2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9340913

Ultrasound-Guided Robotic Navigation with Deep Reinforcement Learning

Hannes Hase*,1, Mohammad Farid Azampour*,1,2, Maria Tirindelli[1], Magdalini Paschali[1], Walter Simson[1], Emad Fatemizadeh[2] and Nassir Navab[1,3]

***Abstract*— In this paper we introduce the first reinforcement learning (RL) based robotic navigation method which utilizes ultrasound (US) images as an input. Our approach combines state-of-the-art RL techniques, specifically deep Q-networks (DQN) with memory buffers and a binary classifier for deciding when to terminate the task.**

Our method is trained and evaluated on an in-house collected data-set of 34 volunteers and when compared to pure RL and supervised learning (SL) techniques, it performs substantially better, which highlights the suitability of RL navigation for US-guided procedures. When testing our proposed model, we obtained a 82.91% chance of navigating correctly to the sacrum from 165 different starting positions on 5 different unseen simulated environments.

I. Introduction

The rise of robotics and their gradual permeation into the field of medicine is a revolution on its own. By integrating robotic systems in the medical work-space, doctors are enabled to treat individual patients in a more efficient, safer and less morbid way. However, end-to-end automated approaches are constrained by the adaptability to unexpected situations and the poor judgment of robotic systems [1].

With ever-improving ultrasound (US) technology, US is being increasingly used in diagnostics and interventions. Unlike other modalities like computed tomography (CT), US provides real-time dynamic physiologic information while being radiation free and comparatively cheap. Yet, the quality of an US image suffers from artifacts such as speckle and clutter, has a low signal to noise ratio and is strongly subject dependent [2]. Another downside is the high inter-observer variability when acquiring US images, which calls for trained sonographers to guarantee clinically relevant images. It is the lack of specialists that opens the need for robotic imaging techniques [3]. The mentioned difficulties associated with US imaging make the task of autonomous US navigation extremely challenging.

Robotic ultrasound (rUS) in the medical field has been investigated to improve working conditions for doctors and also to increase the accuracy of interventions [4], [5]. Tirindelli et al. in [6] attempt to automate spinal navigation by using a combination of force data and US image. However, this procedure still requires to be set-up by a technician. Automatic navigation towards specific positions without any human intervention on the human body is still not resolved, to the best of our knowledge.

Reinforcement learning offers an interesting and novel approach, as it excels at sequential decision making and exploratory tasks [7]. Reinforcement learning has shown superhuman performance on Atari games [8] in which the agent only decides what to do based on visual input. This has already been translated to real-life applications in visual robotic manipulation, such as the general task of grasping [9] or in visual navigation for humanoid robots playing soccer [10]. Even in the medical field, initial attempts have been made to exploit the strengths of RL. For instance, [11] proposes to use RL to find landmarks in fetal magnetic resonance imaging (MRI) scans, in order to improve 3D-imaging.

With the goal of expanding the applications of RL in the medical sector, we work towards the full automation of spinal navigation exclusively relying on US images for the decision making. Towards this end we propose a method using a combination of RL with a memory buffer for improved state estimation and a separate binary classifier for goal state detection to counter the sparse reward problem.

In detail, our contributions are:

1) The acquisition of an in-house data-set of lower back US sweeps on volunteers using a robot for accurate tracking of the frames.
2) Training an RL agent on simulated lower-back environments to find correct views of the sacrum while navigating the environments only relying on US frames.

II. Related Work

A. Deep Reinforcement Learning

RL is one of the three main paradigms, of machine learning, alongside supervised and unsupervised learning [7]. In RL, an agent interacts with an environment and aims at maximizing an accumulated reward that results from its actions. Arulkumaran et al. provides a comprehensive overview of the developments of deep reinforcement learning (DRL) [12]. In RL an agent is trained to complete a task via specialization in goal-directed learning. An environment is modeled in which the agent can explore and associate actions with rewards and thus, learn how to achieve the defined goal [7]. For matters of this study, we discuss DRL further in the methodology section.

*These authors contributed equally to this work
[1]Computer Aided Medical Procedures, Technische Universität München, Munich, Germany hannes.hase@tum.de
[2]Sharif University of Technology, Tehran, Iran mf.azampour@tum.de
[3]Computer Aided Medical Procedures, John Hopkins University, Baltimore, MD, USA

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

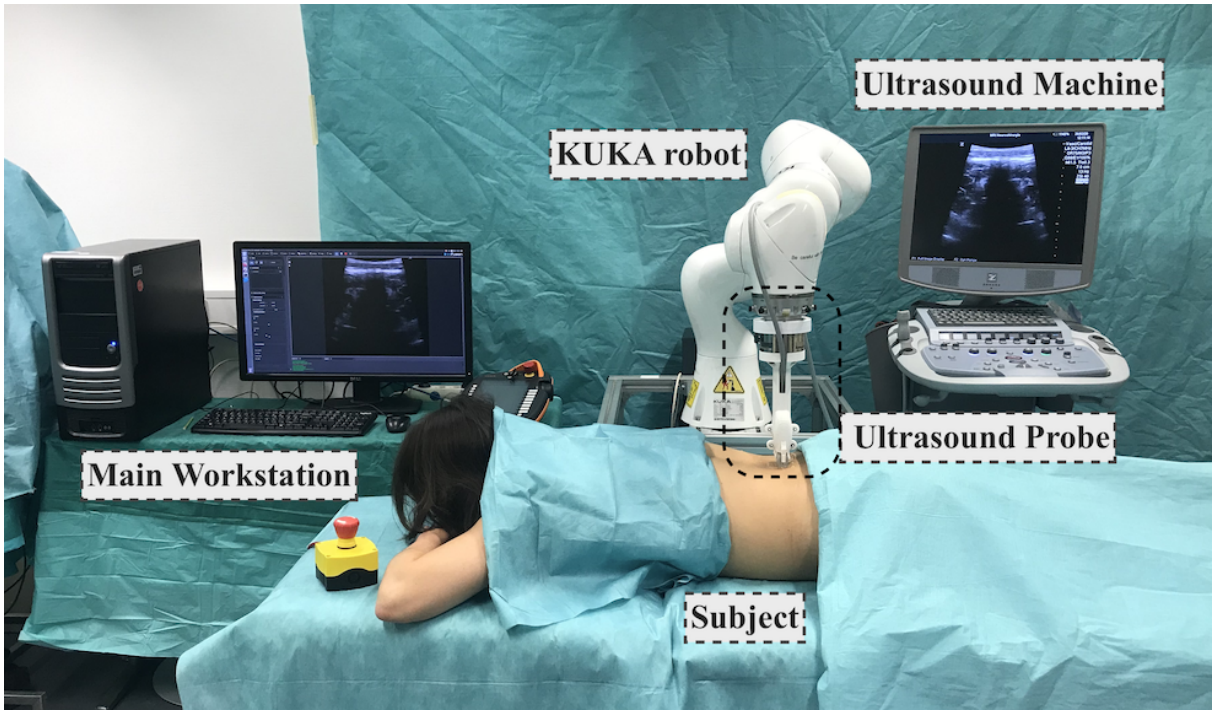

Fig. 1. Setup for robotic ultrasound acquisition. Ultrasound probe is attached to the robot end-effector using a 3D printed holder. Main workstation will store the frames acquired by the US machine alongside the tracking data from the robot.

B. Reinforcement Learning for Robotic Manipulation

Vision-based robotic manipulation with reinforcement learning is first investigated in [13]. Zhang et al. train an agent to autonomously steer a robot to reach a target using raw pixels as the sole input. While training and testing using simulated environments provide promising results, their approach fails when transferred to real-world applications. In [9], the authors propose a benchmark for the general task of grasping using popular RL methods like deep Q-learning (DQL) and deep deterministic policy gradient (DDPG). Based on their results, DQL translates into more stable agents in case of small data-sets, whereas Monte Carlo methods provide better results on larger sets. They report a success-rate of 50% on a relatively small data-set of 10k samples.

C. Reinforcement Learning in Medicine

Chu et al. combine online SL and RL for improving the efficiency of breast cancer diagnosis in clinics on multi-modal data [14]. The online SL assesses breast cancer risk based on the available patient data and examinations. The doctor then decides if the confidence of the diagnostic was high enough. If the confidence is not enough, the RL part of the framework recommends the next best measurements or exams that would improve the diagnostics' confidence.

Initial exploratory works have experimented with visual RL for medical applications. Milletari et al. [15] successfully propose DRL to perform action suggestion for sonographer guidance. In this seminal work a DRL agent successfully learns a policy to guide inexperienced medical personnel to obtain clinically relevant cardiac ultrasound images of the parasternal long-axis view. The authors simulate the RL environments by projecting a grid on subjects' chests and populating the grids' sectors or bins with in-vivo US-frames collected on a set of volunteers. At inference time, the user acts as the agent and is provided motion recommendations by the RL-policy; manually closing the loop of navigation. Building on this work, we close the agent-policy loop by adding a robotic actuator to manipulate the ultrasound probe based on the RL-policy. Additionally, we improve the DQN by adding memory to the model and using a binary classifier for stopping.

III. Methodology

A. Reinforcement Learning

RL-problems are often modeled as Markov Decision Processes (MDP). A MDP is a sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards. It consists of a set of states S, a set of actions for each state $\mathcal{S}_a$, a transition model $P(s'|s,a)$ and a reward function $R(s)$ [7]. In our work, the agent relies exclusively on visual input in the form of an US frame. Thus, the agent does not explicitly know its state and needs to estimate it. This turns the problem into a partially observable MDP (POMDP).

B. Deep Q-Learning

Q-Learning is a form of model-free off-policy RL that enables agents to learn optimal behavior in Markovian domains. The agent learns to estimate Q-values, defined as the

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

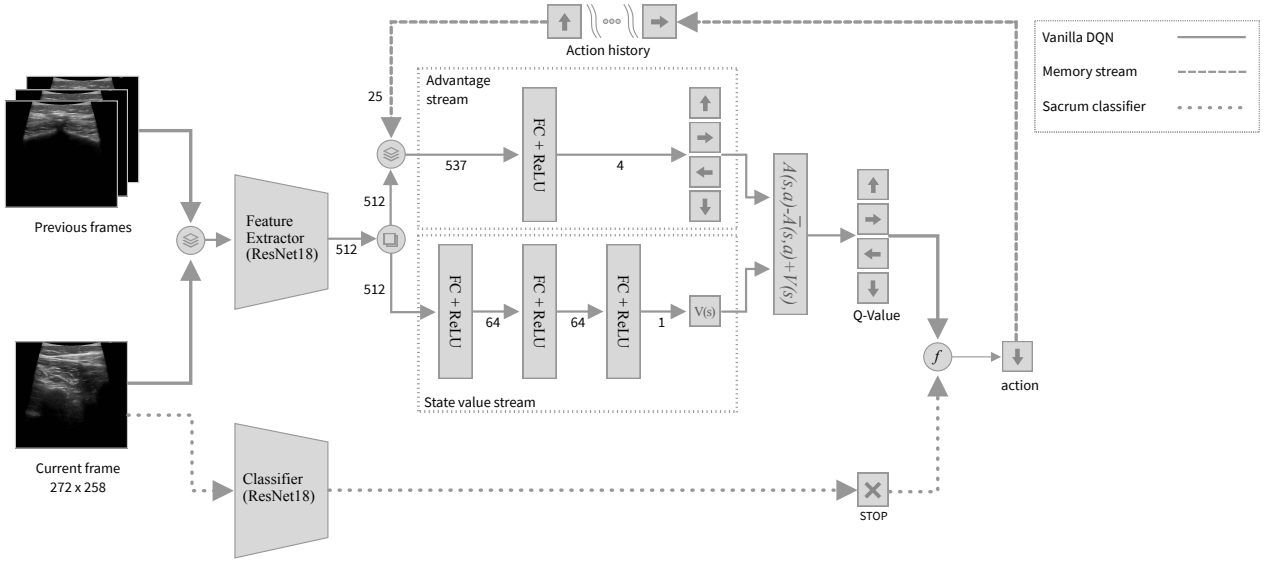

Fig. 2. Overall network architecture. The solid arrow represents the V-DQN. The broken line displays the added memory for the M-DQN. The dotted line shows the binary classifier for stopping of the VS-DQN and MS-DQN. When not using the binary classification network for stopping, the stop action becomes part of the Q-value layer as a fifth value. The action history is composed by five concatenated one-hot-encoded representations of previous actions. The action-space is composed of five possible actions, making this stream have a length of 25.

long term reward of performing a certain action in a given state [16]. An RL-agent is trained by exposing it to random transitions represented by the tuple (s, a, r, s'), where s, a, r are the state, the chosen action and the obtained reward at step t and s' is the state at $t+1$. The transitions are acquired by the agent while interacting with the environment and stored in a replay memory to break temporal correlations. The training batches are sampled from the replay memory and fed into the DQN for training. The Q-values are learned by iteratively improving the estimates based on the results of the interaction with the environment, following the equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad (1)$$

where α corresponds to the learning rate and γ to the discount factor.

When the model converges to an optimal solution, we get the optimal action for a state s by doing $argmax(Q(s,a))$.

The main difficulty of Q-learning's traditional look-up table method is successfully learning in environments with large state-spaces. Mnih et al. [8] propose Deep Q-learning (DQL) as a solution to this issue by approximating the Q-values with neural networks in the context of training a RL-agent to play Atari video-games.

We improved the base DQN by including:

1) **Double Deep Q-Network (DDQN)**: The base DQN setup is difficult to train because the model's neural network (NN) is used for computing at the same time the prediction and the target, leading to the targets changing at each training step and making the training unstable. This is solved by copying the DQN into a second network referred to as the target network, where the weights are fixed and updated based on the current DQN's weights every N training steps. By doing this, we avoid Q-value over-estimations and achieve a more reliable training. [17]
2) **Dueling DQN**: Wang et al. in [18] introduce the splitting of the Q-value estimation into two streams, as shown in Fig. 2. On the one hand, the advantage-value stream $A(s,a)$, estimates the short-term reward that is achievable with each available action. On the other hand, the state-value stream estimates the long-term reward that is possible from that state. The Q-values are then computed as detailed in Eq. 3.
3) **Prioritized Replay Memory**: The time-difference or TD-error is defined in Q-learning as:

$$TD = r + \gamma \max_{a'} Q_{target}(s',a') - Q(s,a) \quad (2)$$

and represents a measure of how unsuspected the transition used for training is. When sampling transitions for training, the transitions probability of being selected is dependent on its TD-error. Hereby, transitions with relevant information are prioritized for training. [19]

This setup we call *V-DQN*. We define I_t as the input frame at time t, $\phi(\cdot)$ as the feature extractor, f_v and f_A as the value and action advantage estimators, respectively. The Q-values of the V-DQN model are a function of the current frame following Eq. 3. Here, $A(s,a)$ represents the action-state value estimates, $\bar{A}(s,a)$ is the average action-state value and $V(s)$ corresponds to the state-value estimate.

$$\begin{aligned} V(s) &= f_v(\phi(I_t)) \\ A(s,a) &= f_A(\phi(I_t), a) \\ Q(s,a) &= A(s,a) - \bar{A}(s,a) + V(s) \end{aligned} \quad (3)$$

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

In this work, we add two input streams of previous transitions in the environment. The first one corresponds to the previous frames, as done in [8]. For the second one, we adapt the method proposed by [20] to take previous actions into account. Eq. 4 defines the Q-value estimation with the modified input streams.

$$\begin{aligned} \Phi_t &= \phi(I_t, I_{t-1}, ..., I_{t-n}) \\ V(s) &= f_v(\Phi_t) \\ A_{s,a} &= f_A(\Phi_t, a, (a_{t-1}, ..., a_{t-m})) \end{aligned} \tag{4}$$

The extracted features Φ_t from the current and previous frames are passed to the value estimator. $A(s, a)$ is defined by the action advantage estimator parameterized by the extracted features and previous actions. The actions are fed to the model as concatenated one-hot-encoded vectors [21]. The setup is referred to as *M-DQN*.

In order to address the sparsity of situations with valid stopping criteria the agent is exposed to (finding itself in a goal bin), we add a binary classifier to determine when the stopping criteria has been reached. By doing so, we modify the reward function detailed in Table I by removing the stopping decision. We call this *VS-DQN* or *MS-DQN* depending on whether the agent has memory or not.

We train the feature extraction for all RL models and the binary classification network using a ResNet18 architecture [22]. Feature extraction is performed by removing the batch-normalization layers and the final average pooling layer to feed raw features into the state and advantage value estimators.

C. Problem setting

With this work we aim at teaching an RL agent to successfully find the sacrum reacting only on information gained from US frames received, while navigating in the spinal region. In other words, we aim to solve a search task with two degrees of freedom (DoF), on the dorsal plane of a subject. To state our problem as an POMDP, we define the following terms:

1) Action space: The action space $\mathcal{S}_a$ is comprised of the actions $up, down, left, right, stop$ in the V-DQN and M-DQN. In the case of VS-DQN and MS-DQN, the $stop$ action is triggered by the binary classifier $f_{\text{stop}}()$.

2) State: The state of the environment is defined as the probe's position relative to the sacrum in the dorsal plane. The state is fully defined by the position, thus complying with the Markovian property of the problem setting and the feasibility of using MDPs.

3) Observation: As our problem setting is modeled by a POMDP, the state is not known to the agent and needs to be estimated based on an observation $O(s)$ in the form of an US frame it receives from the environment. The observations are defined by the state the agent finds itself in, while the observation defines the best action chosen by the agent. Therefore, we can say that an agent that can estimate its state correctly is an agent that understands its environment and is more likely to successfully navigate towards its goal. In our problem setting, the randomness in the observations comes from the anatomical differences of the subjects and eventual interference during data acquisition.

4) Reward function: We label bins that contain frames showing the sacrum as correct and defined numerical rewards given to the agent depending on direction of the actions in relation to the goals. The used reward function is detailed in Table I. The reward function heavily punishes incorrect stopping, as this would terminate the exploration in a wrong position. It also penalizes getting caught in back and forth movements, as by that behavior the agent would accumulate a net negative reward over time.

TABLE I
DISCRETE REWARD FUNCTION FOR TRAINING THE AGENT

Situation	Reward
Move closer	0.05
Move away	-0.1
Correct stop	1.0
Incorrect stop	-0.25

5) Simulated robot navigation implementation: We conduct simulated testing, by initializing our test environments at determined positions or states. We face our models with US frames obtained at that state and acted on the environment based on the action chosen by the agent. The simulated navigation is implemented as explained in Alg. 1.

Algorithm 1: Simulated Robot Navigation

Result: MS-DQN Robotic Navigation

$s_t = int(rand() * 164)$; // init state
$t = 0$;
$t_{max} = 20$;
F = [] ; // frame memory buffer
A = [] ; // action memory buffer
while $t < t_{max} \wedge a_t \in \mathcal{S}_a$ **do**
  $O_t = f_{Env}(s_t)$; // US frame
  $a_t = f_{\text{stop}}(O_t)$; // check stop
  if $a_t \neq$ *stop* **then**
    $a_t = argmax(f_{MS-DQN}(O_t))$; // action
  end
  if $a_t ==$ *stop* **then**
    break ; // sacrum reached
  else
    $s_{t+1} = Env(a_t)$; // update state
  end
  F[t] = O_t ; // frame to buffer
  A[t] = a_t ; // action to buffer
  $t = t + 1$
end

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

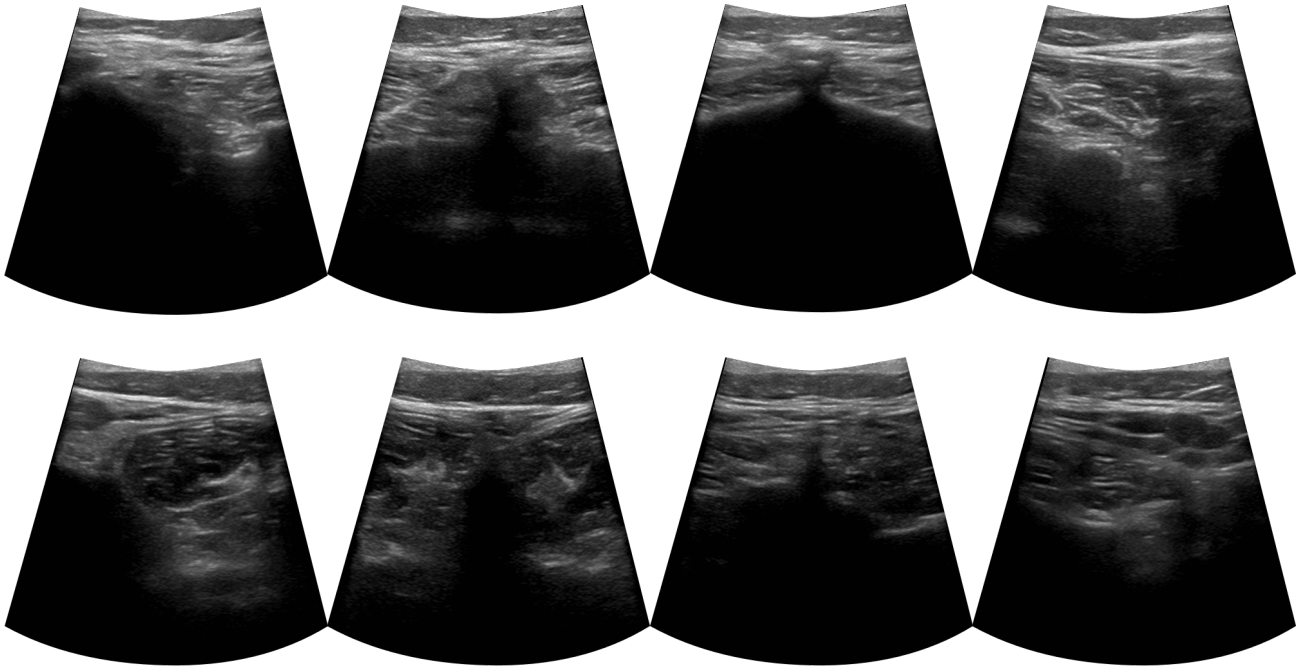

Fig. 3. The images above display exemplary US image samples from two of the subjects in the data-set. Each row belongs to one subject. The images correspond to (a) Left posterior pelvis; (b) L3 vertebra; (c) Sacrum; (d) Right lumbar region. In Fig. 4, the position of each frame in the projected grid on subjects is shown. These images show the variability of the same anatomical structure as seen in the US images between different subjects.

IV. Experimental setup

A. Project setup

For data acquisition, we use a 7-axis robot certified for human interaction of the model KUKA LBR iiwa 7 R800 manipulator (KUKA Roboter GmbH, Augsburg, Germany). The robot control runs on the Robotic Operating System (ROS)[1] using a custom software interface developed in our lab[2]. The Ultrasound probe is attached to the end-effector with a 3D-printed mount. To receive the US-frames, we used an Epiphan DVI2USB 3.0 frame-grabber (Epiphan Systems Inc. Palo Alto, California, USA) with a resolution of 800×600 pixels and a sampling frequency of 30 fps. We control the robot and process the images from a fixed workstation (Intel i5, NVIDIA GeForce GTX 1080). The image processing and robot control are implemented via custom software plugins integrated into the visualization framework ImFusionSuite[3] platform (ImFusion GmbH, Munich, Germany).

Ultrasound acquisitions are performed with a L8-3 linear US transducer and a Zonare z.one ultra sp Convertible Ultrasound System (ZONARE Medical Systems, Inc., Mountain View, California, United States). The imaging depth is set to 70 mm and an overall image gain of 90%. The robot is used with a compliant force control set to a maximum applied force of 2 N in the z axis.

[1]http://www.ros.org/
[2]https://github.com/IFL-CAMP/iiwa_stack
[3]https://www.imfusion.de/

B. Data-set

Our data-set[4] collected in-house is comprised of US scans from the lower back of 34 volunteers in total. Each scan consists of eleven 45mm long sweeps parallel to the spine with an horizontal off-set of 1 cm. We divide each sweep into 15 equally long segments and mapped the acquired frames to a grid of 11×15 bins. Under the this grid configuration, each bin contemplates an area of approximately 10×30 mm. This elongated shape of the bin comes from the type of US-probe we used for acquisition. We fill each bin with five frames the agent would encounter when finding itself in that area. With this grid, we can simulate x-y navigation of the environment for training and testing the performance of the agent. In Fig. 3, we showcase different frames the agent could encounter in the grid.

We build one training set of 25 subjects containing a variety of acquisition qualities (artifacts, low quality acquisitions, hard to recognize anatomies) to assure the model would be exposed to non-ideal training data. Because we want to study the feasibility of the approach we assemble a set of nine subjects with high quality scans, to reduce randomness associated to image acquisition. From this high quality set we sample our validation and testing sets with four and five environments respectively. We show the difference of the frames in Fig. 3.

[4]https://hhase.github.io/sacrum_dataset

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

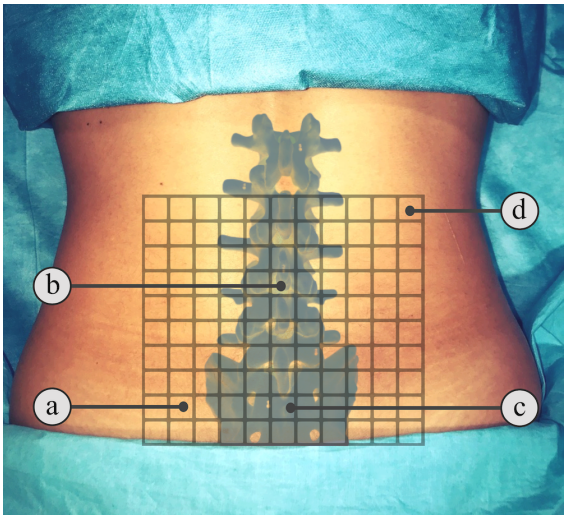

Fig. 4. Frame grid projected on the back of one of our volunteers. Here we show how the grid is positioned over the spine. The letters indicate where each sample frame in Fig. 3 is approximately located.

C. Implementation

1) Framework setup: Our framework is written on the deep learning (DL) library Tensorflow and extends RL-zoo [23] and stable-baselines [24]. Our code is publicly available on Github [5].

2) Model Training: For training our models, we randomly initialize the agent in a random training environment and give it 50 attempts or *steps* to reach the goal. We define this process as a *training episode*. The training episode is terminated when either the agent chooses the *stop* action or reaches the maximal permitted amount of steps. While training, the agent follows an ϵ-greedy policy, meaning that the agent has a probability ϵ of behaving randomly, instead of choosing the action associated with the highest Q-value. By this, we address the exploration-exploitation dilemma [7], giving the agent a possibility to explore its environment to find eventual long term rewards. ϵ decays to 0.02 at a third of the total duration of the training.

For the binary classification model for stopping, we assign the frames containing a correct view of the sacrum to one class and the rest to another. For training, we over-sampled the underrepresented class (frames containing the sacrum) to compensate for the class imbalance. We augment the data-set with rotations and re-sized crops to generalize better. With this network, we obtain consistent accuracy of over 99% on the test set.

Regarding the baseline, we use a standard DenseNet-121 architecture [25] to train a classification network, where the predicted class corresponds to the chosen action.

3) Metrics: For testing our models, we initialize the agent in each of the 165 possible states of the unseen environments and give the agent 20 actions to reach the goal. We call each of these tests a *run*.

[5]https://github.com/hhase/spinal-navigation-rl

As results, we report two performance indicators: policy correctness and reachability. To compute the policy correctness we define n_c as the number of correct actions taken in the run r and n_t as the number of total actions taken in the run on environment e. E is the total number of test environments and R is the total amount of runs tried on each of them. The policy correctness is computed as detailed in Eq. 5.

$$\text{correctness} = \frac{1}{ER}\sum_{e=0}^{E}\sum_{r=0}^{R}\frac{n_c(e,r)}{n_t(e,r)} \tag{5}$$

We define reachability as the ratio between runs that lead the agent to a stopping decision in a goal bin and the total number of runs. A run is not considered successful if the agent ends up in a goal bin but fails to stop. To compute reachability we define g as a boolean variable that is 1 if the goal is reached in run r on environment e and 0 if not. To compute the reachability we use Eq. 6.

$$\text{reachability} = \frac{1}{ER}\sum_{e=0}^{E}\sum_{r=0}^{R} g(e,r) \tag{6}$$

In Fig. 5, we show an example of a successfully testing run. In the case of this run, $n_c = n_t = 5$, as all the actions are taken in direction of the goal. Regarding reachability, $g(e,r) = 1$, because the agent successfully found the sacrum.

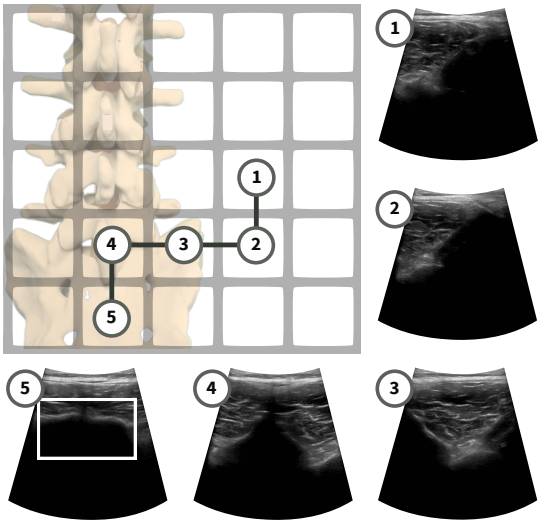

Fig. 5. Possible navigation sequence the agent would follow starting on the right lumbar region. The corresponding frames to the visited bins are labeled with the step number. In step number five the agent identifies a goal state. The sacrum is enclosed in the white bounding box.

V. Results and Discussion

We choose the best model in each case, based on the median reachability value achieved on the validation set. We find that the median gives a more reliable measurement of the performance of the model, given the small validation set and strong subject dependency on the performance.

To begin with discussing the results from Table II, we can see that the V-DQN is outperformed by the M-DQN,

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

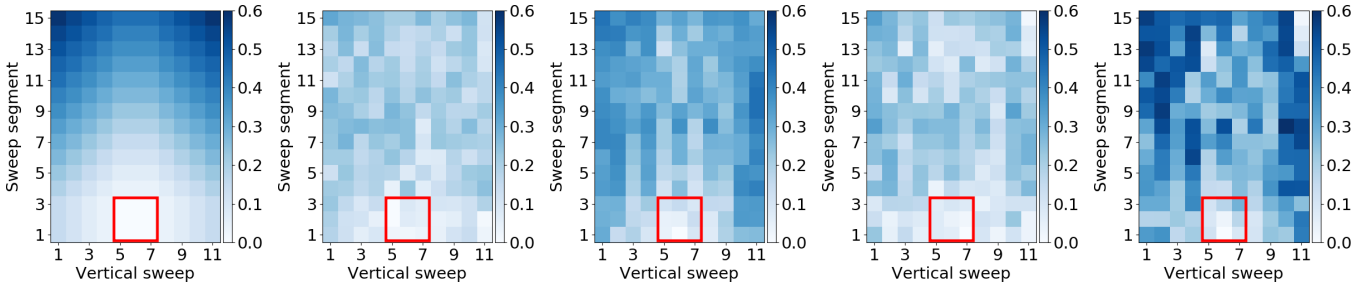

Fig. 6. State value estimate maps. (a) corresponds to the ideal state-value map serving as the ground truth for comparison. (b) is estimated with the V-DQN, (c) with the VS-DQN, (d) with the M-DQN and (e) with the MS-DQN. For these images, we subtracted the minimum state-value estimate of each map, to be able to compare them with VS-DQN and MS-DQN, as this setups do not have rewards associated with stopping. The red bounding boxes show the goal bins.

TABLE II

PERFORMANCE OF THE DIFFERENT PROPOSED ARCHITECTURES

NN architecture	Policy correctness	Reachability
Classification CNN	58.42%	59.64%
V-DQN	55.37%	18.30%
VS-DQN	72.01%	58.06%
M-DQN	49.49%	36.97%
MS-DQN	**79.53**%	**82.91**%

by 20% when it comes to reachability. We attribute this to the inclusion of previous frames and actions. Now, the agent can recognize when it is stuck in a loop and break out of it. Therefore, the M-DQN can perform substantially better than the V-DQN in that aspect. However, the V-DQN still outperforms the M-DQN in terms of policy correctness by 6%, and we can attribute this fact that the memory makes agent of the M-DQN follow sub-optimal paths when navigating towards the goal.

The addition of the binary classifier for stopping leads to significant improvements in the V-DQN and M-DQN variant. When combining both improvements in the MS-DQN, our proposed approach substantially outperforms the other baselines in both, policy correctness by 5 to 30% and in reachability by 40 to 60%.

These results signify the fact that the proposed RL approach is suitable for the task at hand since it delivers promising results in a challenging task like navigating the spinal region and successfully localizing the sacrum. We attribute the improvement to the inclusion of the binary classifier for stopping because, in our problem statement, the stopping action is the most difficult to achieve for pure DQL. This difficulty arises because during the initial exploration phase during training, when following the ϵ-greedy policy with a high probability of choosing random actions, the stopping action is most likely to be incorrect and thereby, heavily punished. Also, because the reward function assigns comparatively large positive and negative rewards to the stopping action, the agent learns to avoid to stop when not entirely confident. The inclusion of a prioritized replay memory trying to counter the sparsity of transitions leading to a successful stop does not solve this shortcoming.

When looking at the classification network approach, we find that by not having memory, the classification agent easily gets stuck in loops and does not reach the goal. However, it proves to have better results when comparing to our V-DQN as its RL counterpart, as it is easier to train a classification network than a DQN. The difference between SL and RL in visual navigation lays in the fact that SL decides the next-best-action based on features extracted from the input frame. In contrast, RL selects actions based on the estimated reward it can achieve from the state it is on. Nonetheless, comparing our proposed DQN setup with the classification network, the results still highlight the advantage of RL for navigation tasks.

A determining factor of the performance an RL agent has on unseen environments is the capability to correctly estimate the state it is in, as this gives the agent a notion on the value of its position within the environment. In Fig. 6, we show the state-value estimates on the same test environment for each of our DQN models and compare them to the ideal case. When comparing the ranges of the values on the different state value maps, we see that the only model achieving a similar range in values as the ground truth is our proposed MS-DQN. The differences in estimation performance across models also reflect the results from Table II.

Besides the differences in the state-value estimations, we can see that it is hard to estimate state-values in unseen environments accurately. However, the ultimate goal of our models is mapping US-frames to actions. The information about the best action choice is contained in the advantage-value estimates, meaning that the agent is still able to take correct actions, despite being wrong about its state.

As shown in our results, however, pure RL struggles on its own with issues like reward sparsity and performance in unseen environments. Solving specific shortcomings of RL with SL proves to be very beneficial and needs to be explored further.

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.

VI. CONCLUSIONS

In this paper, we introduce a reinforcement learning-based ultrasound-guided robotic navigation. Despite the large anatomical variability within our volunteers, in a challenging task of spinal navigation to locate the sacrum, we showcased the superiority of our proposed approach against DQN and classification baselines. Introducing a binary classifier for deciding when to stop, brought substantial improvement to the method. Better results can likely be obtained by increasing our data-set. To move forward to an online implementation in a medical setting an ethical approval would be needed.

REFERENCES

[1] R. H. Taylor, "A perspective on medical robotics," *Proceedings of the IEEE*, vol. 94, no. 9, pp. 1652–1664, Sep. 2006, ISSN: 1558-2256.

[2] A. Hindi, C. Peterson, and R. G. Barr, "Artifacts in diagnostic ultrasound," *Reports in Medical Imaging*, vol. 6, pp. 29–48, 2013.

[3] J. Guo, H. Li, Y. Chen, P. Chen, X. Li, and S. Sun, "Robotic ultrasound and ultrasonic robot," *Endoscopic Ultrasound*, vol. 8, p. 1, Jan. 2019.

[4] J. Esteban, W. Simson, S. Requena Witzig, A. Rienmüller, S. Virga, B. Frisch, O. Zettinig, D. Sakara, Y.-M. Ryang, N. Navab, and C. Hennersperger, "Robotic ultrasound-guided facet joint insertion," *International Journal of Computer Assisted Radiology and Surgery*, vol. 13, no. 6, pp. 895–904, Jun. 2018, ISSN: 1861-6429.

[5] C. Hennersperger, B. Fuerst, S. Virga, O. Zettinig, B. Frisch, T. Neff, and N. Navab, "Towards mri-based autonomous robotic us acquisitions: A first feasibility study," *IEEE transactions on medical imaging*, vol. 36, no. 2, pp. 538–548, 2016.

[6] M. Tirindelli, M. Victorova, J. Esteban, S. T. Kim, D. Navarro-Alarcon, Y. P. Zheng, and N. Navab, *Force-ultrasound fusion: Bringing spine robotic-us to the next "level"*, 2020. arXiv: 2002.11404 [eess.IV].

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. arXiv: 1312.5602.

[9] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6284–6291.

[10] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-del-Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247–3254, Oct. 2018, ISSN: 2377-3774.

[11] A. Alansary, O. Oktay, Y. Li, L. Folgoc, B. Hou, G. Vaillant, K. Kamnitsas, A. Vlontzos, B. Glocker, B. Kainz, and D. Rueckert, "Evaluating reinforcement learning agents for anatomical landmark detection," *Medical Image Analysis*, vol. 53, Feb. 2019.

[12] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "A brief survey of deep reinforcement learning," *IEEE Signal Processing Magazine*, vol. 34, Aug. 2017.

[13] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[14] T. Chu, J. Wang, and J. Chen, "An adaptive online learning framework for practical breast cancer diagnosis," in *Medical Imaging 2016: Computer-Aided Diagnosis*, G. D. Tourassi and S. G. A. III, Eds., International Society for Optics and Photonics, vol. 9785, SPIE, 2016, pp. 537–548.

[15] F. Milletari, V. Birodkar, and M. Sofka, "Straight to the point: Reinforcement learning for user guidance in ultrasound," *CoRR*, vol. abs/1903.00586, 2019.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[17] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. arXiv: 1509.06461.

[18] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[20] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2711–2720.

[21] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012, ISBN: 0262018020.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385.

[23] A. Raffin, *Rl baselines zoo*, https://github.com/araffin/rl-baselines-zoo, 2018.

[24] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, *Stable baselines*, https://github.com/hill-a/stable-baselines, 2018.

[25] G. Huang, Z. Liu, K. Weinberger, and L. van der Maaten, "Densely connected convolutional networks. arxiv 2017," *arXiv preprint arXiv:1608.06993*,

Authorized licensed use limited to: Purdue University. Downloaded on September 22,2022 at 15:20:23 UTC from IEEE Xplore. Restrictions apply.