Learning Open-Loop Saccadic Control of a Biomimetic Eye using the Soft Actor-Critic Algorithm

Henrique Granado
henrique.granado@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

December 2021

Abstract

The application of reinforcement learning algorithms to robotics has been increasing over the last decades, specially in the implementation of non-trivial tasks where the control of robots with a high number of degrees of freedom is very difficult using classic control techniques. The human body can be seen a system composed of very complex subsystems, and the ORIENT project study and learn the control of a robotic version of one of these subsystems – the human eye. Using the soft actor-critic algorithm, this work aims to link reinforcement learning to this control problem, and create a framework that will learn the open-loop control of the movement of the eye. The base for the type of control implemented is inspired on the human eye control signal, where a pulse signal is generated, integrated and sent through the nervous system to the muscles required to perform the wanted movements (saccades). The metric that evaluates the saccades produced is also inspired by the human system and is used to lead the learning algorithm to the desired results. This methodology was applied on a very simplified version of the human eye as a proof of concept. The algorithm managed to learn the optimal saccadic control strategy, for three different versions of pulse functions. The trajectories obtained, have non-linear properties similar to the ones registered in humans.
Keywords: Soft Actor-Critic, Open-Loop Control, Biomimetic Eye, Saccades

1. Introduction

Over the past decades, the speed of technological developments has increased tremendously. Part of these developments aim at alleviating everyone's daily burdens in life, for example, through designing machines, including robotic systems, which may eventually automate our daily routines. One of the major conceptual advances that is guiding these developments is the statistical technique called "machine learning", which is defined "as the study of statistical computer algorithms that improve automatically through experience" [14], and which manage to perform tasks that would otherwise be extremely difficult, if not impossible, to be carried out by humans. At the same time, these developments run into the risk to causing distress to people who fear to lose control over their lives, or lose human connection. It is with this in mind that the development of humanoid robots may be thought to better connect this complex technology with humans. Humanoid robots could be employed to perform tasks in ways that resemble humans. However, the human body is a highly complex system that is shaped by an evolutionary process during many millions of years. Mimicking such systems in a humanoid robotic system with traditional control techniques is therefore not a trivial task. One example of a complex subsystem of the human body is the eye. Within the ORIENT project (a EU Horizon 2020 ERC advanced grant) our group learns and studies how to control a humanoid robotic version of the human eye through the application of different control techniques [10, 13, 19, 4, 6]. So far, the group has worked on different robotic models, and used classical control techniques to generate the desired movements of the robotic eye. This thesis follows up on previous work done under the ORIENT project, and aims to develop a framework with a new methodology to study how the eye moves. Unlike the other works, this will follow a machine-learning approach for the control of the eye, where the algorithm will be put in situations where a certain movement is desired and by trial and error the algorithm will learn how to perform desired motions. Due to lack of time this will only be done for one dimension of the eye orientation, on a simplified model.

2. Background

2.1. Saccades

The two main types of conjugate eye movements of the human oculomotor system are smooth pursuit, where the retinal slip [3] is kept to a minimum in order to track and maintain moving objects on the fovea, and saccades, which are described by a quick change of the line of sight to allow for a quick re-orientation of the fixation point. Since the brain cannot perceive clear images during fast motions [5], it is imperative to keep the duration of saccadic movements to a minimum, allowing for swift scans of the environment.

2.2. Neural Control

The model of the eye-plant (extra-ocular muscles, the eye ball, and surrounding tissues) has been approximated by a second-order linear over-damped system with a long time constant of about $T_1 \sim$ 200 ms. The fatty tissues around the eye are the major cause of this slow over damped property, far exceeding the average saccade duration raging between 20 to 100 ms [15]. As a consequence, the brain has to employ a non-linear pulse-step control strategy to overcome this overdamped system for fast and accurate saccades. David A. Robinson [16] proposed a model (Figure 1) for the control of the eye which has a solid conceptual framework to study and understand the eye control system. Based on the dynamic error between the desired , s_0, and the current eye orientation, $e(t)$, a high-frequency pulse signal, $\dot{e}$, is sent to the motor neurons to make the saccadic movement, and which overcomes the eye's frictional forces, e. In parallel, this pulse signal is neurally integrated to let the motor neurons produce the required elastic force that keeps the eye at the desired orientation. So, the saccadic movement of the eye is caused by a pulse-step control signal, where the appropriate pulse is programmed by the brain, which is also neurally integrated.

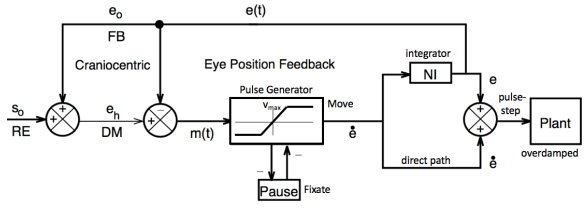


Figure 1: Robinson's model of the saccadic system [16]. s_0 represents the position of the target in the retina, e_h is the head-centered desired position, e_0 is the initial position of the eye, $e(t)$ is the current eye position, $m(t)$ represents the motor error and $\dot{e}$ is the pulse signal.

2.3. Non-linear Dynamics

A saccade "main sequence" is a term that describes relationships between duration, peak velocity and amplitude. Skewness is another important property of saccades describes the ratio between the time of deceleration of a saccade and its duration. These properties have been proven to be an important tool to investigate the motion of the eye, both for, the design and test of various models of saccadic control and for diagnosing the integrity of the human saccadic system [1].

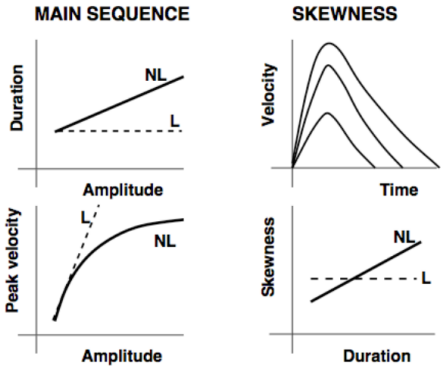


Figure 2: Non-linear properties of saccades [15]

In Figure 2, it's illustrated the stereotyped main sequence properties. The non-linear dynamics of the saccades can be seen in all four 4 graphs. The relationship between amplitude and duration in a linear system would result in constant saccadic duration, thus the increase in duration with amplitude represents the non-linear behaviour. Similarly for larger amplitudes, larger peak velocities would have to be reached to achieve constant saccade durations, however in the non-linear system this peak velocity saturates. Skewness increases with duration so that the velocity profiles are not scaled version of each other, independently of amplitude. These dynamics will be key in the work to understand and evaluate whether or not the saccadic control is approximates to that of the human eye.

2.4. Reinforcement Learning Framework

Reinforcement learning is a field of artificial intelligence that aims to solve problems through a system mechanism of punishment and reward. Generally speaking there will be an entity that is aimed to train, the agent, that interacts with an environment; the agent retrieves a state from the environment and applies an action to it, then after a state update it receives a reward. The agent knows nothing about the environment but has the goal of maximizing the reward obtained. Depending on the application there are various ways that the agent can make decisions. The agent could use a value-based algorithm, where the agent tries to predict the rewards obtained from applying an action to a state - from a set of actions the agent would apply the one that outputs the highest predicted reward. The

agent could use a model-based approach, where it creates a simulation of the environment, predicting the follow up state from applying an action to the environment - the action applied would be obtained from simulated trajectories. Lastly, the agent could adopt a policy based learning methodology where it immediately maps a state to an action - it generates a sequence of actions that lead the agent to the final objective. The algorithm that is going to be used in this project, the soft actor-critic, will have both a policy approach and a value approach; as it does not aim to know anything about the environment this will be a model-free based approach. Before detailing the structure of the algorithm, there is the need to introduce some key concepts that will help give insight to how the algorithm works.

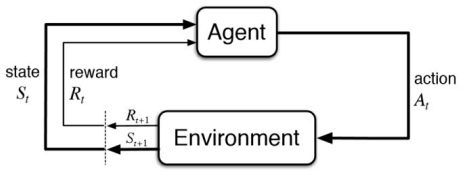


Figure 3: Diagram portraying the general idea behind reinforcement learning algorithms

2.4.1 Markov Decision Process

A Markov decision process is the mathematical framework that will serve as a basis for the learning algorithm. This is going to be defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ and $\mathcal{A}$ correspond to a continuous state space and continuous action space respectively, p is the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and the current action $a_t \in \mathcal{A}$, and r is the reward issued by the environment on each transition. Using this notation it is now possible to define the reinforcement learning problem: at each time step, $t = 0, 1, ...$, the agent will receive a representation of the state of the environment $s_t \in \mathcal{S}$ and apply an action $a_t \in \mathcal{A}$. The environment will then provide a reward r from taking the action a_t at a state s_t, $r(s_t, a_t)$. As the objective, the agent wants to maximize the cumulative reward from taking each action at each state. Then it will have to learn a policy π^* that maximizes the sum of expected rewards:

$$\pi^* = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t)] \quad (1)$$

where $\pi(a_t|s_t)$ is a probability distribution that represents the probability of taking the action a_t at a state s_t, and $\rho_\pi(s_t, a_t)$ denotes the state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$:

$$\rho_\pi(s_t, a_t) = p(s_0) \prod_{t=0} \pi(a_t|s_t)p(s_{t+1}|s_t, a_t) \quad (2)$$

or in other words, the probability of (s_t, a_t) independently of the succession of states and actions (trajectory) taken to get there [9]. Equation (1) defines the standard objective of reinforcement learning.

2.4.2 Value function and Q-function

The value function is something that is used on the value based approach; it will evaluate the benefit of being in a certain state given a policy π [9]. Since the objective is to maximize the cumulative sum of the expected reward, there is a need to represent this:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{j=0}^{\infty} \gamma^j r_{t+j+1} \quad (3)$$

where r_{t+1} represents the reward obtained at time $t+1$ and γ represents a discount factor which is in the expression to account for future uncertainty. As such G_t is the total discounted reward for time-step t, and so as the value function of a state s_t through the use of a policy π, $V_\pi(s_t)$ is the expected sum of future rewards for state s_t:

$$V_\pi(s_t) = \mathbb{E}_\pi[G_t|s_t] = \mathbb{E}_\pi[\sum_{j=0}^{\infty} \gamma^j r_{t+j+1}|s_t] \quad (4)$$

Similar to the value function, the Q-function, $Q_\pi(s_t, a_t)$, is defined as the expected reward obtained from applying the action a_t to a state s_t, through a policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|s_t, a_t] = \mathbb{E}_\pi[\sum_{j=0}^{\infty} \gamma^j r_{t+j+1}|s_t, a_t] \quad (5)$$

2.4.3 Bellman Equation

The Bellman equation is a key element in reinforcement. Its use simplifies the computation of the value function, by breaking down the problem into simpler recursive sub-problems [20]. As such, the expressions for Value function can be rewritten as:

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}|s_t] = \\ &\mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})|s_t] \\ &= r(\cdot|s_t) + \gamma \sum_{s_{t+1}} \rho_\pi(s_{t+1}|s_t) V_\pi(s_{t+1}) \end{aligned} \quad (6)$$

As the reward of r_{t+1} is simply the the reward at state s_t, this is expressed as $r(\cdot|s_t)$ and the sum is just the definition of expected value, where $\rho_\pi(s_{t+1}|s_t)$ is the probability of being at a given state s_{t+1} from a state s_t through a policy π.
Similarly the Q-function can be rewritten:

$$\begin{aligned} Q_\pi(s_t, a_t) &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}|s_t, a_t] = \\ &\mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t, a_t] \end{aligned} \quad (7)$$

As the sum of the probabilities of taking all the possible actions of a policy π at state s_t equal to 1, $\sum_{a_t} \pi(a_t|s_t) = 1$, the value function can be rewritten in terms of the Q function and the policy π:

$$V_\pi(s_t) = \sum_{a_t} \pi(a_t|s_t) Q_\pi(s_t, a_t) \tag{8}$$

Similarly, it is possible to rewrite the Q-function in terms of the value function:

$$Q_\pi(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1}} \rho_\pi(s_{t+1}|s_t, a_t) V_\pi(s_{t+1}) \tag{9}$$

By substituting equation 8 into 9:

$$Q_\pi(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t) \sum_{a_t} \pi(a_{t+1}|s_{t+1}) Q_\pi(s_{t+1}, a_{t+1}) \tag{10}$$

And vice-versa:

$$V_\pi(s_t) = \sum_{a_t} \pi(a_t|s_t) \left(r(s_t, a_t) + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t) V_\pi(s_{t+1})\right) \tag{11}$$

And so there are now two recursive expressions to obtain the value for the Q-function and value function.

2.4.4 Entropy

Entropy is defined as 'lack order or predictability' [18], and this idea will be present in the soft actor critic algorithm. For example, winning the coin game of heads or tails is much less predictable than winning the lottery. Since the probability of the first one is 50% the outcome is uncertain, whereas on the latter case the outcome of a loss is mostly certain. So, the game of heads or tails has higher entropy than the lottery. Formally speaking the entropy $\mathcal{H}(X)$ is defined as:

$$\mathcal{H}(X) = \mathbb{E}\left[-\log(p_d(x))\right] = -\int_x p_d(x) \log(p_d(x)) dx \tag{12}$$

where $p_d(X)$ is the probability distribution of a random variable X, and the base of the logarithm can vary from application to application; here it will be kept as the standard natural logarithm for consistency reasons. This concepts is used in reinforcement learning to promote exploration on the agents, making their actions less predictable, which is usually useful to avoid the agents getting stuck in suboptimal solutions.

2.5. Soft Actor Critic

This section is critical to understand the algorithm that is going to be used in chapter 4 to train the model of the eye. The explanation of this algorithm will be based on [7, 8], where it was firstly introduced and developed. This section will give a brief overview of its properties and also provide the key equations necessary for its understanding and implementation.

2.5.1 Overview

The soft actor-critic algorithm is an efficient design for a continuous state and action space using the maximum entropy framework which has been proven to be stable and robust [21]. The addition of this framework aims to reward exploration besides just learning the best policy, and although the objective is altogether modified, a temperature parameter, α, is used to regularize this entropy factor, ultimately making it possible to recover the original objective.

All-in-all, the soft actor-critic is an algorithm that maximizes reward and exploration, making use of a replay buffer, from a off-policy formulation, to improve sample efficiency and a maximum entropy to enhance stability and promote exploration. The agent will start with a random policy that will use it to apply a random actions to the environment and retrieve information about it. Once it has enough samples, it will be able to distinguish better actions from worse. Then it will improve itself in a way that it will start taking better and better actions as time goes on, until a point where it is always applying the best actions to environment.

2.5.2 Framework

Putting together the standard objective of reinforcement learning (1) and the maximum entropy objective [21]:

$$\pi^* = \arg\max_\pi \sum_t \left(\mathbb{E}_{(s_t, a_t) \sim \rho_\pi}[r(s_t, a_t)] + \alpha \mathcal{H}(\pi(\cdot|s_t))\right) \tag{13}$$

where α is the temperature parameter mentioned above, which will control stochasticity of the optimal policy. As α tends to 0, the traditional objective (1) is recuperated. It is important to mention that the soft actor-critic models actions for a continuous space, so the policy will output actions that have some kind of continuous distribution. On our case this action will be modelled by a gaussian distribution. So, the policy will output a mean and a standard deviation, and a random sample of these values would be the action applied to the environment.

With this all in mind, the characteristics and

implementation of the algorithm can be introduced. To model the soft value function, $V_\pi(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t,a_t) - \alpha\log(\pi(s_t|a_t))]$, the soft Q-function ($Q(s_t,a_t) = r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim\rho_\pi}[V(s_{t+1})]$) and the policy, there will be used 3 neural networks. The soft Q-functions will be parameterized by θ and specified by $Q_\theta(s_t,a_t)$ and the policy networks will be parameterized by ϕ, and designated by $\pi_\phi(a_t,|s_t)$. Also, these two neural networks can be identified as the critic and the actor respectively. The value function V will be parametrized by $\overline{\theta}$.
The soft Q-function parameters can be trained by minimizing the soft Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}[\frac{1}{2}(Q_\theta(s_t,a_t)- (r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\theta}}(s_{t+1})]))^2] \quad (14)$$

where $\mathcal{D}$ represents the replay pool where the values of the state (s_t), the action applied to the state (a_t), the reward obtained $r(s_t,a_t)$, and following state (s_{t+1}) are stored. As such this function can be optimized through stochastic gradient descent.
For the update of the parameters of the value function, $\overline{\theta}$, it is done through an exponentiated moving average of the parameter θ.
The policy parameters are learned through the minimization of the Kullback-Leibler divergence, an operator that measures the the difference of two probability distributions [12, 11]. The new policy will be updated towards the exponential of the new new soft Q-function since this will guarantee an improvement in the soft policy in terms of the soft value [8], leading to a soft bellman residual of π of:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}[\mathbb{E}_{a_t\sim\pi_\phi}[\alpha\log(\pi_\phi(a_t|s_t))-Q_\theta(s_t,a_t)]] \quad (15)$$

To update the parameters ϕ a trick is used, which results in a lower variance estimator. The value of the policy neural network is reparameterized using the transformation $a_t = f_\phi(\epsilon_t;s_t)$ where ϵ_t is an input noise vector from some fixed distribution. From here equation 15 can be rewritten as:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}[\alpha\log(\pi_\phi(f_\phi(\epsilon_t;s_t)|s_t))- Q_\theta(s_t,f_\phi(\epsilon_t;s_t))] \quad (16)$$

which can also be optimized through stochastic gradient descent. Lastly the term that needs to be trained is α. Again this parameter affects how much the algorithm will explore, and for some tasks it is better to explore than others; not only that, even within tasks, it is better to explore in certain situations than others - so keeping this value fixed can be ill-advised and non-trivial task to regulate manually.Instead, this parameter will be automatically adjusted by the algorithm, and the problem is formulated such that the average entropy of the policy is constrained. The residual for the update of this parameter is derived in [8]:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi_t}[-\alpha\log(\pi_t(a_t|s_t)) - \alpha\overline{\mathcal{H}}] \quad (17)$$

where π_t is the policy applied at time t. This result was derived to obtain higher value in regions where the optimal policy is uncertain, promoting exploration, and lower value when the algorithm is able to make distinction between good and bad solutions.

3. Model

We have developed a simulator using physics equations to simulate the movement of the eye. Actuation is provided by three motors with rods, which would have elastics attached to their tips, connected to a spherical joint, simulating an eye, that rotates freely. Depending on the differential in tension between the elastics, due to the motor position, this induces a torque on the eye.

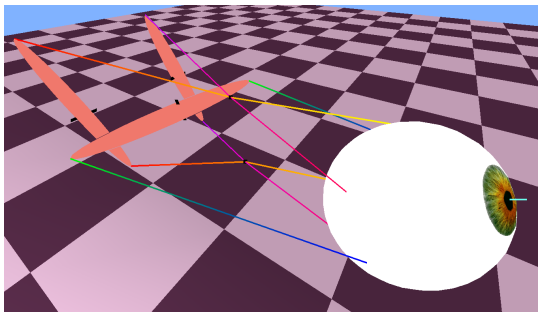


Figure 4: Visual representation of the simulator built using Pyglet

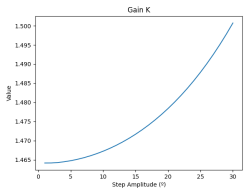
However, for a better understanding of the behaviour of the learning mechanisms, we started with the study of only horizontal saccades (1D) and a linear first order approximation to the eye dynamic model.

3.1. Low Pass Filter

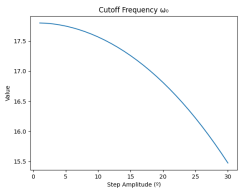
The model of the eye will then be approached by a Low-Pass Filter (LPF), $H(s)$,with a cutoff frequency of ω_0 and a gain, K:

$$H(s) = K\frac{\omega_0}{s+\omega_0} \quad (18)$$

These 2 parameters can be obtained by doing a system identification to the simulator, applying step commands to the motor with different amplitudes and using a Least Square Estimator, it was possible to determine K and ω_0 with respect amplitude (see Figure 5).



(a) Value of K vs. Step Amplitude



(b) Value of ω_0 vs. Step Amplitude

Figure 5: The values ob of K (left) and ω_0 (right) as a function of the step amplitude applied as inputs to the motor

As it can be seen in Figure 5, it is possible to recognize the non-linearity of the system in both graphs. If the system was linear the both graphs would present horizontal lines - from the value of K, the final orientation of the eye would be proportional to the steady state position of the motor, and from the value of ω_0, the duration of the movement would be the same independently of the step amplitude.
To select a value of K for the LPF it was decided to simply pick a value that would minimize the least square error from all the values for each step size, giving a value of $K = 1.4758$. The value of ω_0 was picked to be 17.5, representing a maximum motor speed of 10000 $^o\ s^{-1}$, on the simulator - simulations won't be expected to have the motors move fasted than 10000 $^o\ s^{-1}$ and so the plant won't have a value lower than 17.5. In the end though, regardless of what this value is, some strategy will be employed to account for this value of the plant, such that when, there's the need to implement the algorithm in a non-linear model, the non-linearity of the plant will be taken into account - same thing for the value of K. This will be furthered explained in Section 4. In figure 6 there is the step response

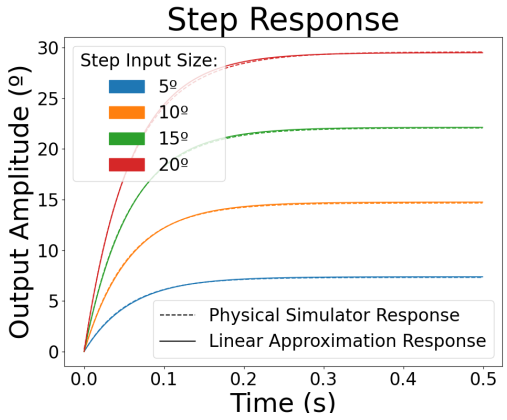


Figure 6: Validation of the first order linear approximation

comparison between the physical simulator and its first order linear approximation. As it can be seen, the approximation can be considered valid since for the same sized inputs both saccades trajectories are very close to each other.

4. Implementation

In this Section it will be given a description of the implementation of the Soft Actor Critic algorithm. Key components will be defined in order to link the algorithm to the model based on the task at hand. These components, state, action, and rewards are stored in a replay pool and are used to train the neural networks.

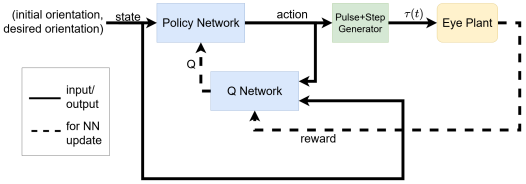


Figure 7: Simplified diagram of the soft actor-critic implementation

4.1. State

Since the control will be done in open loop, the state fed into the policy network will just be the initial eye orientation, and the desired eye orientation. The state vector is represented by a tuple of quaternions parameters, $(q_{i_0},\ q_{i_x},\ q_{i_y},\ q_{i_z},\ q_{d_0},\ q_{d_x},\ q_{d_y},\ q_{d_z})$, where the initial eye orientation quaternion $\mathbf{q_i} = q_{i_0} + (q_{i_x}, q_{i_y}, q_{i_z}) \cdot \mathbf{I}$, where $\mathbf{I}$ is a complex vector of i, j and k, and the final desired orientation is $\mathbf{q_d} = q_{d_0} + (q_{d_x}, q_{d_y}, q_{d_z})$. q_i is a value retrieved from the plant, and q_d is a value generated randomly from episode to episode.

4.2. Action

The action applied to the system is what will change the orientation of the eye. In the real model, this would represent a change in the position of the motor. Since the control will be done in open-loop it is necessary for the output of the policy network to define a movement of the motor from an initial time until infinity. The approach taken is to define a function of time t $\tau(t)$ presenting a trajectory of the motor with a pulse+step shape configured by a set of parameters. This is inspired by the model of saccades established by David A. Robinson [16], (Section 2.2). When a saccade is programmed, a pulse is generated that, with the addition of a neural integrator, will output a pulse-step command that will drive the plant and direct the gaze to the desired position:

$$\tau(s) = p_m(s)(1 + \frac{k_0}{s}) = p_m(s)\frac{s + k0}{s} \qquad (19)$$

The relationship between the motor command τ applied and the pulse generated p_m in the Laplace domain and when such commands are applied to the plant, $H(s)$, it results in the following horizontal angular position of the eye, denoted yaw θ_y:

$$\theta_y(s) = \tau(s)H(s) = p_m(s)K\omega_0\frac{s+k_0}{s(s+\omega_0)} \quad (20)$$

As it can be seen, if $k_0 = \omega_0$, the zero that comes from the integrator and the pole that comes from the plant cancel each other. This will give a saccade with $\theta_y(s) = K\omega_0\frac{p_m(s)}{s}$ which will produce a step-like function based on the pulse properties and the plant static gain - the plant time constant will have no influence. However, pure impulse signals are impossible to generate in practice. We propose three different variants that are going to be used. Figure 8 demonstrates a visual representation of the three pulse variants, corresponding pulse+step functions, and responses with $k_0 = \omega_0$ and $k_0 < \omega_0$. A step response is present as well to show that pulse+step responses are faster. The first pulse variant is:

$$p_m(t) = Ae^{-Bt}t \quad (21)$$

which will produce a saccade with the following function:

$$\theta_y(s) = AK\omega_0\frac{s+k_0}{s(s+B)^2(s+\omega_0)} \quad (22)$$

In Figure 8 this corresponds to the red line, and so,to learn the open-loop control, three parameters must be trained: A, B and k_0. k_0 is the gain of the integrator, B is the decay of the pulse, and A is the amplitude of the pulse. However, this last parameter was found to be unreliable search wise, due to the lack of correlation between it a real world concept, so instead of training it, we train a parameter D that represents the final position of the motor and is related to A using the Final Value Theorem:

$$\lim_{s\to 0} s\tau(s) \Rightarrow D = \frac{Ak_0}{B^2} + F \quad (23)$$

where F is the initial position of the motor. The second proposed pulse variant is:

$$p_m(t) = Ae^{-Bt}(1 - e^{-Ct}) \quad (24)$$

which will produce the saccade (blue line in Figure 8):

$$\theta_y(s) = ACK\omega_0\frac{s+k_0}{s(s+B)(s+B+C)(s+\omega_0)} \quad (25)$$

Now, the final motor position D is given by:

$$\lim_{s\to 0} s\tau(s) \Rightarrow D = \frac{ACk_0}{B(B+C)} + F \quad (26)$$

The new parameter C is set such that the maximum velocity $\mathcal{V}$ of the motor is limited:

$$AC \leq \mathcal{V} \quad (27)$$

obtained through first and second order derivatives of $\tau(t)$. This can be used in a real life scenario where the motor speed cannot go over a certain threshold. This parameter will be fixed and as a consequence the motor will output higher peak velocities for higher saccade amplitudes.
The third variant of the pulse is:

$$p_m(t) = Ae^{-Bt}(1 - e^{-Ct} - Cte^{-Ct}) \quad (28)$$

which will produce a saccade with the following function (green line in Figure 8):

$$\theta_y(s) = AC^2K\omega_0\frac{s+k_0}{s(s+B)(s+B+C)^2(s+\omega_0)} \quad (29)$$

with a value of D:

$$\tau(t) = \lim_{s\to 0} s\tau(s) \Rightarrow D = \frac{AC^2k_0}{B(B+C)^2} + F \quad (30)$$

and C is constrained by:

$$\frac{AC}{e} + \frac{Ak_0C^2}{(B+C)^2} \leq \mathcal{V} \quad (31)$$

4.3. Reward

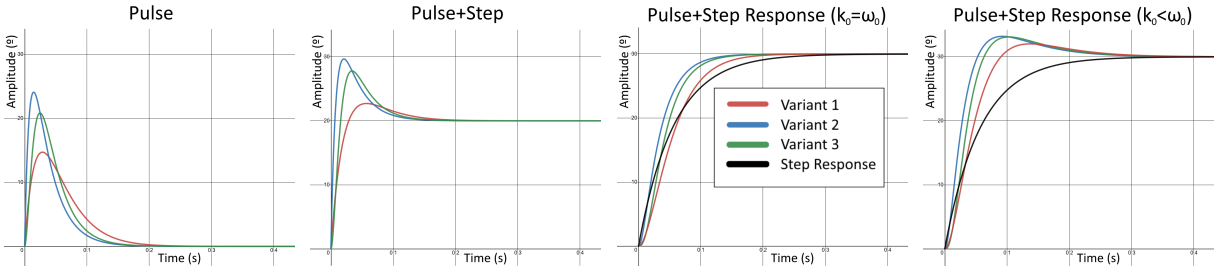
The reward function takes into account costs (penalties) on lack of accuracy, energy spent and saccade duration, as in previous works [10, 13, 19, 4, 6], to which we add overshoot introduced by parameter k_0, which is an undesired and atypical property (Figure 8(d)). In reinforcement learning, it is intended to maximize the value of the reward, and so the negative of these 4 costs will define the reward:

$$R_{total} = \lambda_a R_a + \lambda_e R_e + \lambda_d R_d + \lambda_o R_o \quad (32)$$

where R_{total} corresponds to the total reward, R_a corresponds to the accuracy reward, R_e corresponds to the energy reward, R_d corresponds to the duration reward and finally R_o is the overshoot reward. The λs are weights that control the relevance of each of the rewards - depending on the values of these parameters, the optimal solutions obtained after training are different.
The energy term is the integral of the squared velocity of the motor:

$$R_e = -\int \left(\frac{d\tau}{dt}\right)^2 dt \quad (33)$$

The expression for duration reward will be based on the work of Shadmehr [17], where he found the



(a) Pulse function (b) Pulse+Step function (c) Response with $k_0 = \omega_0$ (d) Response with $k_0 < \omega_0$

Figure 8: Pulse function, pulse+step function, and corresponding responses of the three pulse variants with $k_0 = \omega_0$ and $k_0 < \omega_0$, as well as a pure step response

duration cost of a saccade in humans follows a hyperbolic function:

$$R_d = -\left(1 - \frac{1}{1 + \beta t_d}\right) \tag{34}$$

where t_d corresponds to the duration of a saccade and β is the temporal discount rate. The accuracy cost is the squared difference between the target saccade direction and the the final saccade orientation. In quaternions, the representation of the reward of this expression is [2]:

$$R_a = -(1 - (q_d \cdot q_f)^2) \tag{35}$$

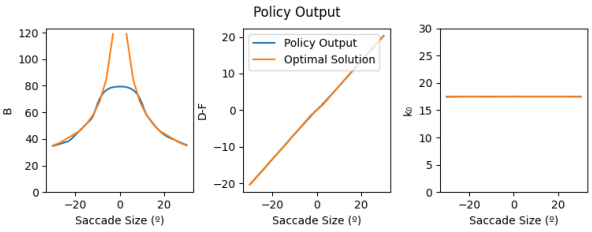
Where q_d corresponds to the desired/ target quaternion orientation, q_f corresponds to the final quaternion orientation of the eye and $\cdot$ correponds to the inner product of the two quaternions.
The overshoot reward penalizes the existence of overshoot, which is undesired in saccade movements:

$$R_o = -\left(\frac{\theta_{peak} - \theta_0}{\theta_f - \theta_0} - 1\right) \tag{36}$$

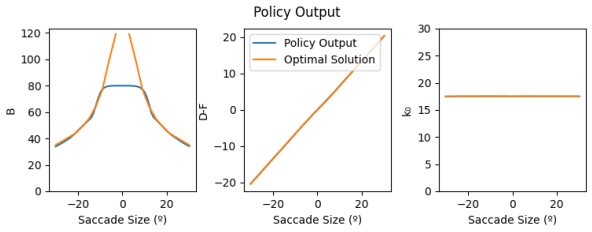
where θ_0 is the initial yaw, θ_f is the a final yaw and θ_{peak} is the maximum value of the yaw for the saccade.

5. Results

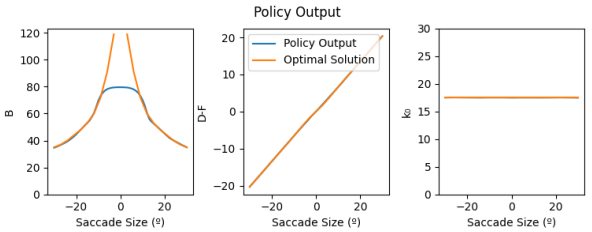
The soft actor-critic algorithm is used to train the policy network to learn the optimal policy to drive the system to the desired orientation. We present in Figure 9 the average output of the policy network - B, k_0 and $D-F$ (the accuracy error) as functions of saccade amplitude, compared to the optimal values computed through exhaustive search, for each defined pulse of the variants. Figure 10 shows the saccades obtained from applying the policy network for a 10, 20 and 30 ^{o} desired amplitude and their equivalent pure step response. Plots about the main sequence properties and skewness are shown in Figure 11 to evaluate the dynamics of the saccades obtained.



(a) Variant 1



(b) Variant 2

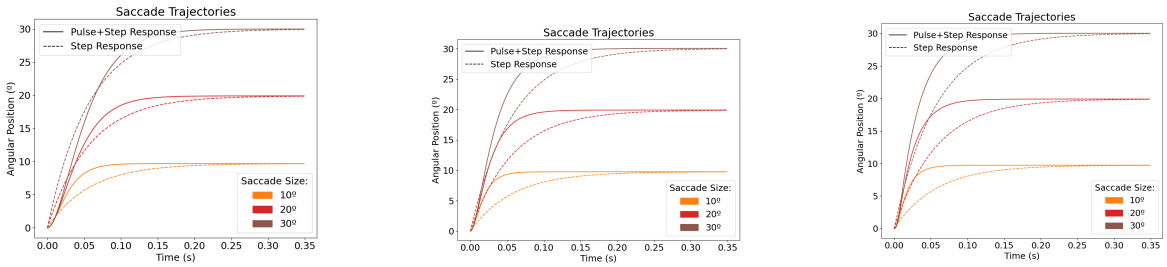


(c) Variant 3

Figure 9: Policy output of the parameters against the desired saccade size of the three variants. The blue line represents the policy output and the orange line the estimated optimal value. The graph on the left is parameter B, the one on the center is $D - F$, and k_0 is the graph on the right.

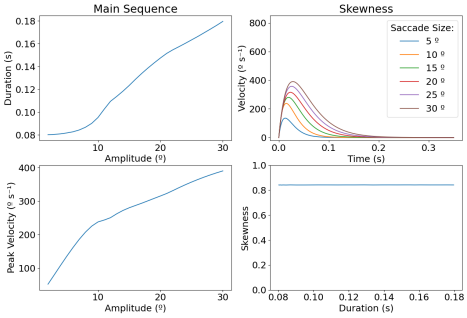
5.1. Discussion

On all variants there is definitely overlap between the learned policy output and the estimated output, which means conclusively that the algorithm does work. The range of values in which B does not match the desired values is due to a search limit that had to be set before training (algorithm limitation), and this value was set to 80 as the upper bound, so

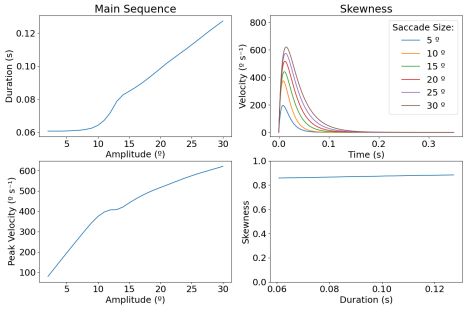


(a) Variant 1 (b) Variant 2 (c) Variant 3

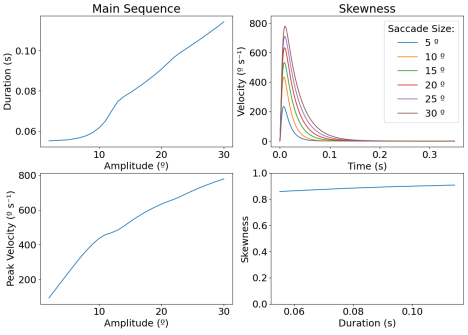
Figure 10: Comparison of the response between a pulse+step signal to a pure step signal for all variants



(a) Variant 1



(b) Variant 2



(c) Variant 3

Figure 11: Main sequence and skewness of the saccades obtained for all variants. It follows the same scheme as figure 2

what happens is just a saturation of the value of B. As it can be seen, the the desired value of B tends to infinity, so even if the upper bound of B is increased there would always be saturation. This is also reflected on the results obtained for low values of amplitude on the main sequence properties, which seem to have the same non-linear properties as the ones found in human data. The same can be said about skewness, although it is slightly inconclusive on variant 1, because upon close inspection the data is very noisy.

Nonetheless, from the results obtained it can be concluded that variant 3 is the best one. Although, all variants output faster response than one from a pure step (Figure 10), variant 3 has the best performance, as not only it performs saccades with the lowest duration, consequence of higher saccade peak a velocities, it has also been found that it spends the least amount of energy (33) to execute a saccade. Moreover, this variant has also the peculiarity of performing a motion that is differentiable at time 0 - it is a realistic movement.

6. Conclusions

In this work it has been designed a framework that uses the machine learning algorithm soft actor-critic to learn a open-loop saccadic control of a biomimetic eye. Due to time constraints, we focused on horizontal saccades, and approximated the system dynamics to a first model. Nonetheless, since the actor-critic method does not assume any particular form of model, the algorithm would also work, in principle, in the full 3D non-linear model. of a human eye. The soft actor-critic learns by maximizing the rewards obtained from applying the control to the model. These rewards were calculated using metrics of accuracy, energy, duration and overshoot of a saccade. The results of the training done were found to be overall successful, the algorithm managed learn the optimal policy for all the variants, and these policies reflected non-linear control properties similar to human records. Variant 1, despite being the one with easiest implementation, performed the slowest. Variant 2,

with the introduction of parameter C, to account for real life limitation, performed better in terms of time, but spent much more energy; factor further corrected by variant 3, conclusively with the best results and with the extra advantage of portraying a realistic motion. Since this work only focused on the movement of the eye in one dimension, using a single motor, for future work, the expansion of the number of motors and number of dimensions for the eye orientation are an option. Other options could be maybe apply this algorithm in closed-loop using inputs with noise.

References

[1] S. P. S. Agostino Gibaldi. The saccade main sequence revised: A fast and repeatable tool for oculomotor analysis. *Behavior Research Methods*, 2021.

[2] J. Belk. Quaternion distance. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/90098 (version: 2011-12-10).

[3] M. D. Binder, N. Hirokawa, and U. Windhorst, editors. *"Retinal Slip"*, pages 3526–3526. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[4] R. E. M. Cardoso. Feedback control of saccades on a model of a 3d biomimetic robot eye. Master's thesis, Instituto Superior Técnico, 2019.

[5] E. Chekaluk and K. Llewellyn. Masking effects in saccadic eye movements. *Studies in Visual Information Processing*, 5:45–54, jan 1994.

[6] B. das Chagas e Silva Colaço Dias. Modeling, simulation, analytic linearization and optimal control of a 6 tendon-driven biomimetic eye: a tool for studying human oculomotor control. Master's thesis, Instituto Superior Técnico, 2020.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[8] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[9] R. Jagtap. Understanding markov decision process (mdp). https://towardsdatascience.com/understanding-the-markov-decision-process-mdp-8f838510f150, September 2020. Accessed: 26/10/2021.

[10] A. John, C. Aleluia, A. J. Van Opstal, and A. Bernardino. Modelling 3d saccade generation by feedforward optimal control. *PLOS Computational Biology*, 17(5):1–35, 05 2021.

[11] S. Kullback. *Information Theory and Statistics*. Dover Publications, 1968.

[12] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[13] M. Lucas. Construction and characterization of a biomimetic robotic eye model with three degrees of rotational freedom. Master's thesis, Instituto Superior Técnico, 2017.

[14] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[15] J. V. Opstal. *The auditory system and human sound-localization behavior*, volume 1. Academic Press, 2016.

[16] D. A. Robinson. Models of the saccadic eye movement control system. *Biologic Cybernetics*, 1973.

[17] R. Shadmehr, J. Orban, M. Xu-Wilson, and T.-Y. Shih. Temporal discounting of reward and the cost of time in motor control. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30:10507–16, 08 2010.

[18] A. Srivats. Information entropy. https://towardsdatascience.com/information-entropy-c037a90de58f, April 2019. Accessed: 26/10/2021.

[19] C. Tavares. Control of saccades with model of artificial 3d biomimetic eye. Master's thesis, Instituto Superior Técnico, 2019.

[20] J. TORRES.AI. The bellman equation. https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7, June 2020. Accessed: 26/10/2021.

[21] B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnagie Mellon University, 2010.