Automatic Catheter Navigation through Deep Reinforcement Learning

ENJALBERT Robin

09/03/2021

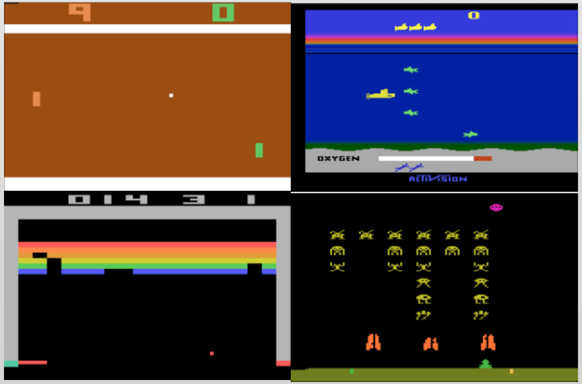
Outline

1) Deep Reinforcement Learning
2) Deep Q-Network
3) Automatic Catheter Navigation

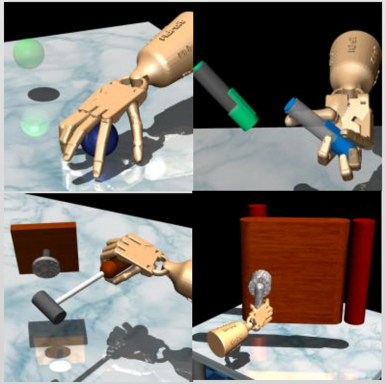
I) Deep Reinforcement Learning

Introduction

- Machine Learning algorithms for sequential decision-making
- Learn optimal decision-making behaviour through experience
- Popularized with its use in Atari video games and the Go game
- Applications : robotics, health, finance, autonomous cars…



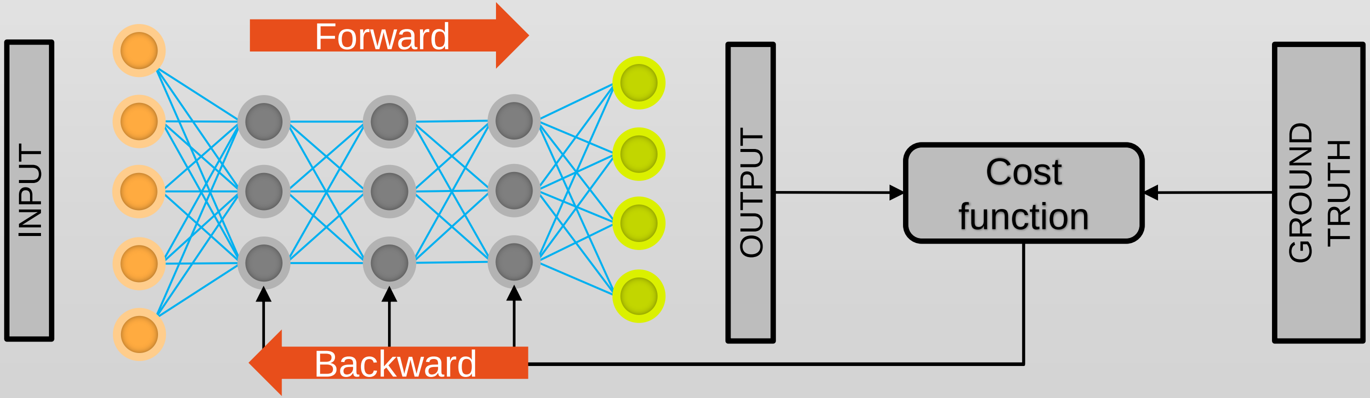
[1]



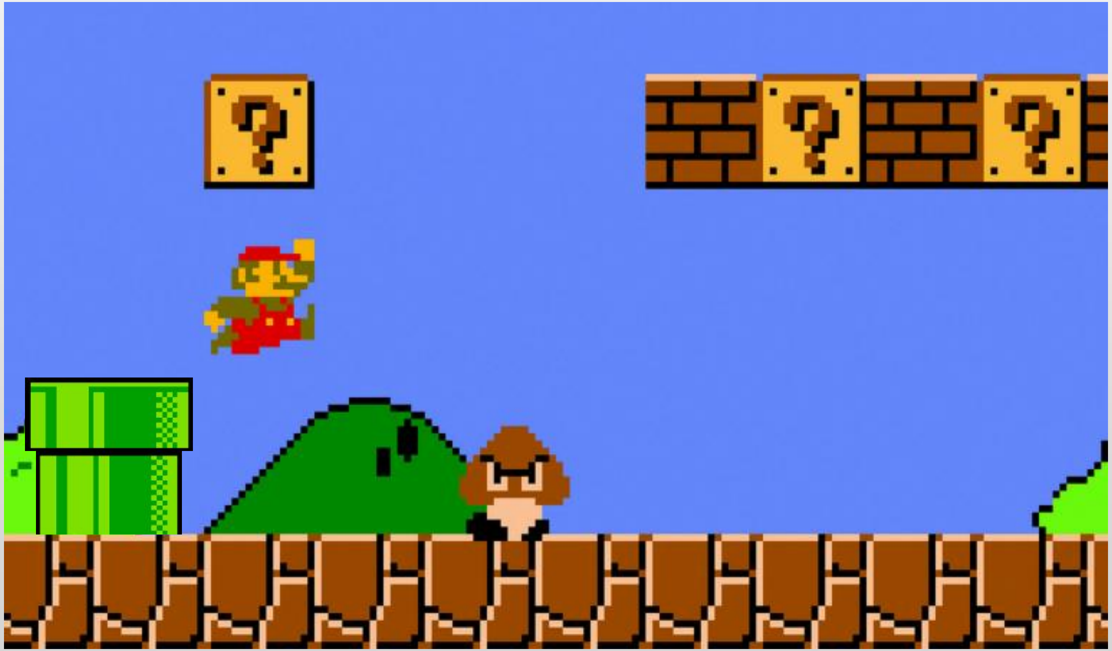
[2]

Deep Learning – Quick Recap

- Optimize the parameters of an Artificial Neural Network
- Forward propagation : inputs go through the layers
- Loss : cost function, compare outputs with the expected values
- Backward propagation : adjust the parameters of the network according to the gradients of the cost function

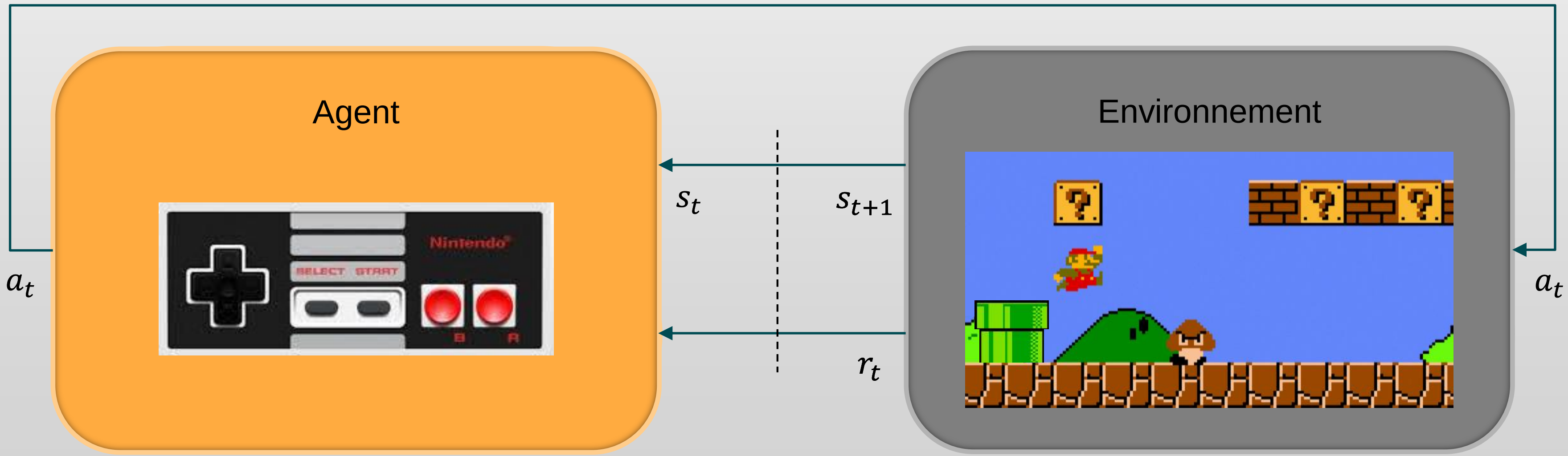


NES video game example



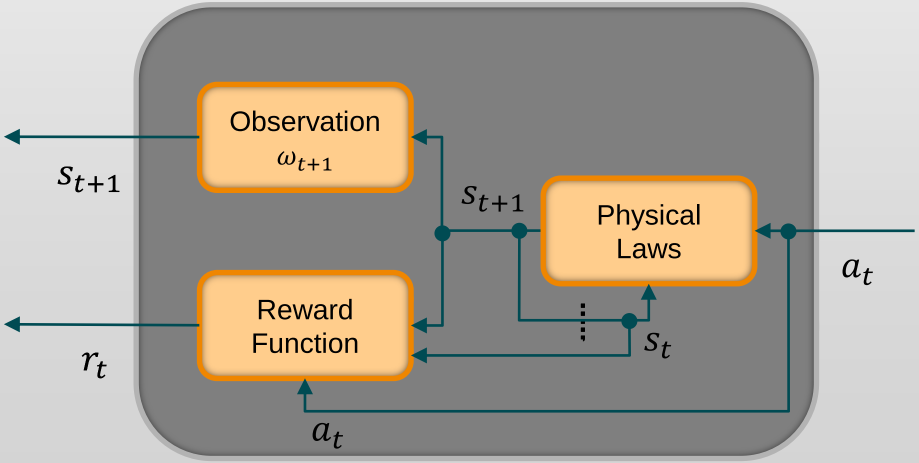
DRL main principle

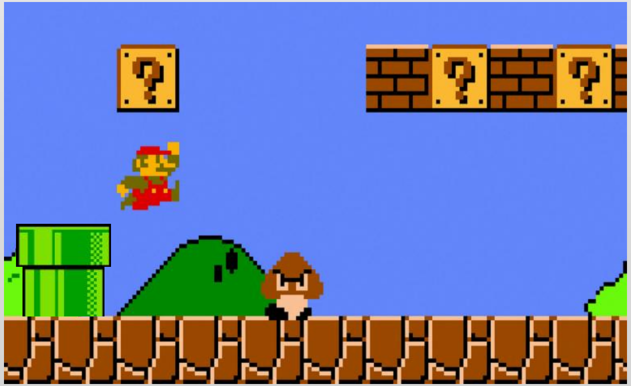
- Agent : choose an action a_t
- Environment : transition from state s_t to s_{t+1}, reward r_t



DRL Environment

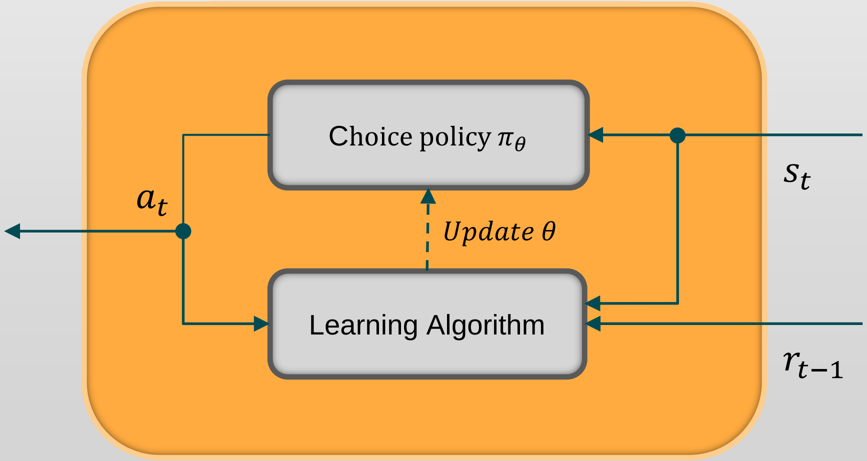
- Physical laws to compute transition from state s_t to s_{t+1}
- Observation of the current state ω_{t+1}
- Reward function to return r_t





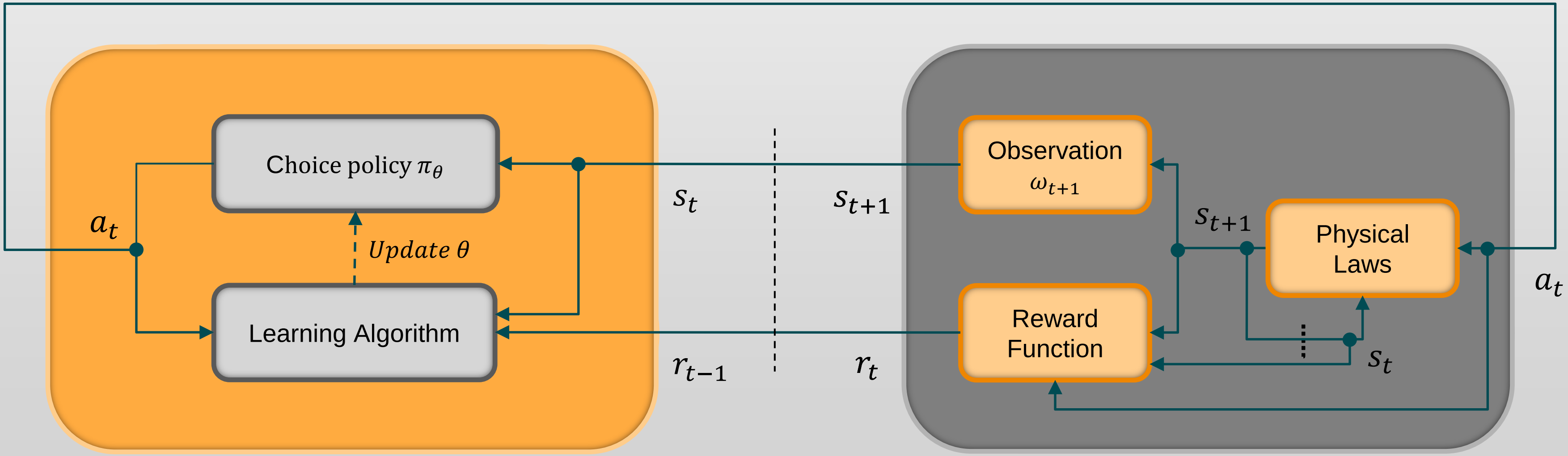
DRL Agent

- Choose action a_t according to s_t to optimize rewards
- Update the policy parameters θ according to gathered experience



DRL Whole Pipeline

- Let the agent evolve in the environment a lot of times
- Gather experience and update the decision-making process to maximize rewards



2) Deep Q-Network

Deep Q-Network main principle

- Define a value function Q to maximize the expected rewards according to a choice policy
- Represent Q with a neural network (DQN) of weights θ
- Optimize weights θ with deep learning process

Decision-making policy

- Reward for a transition: $r_t = r(s_t, a_t, s_{t+1})$
- Cumulative reward: $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$
- Optimal choice policy: $\pi^* = \arg\max_{\pi_\theta} \mathbb{E}[R(\tau) \mid \pi_\theta]$
- Q value function:

$$Q^\pi(s,a) = \mathbb{E}[R(\tau) \mid s_0 = s, a_0 = a, \pi_\theta] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi_\theta]$$

- Q value function (Bellman's equation) :

$$Q^\pi(s,a) = \mathbb{E}[r(s,a,s') + \gamma\, \mathbb{E}[Q^\pi(s', a' = \pi_\theta(s'))]]$$

Decision-making policy

- Q value function (Bellman's equation):

$$Q^{\pi}(s,a) = \mathbb{E}\left[r(s,a,s') + \gamma\, \mathbb{E}\left[Q^{\pi}\left(s', a' = \pi_{\theta}(s')\right)\right]\right]$$

- Optimal Q^* value function:

$$Q^{*}(s,a) = \mathbb{E}\left[r(s,a,s') + \gamma \max_{a'} Q^{*}\left(s', a' = \pi^{*}(s')\right)\right]$$

- Optimal action: $$a^{*}(s) = \arg\max_{a} Q^{*}(s,a)$$

Q-Network

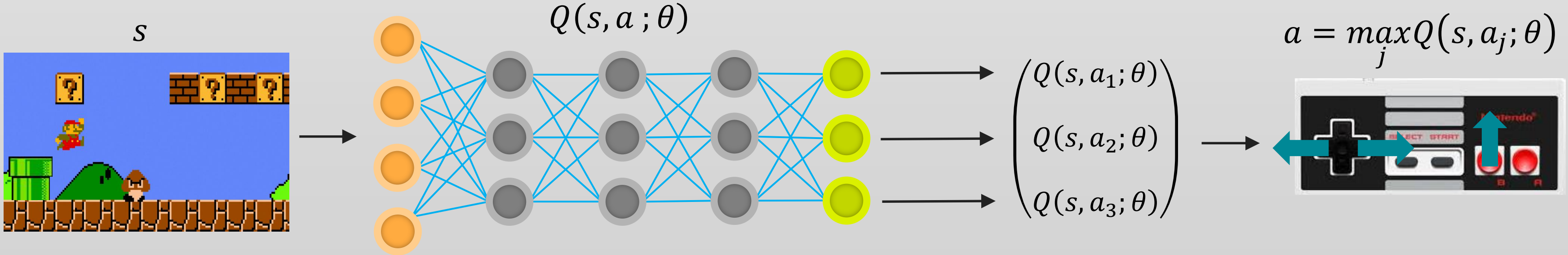
- Iterative optimal Q^* value function:

$$Q_{i+1}(s,a) = \mathbb{E}\left[r(s,a,s') + \gamma \max_{a'} Q_i(s',a')\right] \underset{i\to\infty}{\rightarrow} Q^*$$

- Represent Q with a neural network of weights θ

$$Q(s,a\,;\theta) = \mathbb{E}\left[r(s,a,s') + \gamma \max_{a'} Q(s',a';\theta)\right]$$

- Compute a output value for each possible action and select max



Training process

- Let the agent evolve in the environment a lot of times
- Exploration strategy: less and less random action taken
- Gather experience
- Optimize the θ weights of the Q-Network

Optimize θ weights

- Cost function:

$$L_k = \left(y_k - Q(s,a;\theta_k)\right)^2 = \left(r + \gamma \max_{a'} Q(s',a';\theta_{k-1}) - Q(s,a;\theta_k)\right)^2$$

- Weights θ_k^{-} fixed during C iterations to avoid divergence issues:

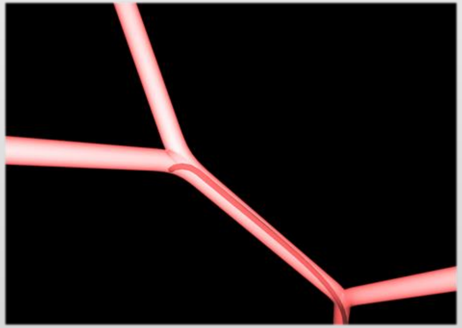
$$L_k = \left(r + \gamma \max_{a'} Q(s',a';\theta_k^{-}) - Q(s,a;\theta_k)\right)^2$$

- Stochastic Gradient Descent along the Q-Network weights (adapt θ according to the gradient of the cost function)

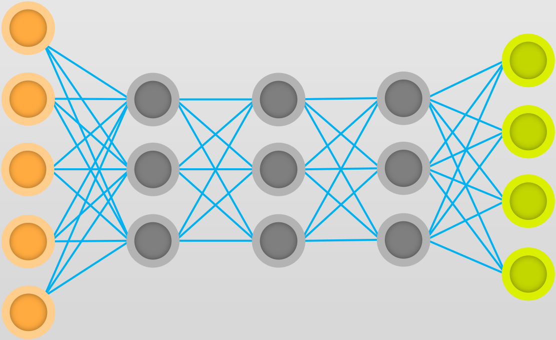
3) Automatic Catheter Navigation

Goal

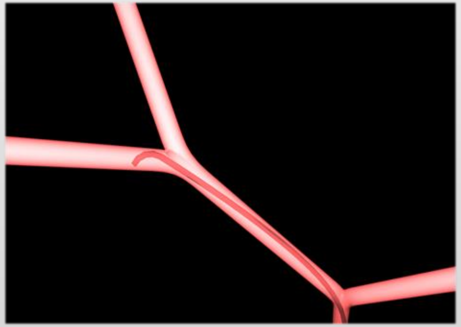
- Choose the appropriate action depending on the target and the catheter state
- Apply Deep Q-Network algorithm on a SOFA simulation



Simulation state



Q value function



Apply chosen action

3) Automatic Catheter Navigation

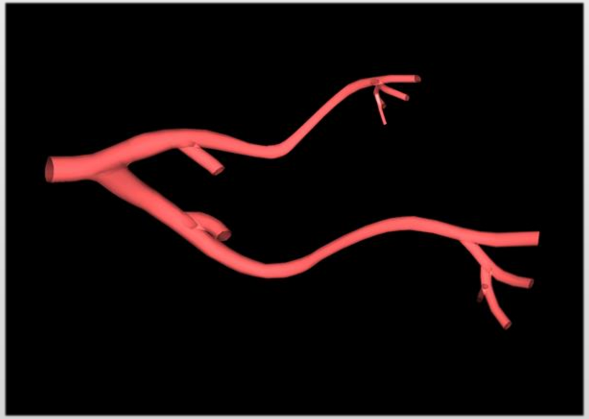
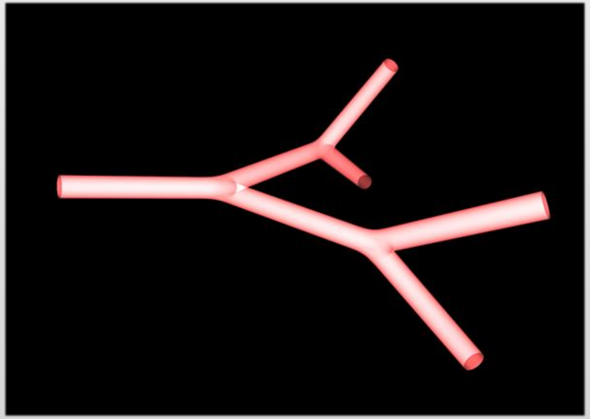
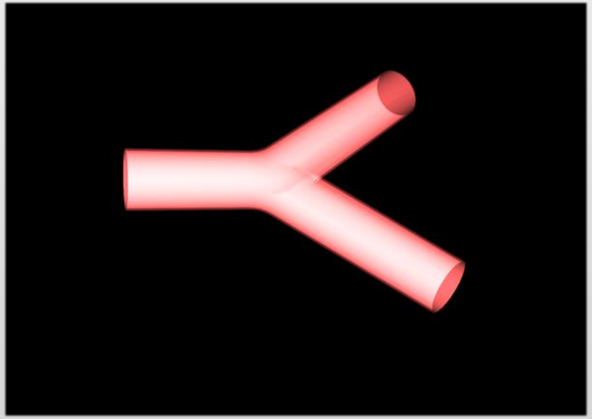
a) **Environment**
b) State, Actions & Reward
c) Results

SOFA Simulation

- Using existing catheter insertion simulation from SofaREBOA plugin
- Mechanical model of the blood vessels
- Mechanical model of the catheter
- Insertion forces and collision model

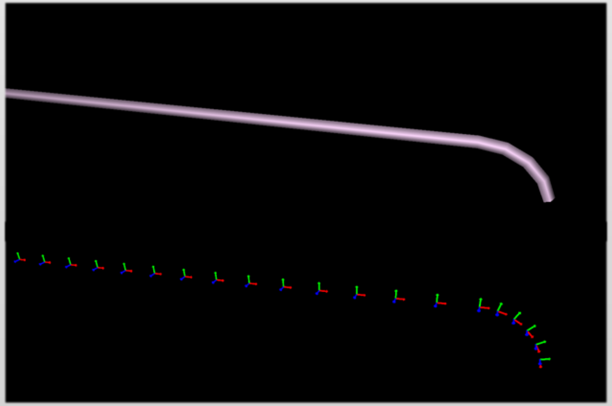
Blood vessels

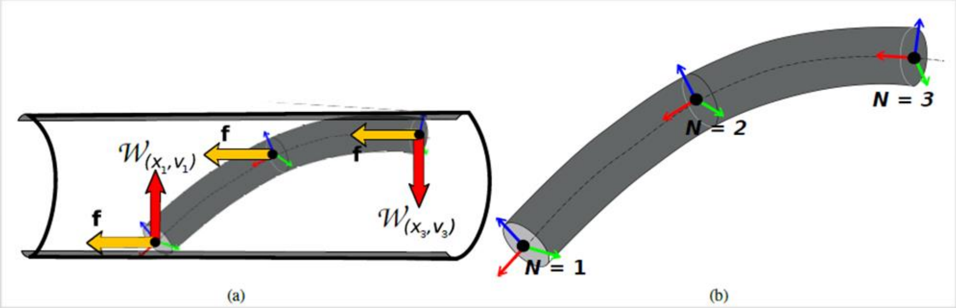
- Walls of blood vessels considered rigid (bones, muscles, organs)
- Geometry progressively simplified in “Y” geometry

Catheter

- Discretised in nodes along the main axis
- Parameterized “J” tip to match the bifurcations in the vascular tree
- Succession of beam element segments
- Insertion force, contact forces

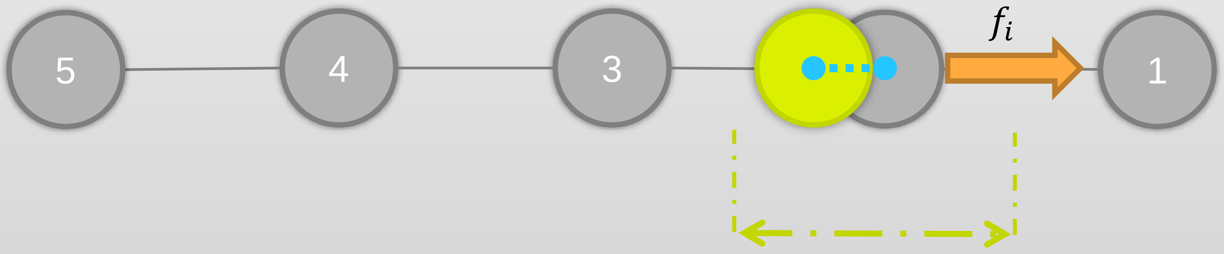




[3]

Simulation control

- Insertion process from Everest plugin
- Insertion point linked to a node of the catheter by a spring
- Control of the DOF of the insertion point (translation, rotation)

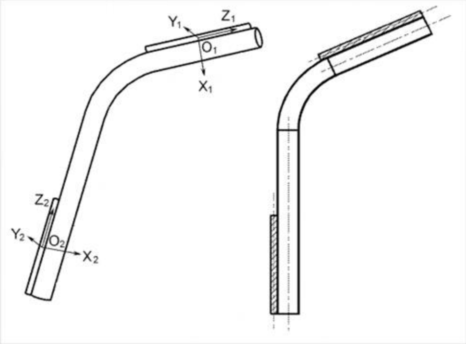


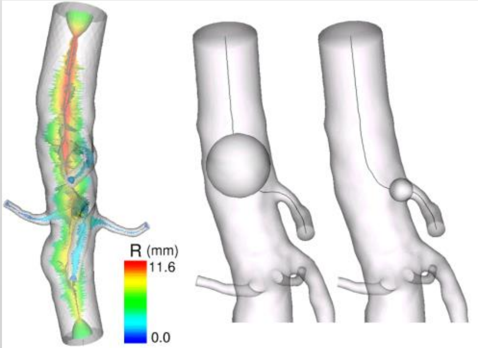
3) Automatic Catheter Navigation

a) Environment
b) **State, Actions & Reward**
c) Results

State

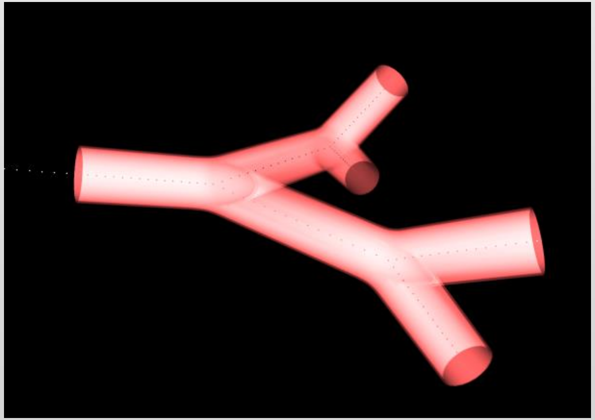
- Relevant information for the navigation : position and orientation of the tip in the vascular tree compared to the target
- Requires position sensors and blood vessels centreline
- Observation given to network = geodesic distance to the target + orientation on a bifurcation

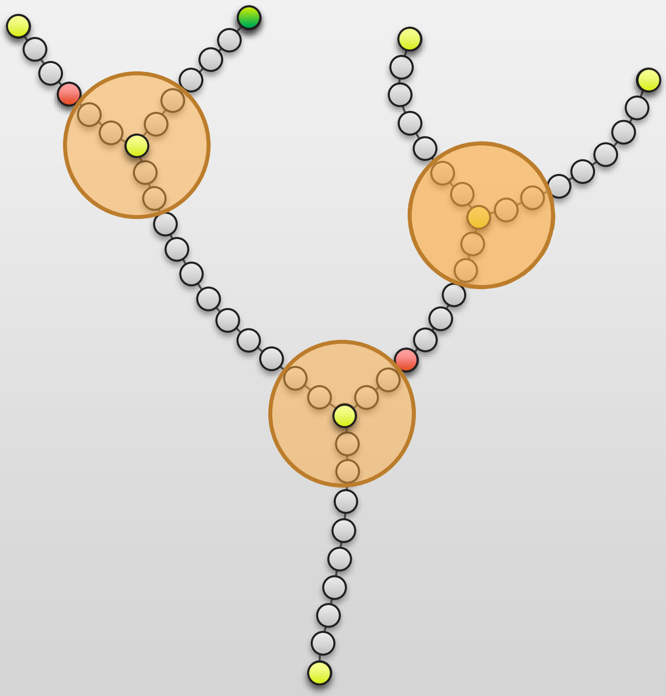




State – Blood vessel centreline

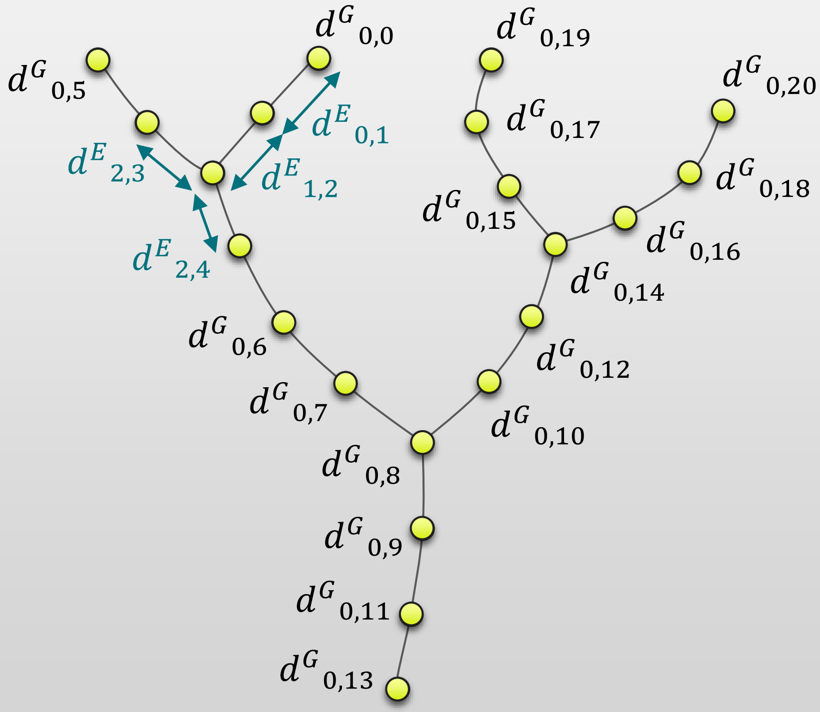
- Center of circular sections of vessels
- Extracts tips and bifurcation
- Define an initial state, a random target and final states





State – Geodesic distance

- Geodesic distance graph
- Interpolation of the catheter tip at the closest centreline node
- Geodesic distance to the target along the centreline $d^G{}_{0,cath}$



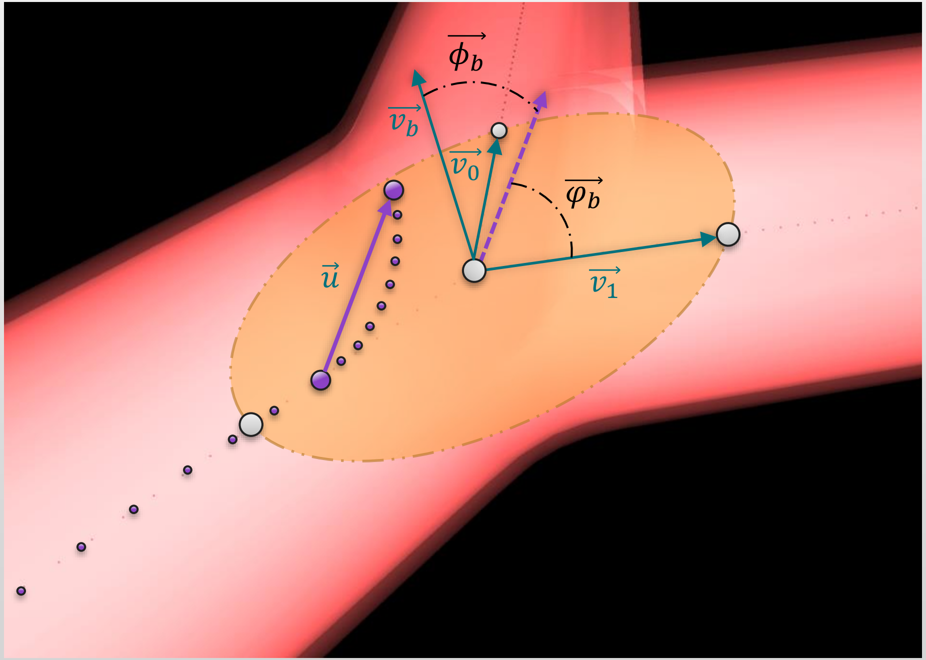
State – Orientation on bifurcations

- "J" tip $\vec{u}$ orientation
- Vessels $(\overrightarrow{v_0}, \overrightarrow{v_1})$ orientations
- Normal vector $\overrightarrow{v_b} = \overrightarrow{v_0} \wedge \overrightarrow{v_1}$

$$\cos(\varphi_{bifurcation}) = \frac{\vec{u} \cdot \overrightarrow{v_1}}{\|\vec{u}\| \|\overrightarrow{v_1}\|}$$

$$\cos(\phi_{bifurcation}) = \frac{\vec{u} \cdot \overrightarrow{v_b}}{\|\vec{u}\| \|\overrightarrow{v_b}\|}$$

$$s_t = \left(d^{G}{}_{0,cath}, \cos(\varphi_b), \cos(\phi_b)\right)$$

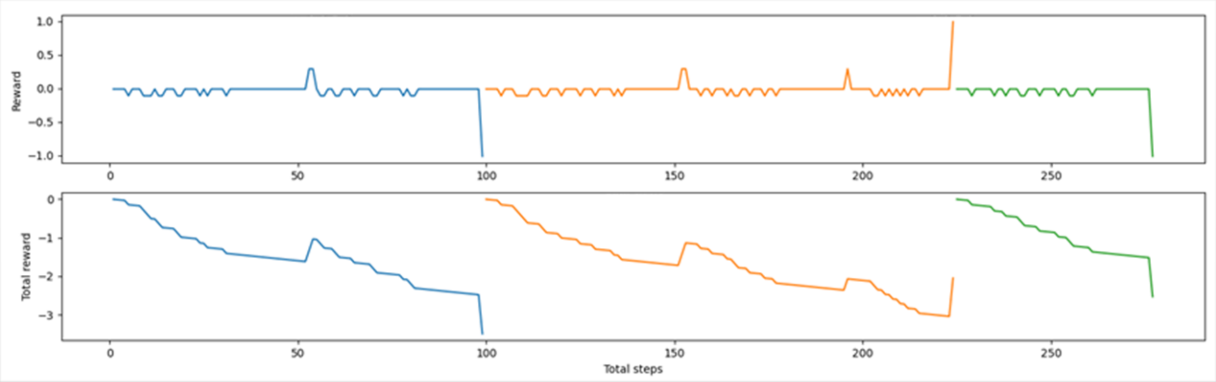


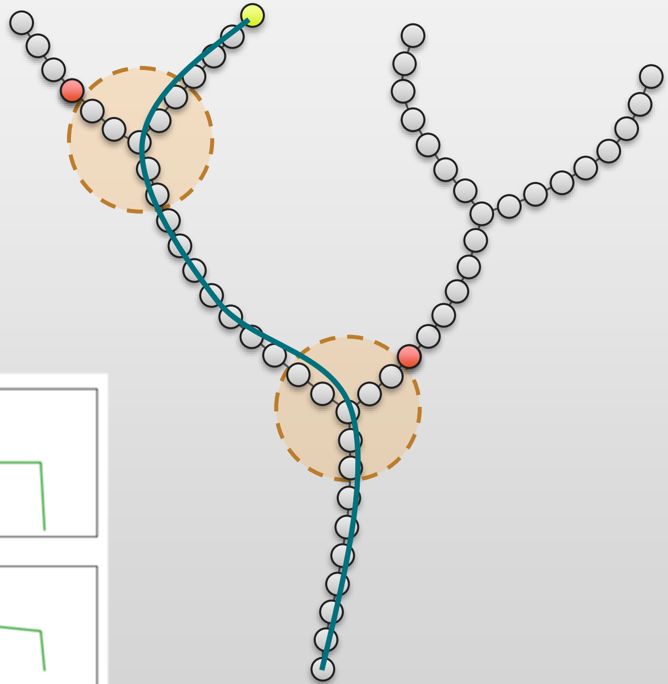
Actions

- Forward translation
- Right and left rotations
- Compute several simulation steps with translations between two actions to amplify the impact of a choice

Rewards

- Success final state: $r = +1$
- Failure final state: $r = -1$
- Good bifurcation: $r = +0.3$
- Rotation outside of bifurcations: $r = -0.1$
- Simulation step: $r = -0.01$



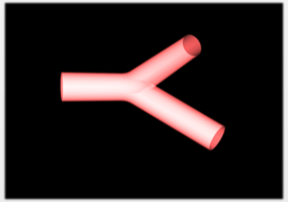


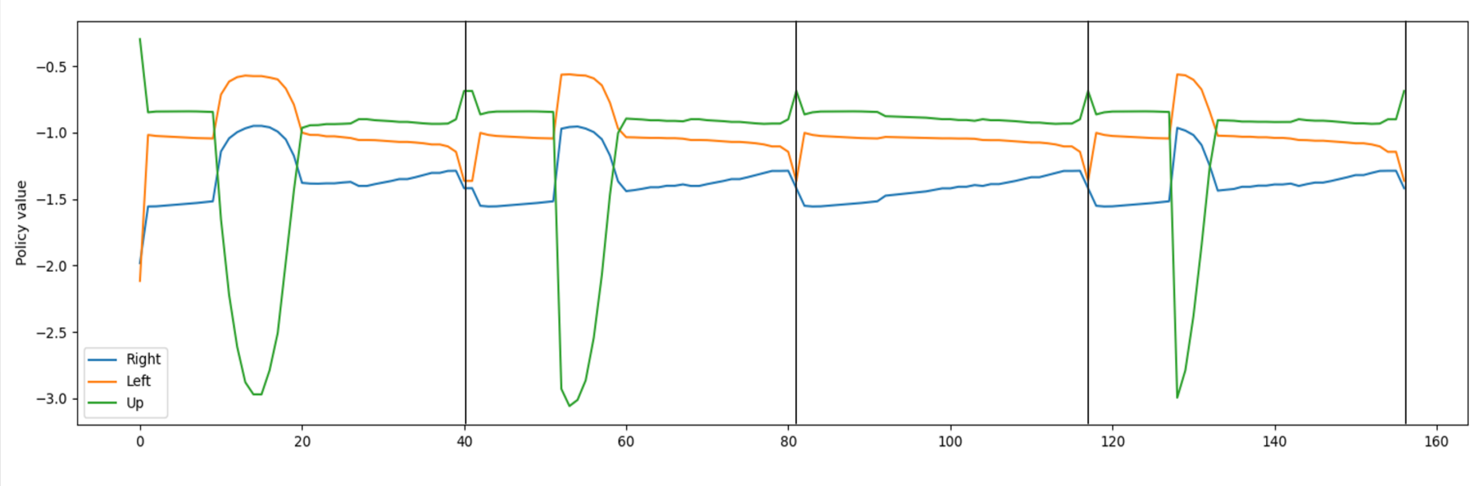
3) Automatic Catheter Navigation

a) Environment
b) State, Actions & Reward
c) Results

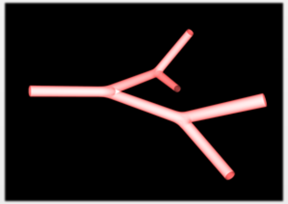
Training results on “single Y”

- Learning on 500 simulations: 100% success rate

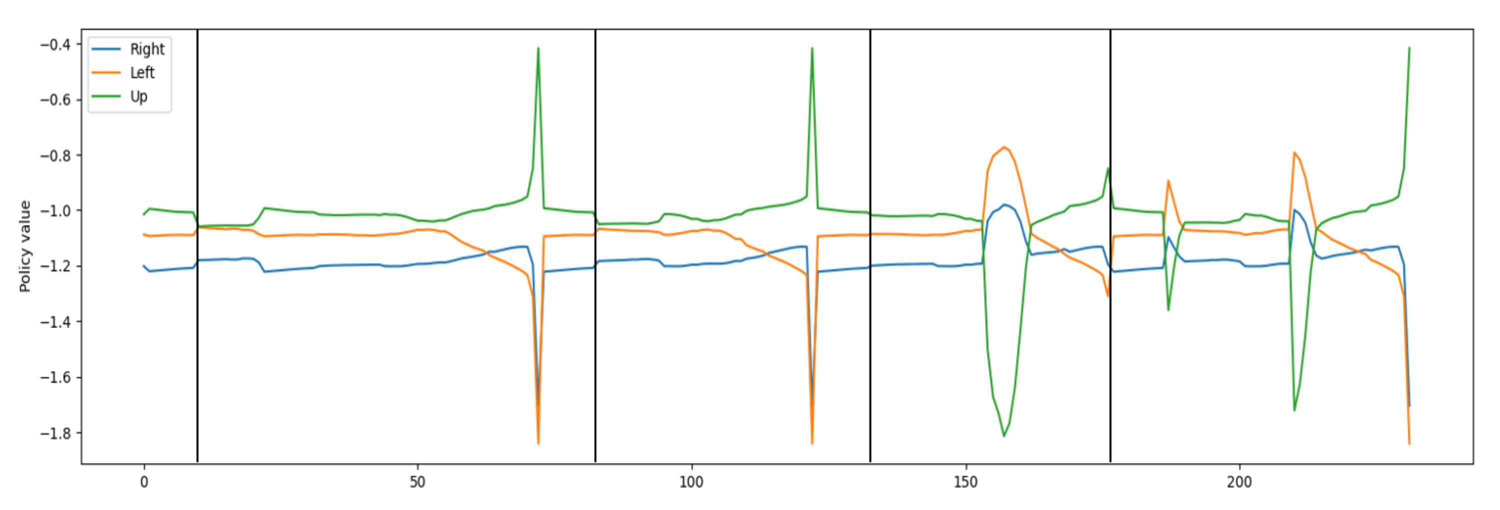




Training results on “double Y”

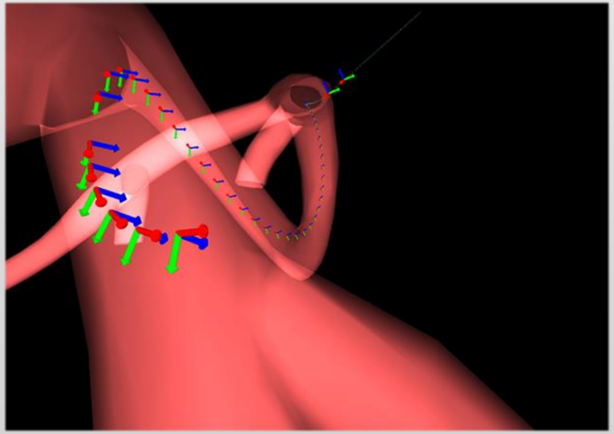


- Learning on 500 simulations: 91% success rate



Perspectives

- Improve the choice of rotation at bifurcations
- Allow the backward translation for technical movements
- Work on complex vascular tree anatomy: manage the non-linear response when the catheter is inserted in small vessels



1. V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning”, December 2013.
2. A. Rajeswaran *et al.*, “Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations”, *Robotics: Science and Systems*, 2018.
3. R. Trivisonne *et al.*, “Constrained Stochastic State Estimation of Deformable 1D Objects: Application to Single-view 3D Reconstruction of Catheters with Radio-opaque Markers”, *Computerized Medical Imaging and Graphics*, vol. 81, April 2020.

Thank you