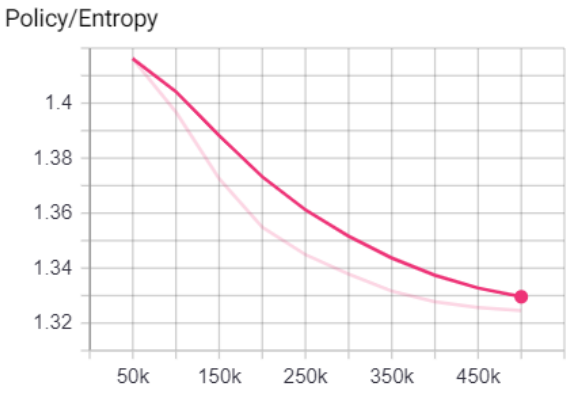


Figure 22: Entropy

IL training statistics

4 Discussion

4.1 Success of Experiments

The results in the Section 3 show that, given the right reward signals and training method, Unity ML Agents can be used to train autonomous agents to execute laparoscopic tasks. Observing the Agent in inference mode shows that it can successfully carry out the localisation and peg transfer tasks it was trained on, despite needing to divide complex tasks into several sub-tasks.

The Agent was initially trained on a simple localisation task. Observing the Agent using both Trial 1 and Trial 2 NN models to carry out the task shows that the reward signal is vital when finding an optimal policy. Although both are able to locate the target, the Trial 2 model converged on a better optimal policy than the Trial 1 model. As mentioned in 3.2, the cumulative reward and entropy decrease are greater. It is also able to generalise the policy, as the Agent can locate the goal, even when there are slight variations from the initial location.

As this method proved successful, the Agent was then trained to localise both the goal and the target using one brain. Although the training graphs show an increase in cumulative reward and decrease in entropy, this could be a result of the agent converging on a sub-optimal policy due to an incorrectly defined reward signal. As mentioned in Section 2.3, the Agent receives a negative reward if it goes out of bounds. Observing the Agent in inference mode shows the Agent repeatedly going out of bounds. This could suggest that the only consistent reward the Agent was able to receive was this negative reward. Therefore, it can be inferred that the Agent's optimal policy was to go out of bounds. Furthermore, it did not prove successful even after placing constraints on the motion. Both PPO and SAC trained NN models observed in inference mode get lost exploring the environment. Although SAC seemed to converge on an optimal policy when considering the training graphs, it seems to keep receiving exploration

rewards, hence the increase in cumulative reward.

The multi-brain switching policy and IL policy were the most successful when completing the peg transfer task. However, looking at the entropy graphs in Figures 20 and 22, the entropy decrease for the RL trained brains was greater than that of the IL trained brain. This suggests that the RL brains converged on an optimal policy during training better than the IL brain. Although it seems as though IL was more successful when comparing the cumulative reward, this can be attributed to extended periods exploration in the environment, during which the tool still receives exploration rewards. Each RL brain was trained to localise the tool tip to a certain position; as each brain managed to learn a straight line trajectory to complete the task, they received much fewer exploration rewards than the IL brain. Hence, even though the RL brains had smaller overall cumulative rewards, they worked more efficiently than the IL brain.

4.2 Limitations

Overall, the Agent can successfully carry out a simple localisation task and a complex peg transfer task using RL trained and IL trained policies, albeit slower than an expert surgeon (shown in Figure 23) [35].

User	Average Time (s)
Expert surgeon	8
RL trained Agent	17
IL trained Agent	38

Figure 23: Average time taken to transfer one peg between stacks

Even though the Agent was only trained on a stationary goal and target, it can generalise its learned policy such that it is able to carry out a peg transfer task in various locations. However, it is limited such that, if the goal is on a different side of the board than it was initially trained on, the Agent will not be able to locate it. This shortcoming prevents the task from being an exact simulation of a physical peg transfer task, as surgeons are often asked to use both hands and transfer rings between pegs on both left and right sides of the board [35]. Therefore, some changes would need to be made to the training environment in order to make the policy more robust to changes. One example would be to use curriculum training to train the Agent on increasingly difficult tasks. Curriculum training introduces tasks gradually throughout training, such that the Agent can successfully finish one task before the difficulty is increased in the next task [29]. By using two rings, one on each side of the board, and increasing their distance from the start position of the tool tip periodically, the tool can learn how to locate rings anywhere on the board.

Additionally, IL, using GAIL and BC, has been shown to work well in training, despite the resultant policy taking longer than the multi-brain switching policy. This

shows that IL has potential to train agents with the same accuracy as expert surgeons. However, due to the time constraints of the project, it was not possible to experiment with changes in training parameters i.e. the strength of BC and GAIL. To create a robust policy learned through IL, the simulation could be trained again using adjusted parameters. Furthermore, GAIL and BC could be combined with RL algorithms and extrinsic rewards to encourage exploration of the environment. Using RL algorithms can decrease the training time, and can encourage the Agent to follow the demonstration path to receive large environmental rewards [29] [34].

Unfortunately, as a result of COVID-19, it was not possible to record accurate demonstrations from expert surgeons using LapSim VR tools available in the SiMMS lab [41]. As the demonstrations were recorded on a laptop using the coded heuristic controls instead, the demonstrations may not have represented typical human behaviour when performing a peg transfer task. Since the training is dependent on the quality of demonstrations, human error could have influenced the learned policy, especially since the Agent did not use other RL algorithms to explore the environment. To overcome this problem and increase the quality of demonstrations, clinician data from different experts could be collected for training.

Additionally, the laparoscopic tool was not taught how to grasp the goal. In benchtop simulations, surgeons would have to learn how to pick up the ring with the tool tip; in the Unity simulation, the tool tip simply attached itself to the goal when they collided with each other. In future simulations, the tool could be taught how to grasp the goal on collision, receiving rewards if it can successfully grab the tool and hold on until it reaches its target. This could be another task the trainees would need to master during training.

4.3 Future Work

As mentioned in Section 4.2, there are various limitations that could be used as starting points for future work. Collecting accurate demonstrations from expert surgeons could improve quality of training using IL. Training a tool how to accurately grasp a goal would improve the fidelity of the simulation and would provide another assessment for the skill of trainees.

As mentioned in Section 4, another possible way of making the resultant NN model more robust is to use curriculum learning during training. By steadily increasing the difficulty of the task, the algorithm ensures the Agent can successfully complete simple tasks before moving onto more difficult tasks. This will make sure the Agent learns an optimal policy for each task, rather than getting stuck exploring the environment. Using curriculum learning with two goals and two targets placed on either side of the board, the environment could also be extended to simulate peg transfer between both sides of the board, rather than just one side to another like in the current environment. This will also increase the realism of the simulated peg transfer task.

PPO, SAC, GAIL and BC were all used during training in this project; however, due to the limited time frame, further changes could not be made to any of the

algorithms. Recent research into adapted RL algorithms have shown potential for better convergence on optimal policies. For example, a study by Nguyen et. al (2019) found that using an adapted PPO algorithm, called Trust Region Policy Optimization (TRPO) [42], to learn a tensioning policy to cut surgical tissue proved superior to other methods [43]. Incorporating RL algorithms such as SAC or PPO with IL could lead to a generalised optimal policy for various experiences in the environment, not just those demonstrated. Tan (2019) used PPO and GAIL to learn objective constrained behaviour and discover latent patterns from human tool manipulation, both of which are combined to create a well-rounded agent that experiences various trajectories in the environment and accurately provides feedback to trainees [25]. Using various adaptions of existing algorithms that could not be explored within this time frame could improve the quality of training.

The Agent in this project was only trained on localisation and peg transfer tasks. However, there are a vast number of laparoscopic tasks that could also be simulated using Unity, that the Agent could be trained on. For example, other tasks such as precision cutting, piercing, suturing and knotting are other fundamental tasks with which laparoscopic trainees must be familiar [35]. Using Unity, these tasks could be simulated similarly, and the laparoscopic tool could be trained to autonomously complete these tasks.

Furthermore, the learning environment currently simulates a benchtop trainer. The stacks and goals are resting on a flat plane that does not simulate a real surgical environment. The next task would be to create a deformable tissue environment, equipped with a needle that can be used for suturing and piercing the deformable tissue. The laparoscopic tool would be used initially to locate the needle, grasp and rotate it until it is able to pierce the tissue. After it has learned that, the tool can be trained to complete complex tasks such as knotting and suturing.

5 Conclusion

This project has attempted to prove the potential of RL and IL algorithms to train a laparoscopic tool to act as an autonomous agent to carry out localisation and peg transfer tasks. My contribution was exploring how different reward signals affect training, investigating how multi-brain switching is possible using RL trained policies, and to investigate the use of the GAIL algorithm to learn a policy in ML Agents Ver. 0.19.0. Although methods using brain switching and IL using GAIL and BC have proven to successfully execute a complex peg transfer task, the time taken to complete the task is much slower than expected of a proficient surgeon. In conclusion, both RL and IL algorithms can be used to train an Agent to conduct laparoscopic tasks. However, further research must be undertaken to optimise the algorithms for training and to improve the fidelity of the simulation.

6 Appendix

6.1 Training Parameters

Parameter Name	Description
Trainer	PPO or SAC trainer
Max steps	Total number of steps in the training session
Learning Rate	Strength of each gradient descent update
Buffer Size	PPO: Number of experiences to collect before policy update SAC: Maximum size of experience buffer
Beta	PPO Specific: Makes the policy more random and ensures agent explores the environment fully
Epsilon	PPO Specific: How rapidly the policy evolves during training – smaller values for stable updates
Initial Entropy Coefficient	SAC Specific: Extent to which the agent should initially explore the environment
GAIL Strength	Gail Specific: How much to multiply the reward given for mimicking demonstration actions
BC Strength	BC Specific: Extent to which BC can influence the policy
Steps	BC Specific: Number of steps after which an agent can begin to generalise a policy
Batch Size	BC Specific: Number of demonstrations used during one update

Figure 24: Description of Training Parameters

6.2 Training Graphs

6.2.1 Localisation Training Statistics - Trial 1

Figure 25: Trial 1 - Value Loss

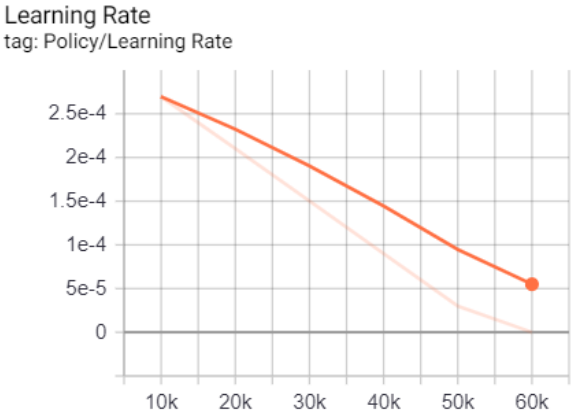


Figure 26: Trial 1 - Learning Rate

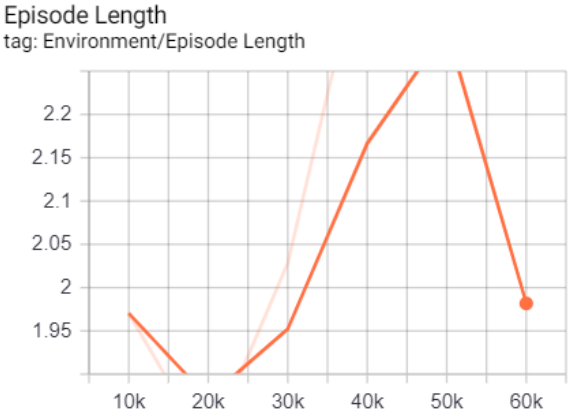


Figure 27: Trial 1 - Episode Length

6.2.2 Localisation Training Statistics - Trial 2

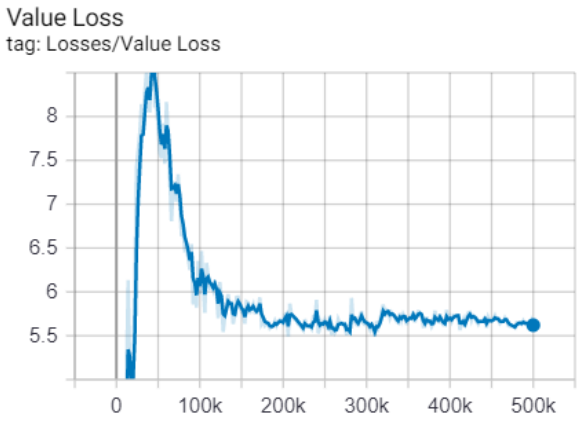


Figure 28: Trial 2 - Value Loss

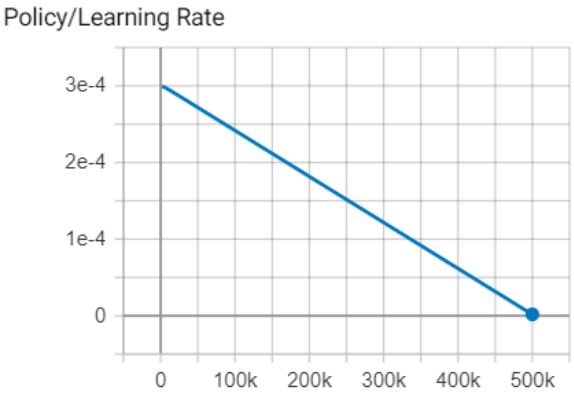


Figure 29: Trial 2 - Learning Rate

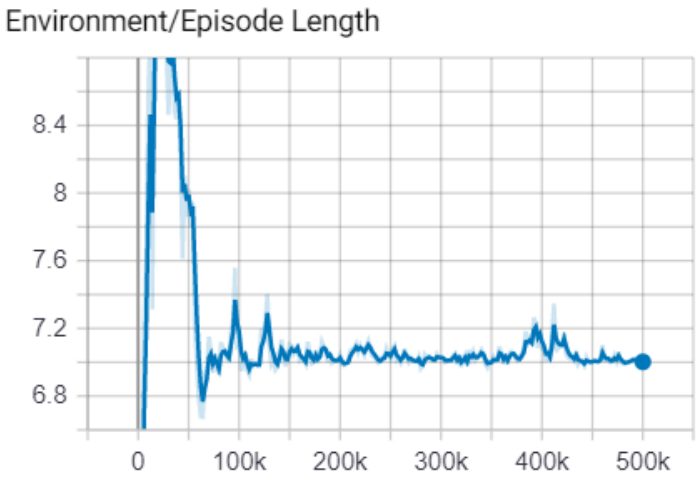


Figure 30: Trial 2 - Episode Length

6.2.3 One Brain Training Statistics (no constraints)

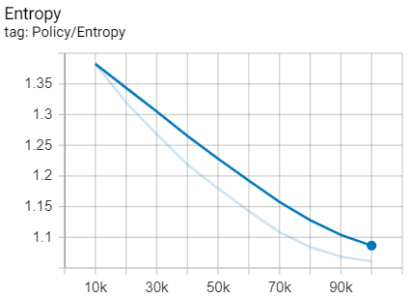


Figure 31: One Brain - Entropy

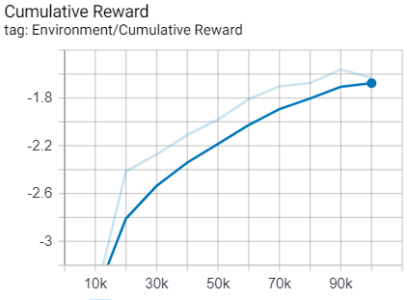


Figure 32: One Brain - Cumulative Reward

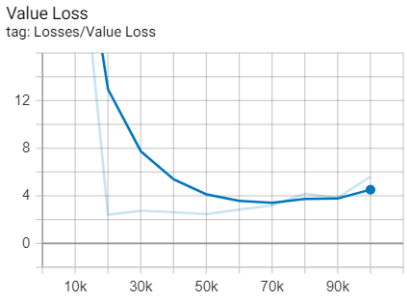


Figure 33: One Brain - Value Loss

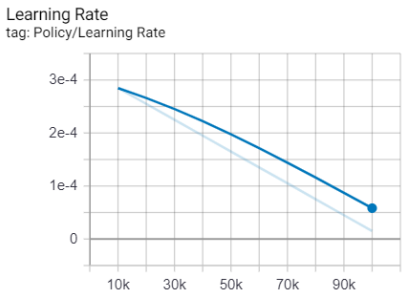


Figure 34: One Brain - Learning Rate

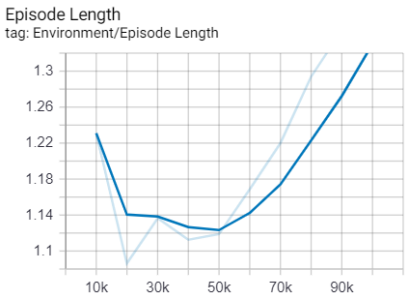


Figure 35: One Brain - Episode Length

6.2.4 PPO-SAC One Brain Training Statistics (with constraints)

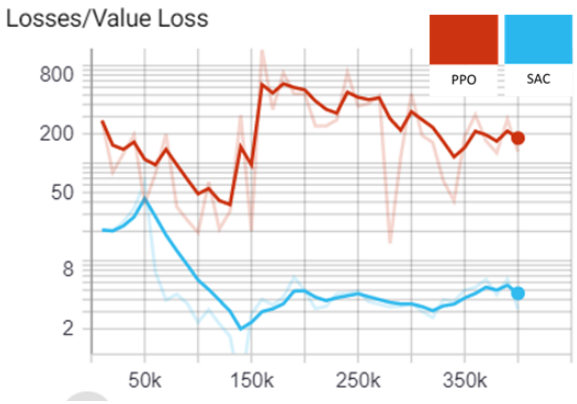


Figure 36: PPO-SAC - Value Loss

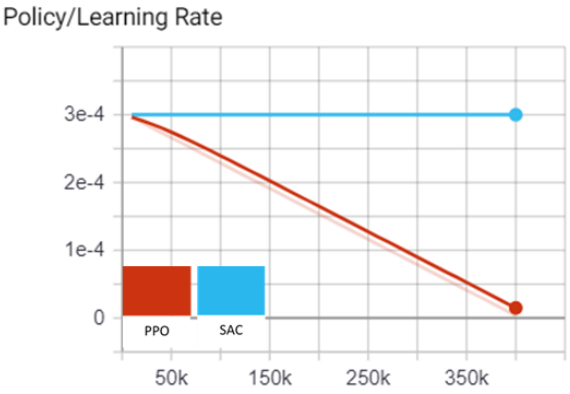


Figure 37: PPO-SAC - Learning Rate

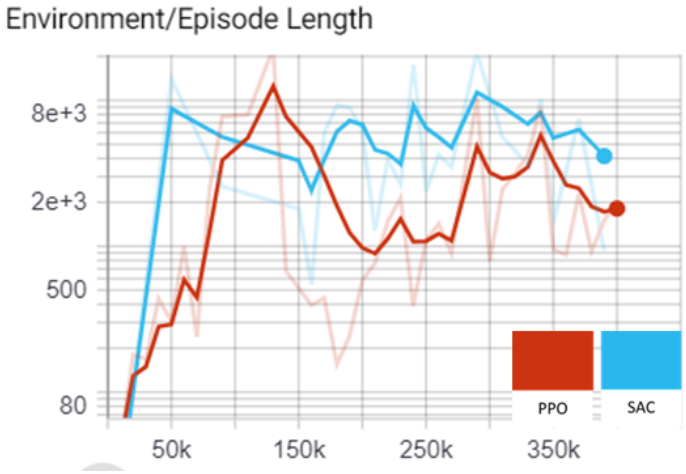


Figure 38: PPO-SAC - Episode Length

6.2.5 Multi-brain Switching Training Statistics

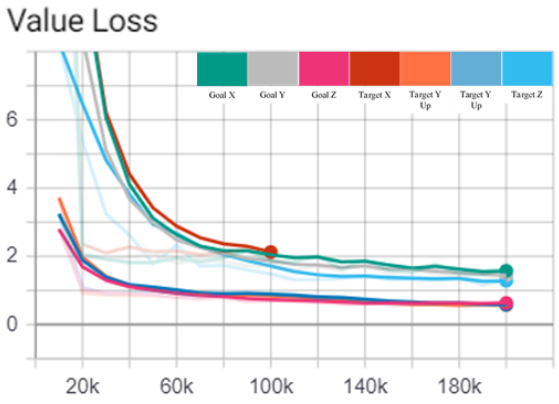


Figure 39: Multi-Brain - Value Loss

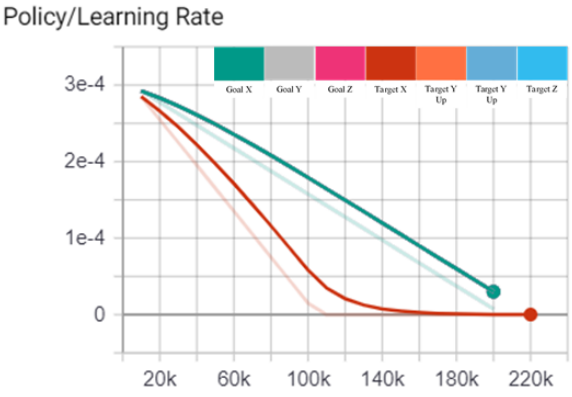


Figure 40: Multi-Brain - Learning Rate

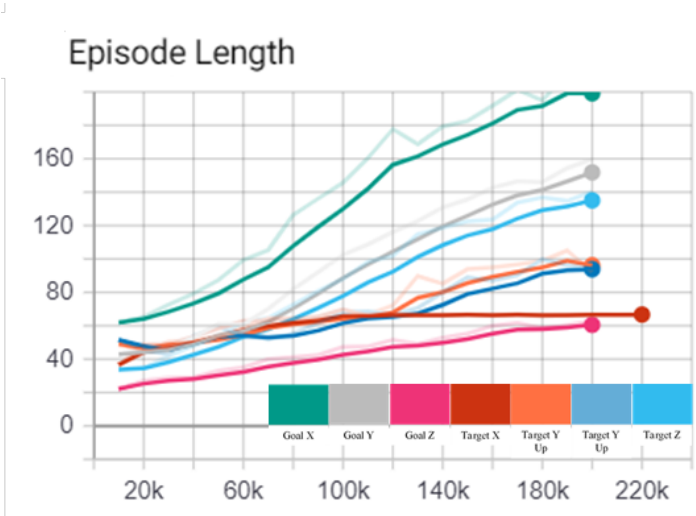


Figure 41: Multi-Brain - Episode Length

6.2.6 IL Training Statistics

Figure 42: IL - Value Loss

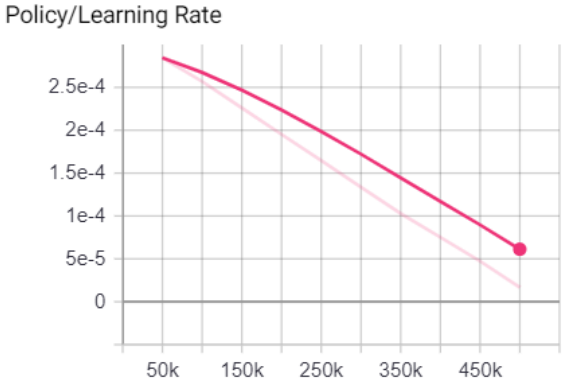


Figure 43: IL - Learning Rate

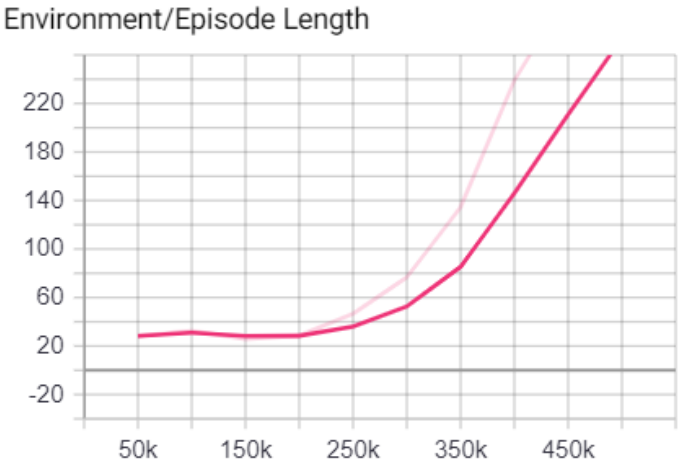


Figure 44: IL - Episode Length

7 Additional Materials

- All project-related materials can be found in this Project Archive
- Videos of the Agent performing in inference mode can be found here.
- Final presentation: here.
- Planning Report: here.

References

[1] Manjunath Siddaiah-Subramanya, Kor Tiang, and Masimba Nyandowe. "A New Era of Minimally Invasive Surgery: Progress and Development of Major Technical Innovations in General Surgery Over the Last Decade". In: *The Surgery Journal* 03.04 (Oct. 2017), e163–e166. ISSN: 2378-5128. DOI: 10.1055/s-0037-1608651. URL: /pmc/articles/PMC5680046/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5680046/.

[2] *Types of Minimally Invasive Surgery (Robotic, Endoscopic, Laparoscopic): Johns Hopkins Medicine in Baltimore, MD*. URL: https://www.hopkinsmedicine.org/minimally%7B%5C_%7Dinvasive%7B%5C_%7Drobotic%7B%5C_%7Dsurgery/types.html (visited on 09/04/2020).

[3] Aslam Ejaz et al. "A comparison of open and minimally invasive surgery for hepatic and pancreatic resections using the nationwide inpatient sample". In: *Surgery (United States)* 156.3 (2014), pp. 538–547. ISSN: 15327361. DOI: 10.1016/j.surg.2014.03.046. URL: /pmc/articles/PMC4316739/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4316739/.

[4] *Laparoscopy (keyhole surgery) - NHS*. URL: https://www.nhs.uk/conditions/laparoscopy/ (visited on 09/04/2020).

[5] H.G. Stassen, J. Dankelman, and C.A. Grimbergen. "Open versus minimally invasive surgery: a man-machine system approach". In: *Transactions of the Institute of Measurement and Control* 21.4-5 (Oct. 1999), pp. 151–162. ISSN: 0142-3312. DOI: 10.1177/014233129902100403. URL: http://journals.sagepub.com/doi/10.1177/014233129902100403.

[6] Michele Tonutti et al. *The role of technology in minimally invasive surgery: State of the art, recent developments and future directions*. Mar. 2017. DOI: 10.1136/postgradmedj-2016-134311. URL: https://pubmed.ncbi.nlm.nih.gov/27879411/.

[7] *Laparoscopic - Minimally invasive & laparoscopic surgery - Dr. Markides*. URL: https://drmarkides.com/laparoscopic-minimally-invasive-amp-laparoscopic-surgery/ (visited on 09/10/2020).

[8] Jeremy L. Emken, Elspeth M. Mcdougall, and Ralph V. Clayman. "Training and assessment of laparoscopic skills." In: *JSLS : Journal of the Society of Laparoendoscopic Surgeons / Society of Laparoendoscopic Surgeons* 8.2 (2004), pp. 195–199. ISSN: 10868089. URL: /pmc/articles/PMC3015539/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3015539/.

[9] Riaz A. Agha and Alexander J. Fowler. "The role and validity of surgical simulation". In: *International Surgery* 100.2 (Feb. 2015), pp. 350–357. ISSN: 00208868. DOI: 10.9738/INTSURG-D-14-00004.1. URL: /pmc/articles/PMC4337453/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4337453/.

[10] Hiten Rh, Patel & Bijen. "Expert Review of Anticancer Therapy Virtual reality surgical simulation in training". In: (2014). ISSN: 1744-8328. DOI: 10.1586/era.12.23. URL: https://www.tandfonline.com/action/journalInformation?journalCode=iery20.

[11] Sandra L. De Montbrun and Helen MacRae. "Simulation in surgical education". In: *Clinics in Colon and Rectal Surgery* 25.3 (2012), pp. 156–165. ISSN: 15310043. DOI: 10.1055/s-0032-1322553. URL: /pmc/articles/PMC3577578/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3577578/.

[12] Neal E. Seymour et al. "Virtual reality training improves operating room performance results of a randomized, double-blinded study". In: *Annals of Surgery*. Vol. 236. 4. Ann Surg, Oct. 2002, pp. 458–464. DOI: 10.1097/00000658-200210000-00008. URL: https://pubmed.ncbi.nlm.nih.gov/12368674/.

[13] Rajesh Aggarwal et al. "A competency-based virtual reality training curriculum for the acquisition of laparoscopic psychomotor skill". In: (2006). DOI: 10.1016/j.amjsurg.2005.10.014.

[14] Ziheng Wang and Ann Majewicz Fey. "Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery". In: *International Journal of Computer Assisted Radiology and Surgery* 13.12 (Dec. 2018), pp. 1959–1970. ISSN: 18616429. DOI: 10.1007/s11548-018-1860-1. arXiv: 1806.05796. URL: https://doi.org/10.1007/s11548-018-1860-1.

[15] *ml-agents/Background-Machine-Learning.md at release_6_docs · Unity-Technologies/ml-agents · GitHub*. URL: https://github.com/Unity-Technologies/ml-agents/blob/release%7B%5C_%7D6%7B%5C_%7Ddocs/docs/Background-Machine-Learning.md (visited on 09/05/2020).

[16] Alexander Winkler-Schwartz et al. "Machine Learning Identification of Surgical and Operative Factors Associated With Surgical Expertise in Virtual Reality Simulation". In: *JAMA network open* 2.8 (Aug. 2019), e198363. ISSN: 25743805. DOI: 10.1001/jamanetworkopen.2019.8363. URL: https://jamanetwork.com/.

[17] Alexander Winkler-Schwartz et al. "Artificial Intelligence in Medical Education: Best Practices Using Machine Learning to Assess Surgical Expertise in Virtual Reality Simulation". In: *Journal of Surgical Education* 76.6 (Nov. 2019), pp. 1681–1690. ISSN: 18787452. DOI: 10.1016/j.jsurg.2019.05.015.

[18] Ngoc Duy Nguyen, Thanh Nguyen, and Saeid Nahavandi. *System Design Perspective for Human-Level Agents Using Deep Reinforcement Learning: A Survey*. Nov. 2017. DOI: 10.1109/ACCESS.2017.2777827.

[19] Kyle Couperus et al. "Immersive Virtual Reality Medical Simulation: Autonomous Trauma Training Simulator". In: *Cureus* (May 2020). ISSN: 2168-8184. DOI: 10.7759/cureus.8062.

[20] *Reinforcement Learning Demystified: Markov Decision Processes (Part 1) — by Mohammad Ashraf — Towards Data Science*. URL: https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes-part-1-bf00dda41690 (visited on 09/09/2020).

[21] Ahmed Hussein et al. "Imitation Learning: A Survey of Learning Methods A:2 A. Hussein et al". In: *ACM Computing Surveys* (). DOI: 10.1145/0000000.0000000. URL: http://dx.doi.org/10.1145/0000000.0000000.

[22] *A brief overview of Imitation Learning — by SmartLab AI — Medium*. URL: https://medium.com/@SmartLabAI/a-brief-overview-of-imitation-learning-8a8a75c44a9c (visited on 09/09/2020).

[23] Chee Kong Chui et al. "Learning laparoscopic surgery by imitation using robot trainer". In: *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011* February 2019 (2011), pp. 2981–2986. DOI: 10.1109/ROBIO.2011.6181759.

[24] Hongbign Wang et al. "Integrating reinforcement learning with multi-agent techniques for adaptive service composition". In: *ACM Transactions on Autonomous and Adaptive Systems* 12.2 (May 2017). ISSN: 15564703. DOI: 10.1145/3058592.

[25] Xiaoyu Tan et al. "Robot-Assisted Training in Laparoscopy Using Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 485–492. ISSN: 23773766. DOI: 10.1109/LRA.2019.2891311.

[26] Annalaura Lerede and David Escobar-Castillejos. "Simulation of autonomous surgical tasks in the Unity – ML-Agent – PPO framework Supervisor :" in: (2019).

[27] Author Szu-yin Chen. "Final Report for MSc Project Machine Learning and Feedback in Virtual Surgical Training Environments". In: September (2019).

[28] Arthur Juliani et al. *Unity: A General Platform for Intelligent Agents*. Tech. rep. arXiv: 1809.02627v2. URL: https://github.com/Unity-Technologies/ml-agents.

[29] *ml-agents/ML-Agents-Overview.md at master · Unity-Technologies/ml-agents · GitHub*. URL: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md (visited on 09/05/2020).

[30] *On-Policy v/s Off-Policy Learning — by Abhishek Suran — Jul, 2020 — Towards Data Science*. URL: https://towardsdatascience.com/on-policy-v-s-off-policy-learning-75089916bc2f (visited on 09/05/2020).

[31] *Proximal Policy Optimization*. URL: https://openai.com/blog/openai-baselines-ppo/ (visited on 09/05/2020).

[32] *Soft Actor Critic—Deep Reinforcement Learning with Real-World Robots – The Berkeley Artificial Intelligence Research Blog*. URL: https://bair.berkeley.edu/blog/2018/12/14/sac/ (visited on 09/09/2020).

[33] Ian J Goodfellow et al. *Generative Adversarial Nets*. Tech. rep. URL: http://www.github.com/goodfeli/adversarial.

[34] *Training your agents 7 times faster with ML-Agents - Unity Technologies Blog*. URL: https://blogs.unity3d.com/2019/11/11/training-your-agents-7-times-faster-with-ml-agents/ (visited on 09/10/2020).

[35] *FLS Manual Skills Written Instructions and Performance Guidelines*. Tech. rep.

[36] William Agnew and Pedro Domingos. *Self-Supervised Object-Level Deep Reinforcement Learning*. Tech. rep. arXiv: 2003.01384v1.

[37] *ml-agents/Using-Tensorboard.md at release_6_docs · Unity-Technologies/ml-agents · GitHub*. URL: https://github.com/Unity-Technologies/ml-agents/blob/release%7B%5C_%7D6%7B%5C_%7Ddocs/docs/Using-Tensorboard.md (visited on 09/05/2020).

[38] *Maximum Entropy Policies in Reinforcement Learning & Everyday Life — by Arthur Juliani — Medium*. URL: https://medium.com/@awjuliani/maximum-entropy-policies-in-reinforcement-learning-everyday-life-f5a1cc18d32d (visited on 09/09/2020).

[39] *Unity - Scripting API: Transform.parent*. URL: https://docs.unity3d.com/ScriptReference/Transform-parent.html (visited on 09/10/2020).

[40] *Fisher–Yates shuffle - Wikipedia*. URL: https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates%7B%5C_%7Dshuffle (visited on 09/05/2020).

[41] *Training simulators for laparoscopy — LapSim® — Surgical Science*. URL: https://surgicalscience.com/systems/lapsim/ (visited on 09/05/2020).

[42] John Schulman et al. *Trust Region Policy Optimization*. Tech. rep. arXiv: 1502.05477v5.

[43] Thanh Nguyen et al. "A new tensioning method using deep reinforcement learning for surgical pattern cutting". In: *Proceedings of the IEEE International Conference on Industrial Technology*. Vol. 2019-February. Institute of Electrical and Electronics Engineers Inc., Feb. 2019, pp. 1339–1344. ISBN: 9781538663769. DOI: 10.1109/ICIT.2019.8755235. URL: http://arxiv.org/abs/1901.03327%20http://dx.doi.org/10.1109/ICIT.2019.8755235.