Journal on Interactive Systems, 2020, 11:1 doi: 10.5753/jis.2020.751

An Autonomous Emotional Virtual Character: An Approach with Deep and Goal-Parameterized Reinforcement Learning

Gilzamir F. Gomes [**Universidade Federal do Ceará** | ***gilzamir@alu.ufc.br***]
Creto A. Vidal [**Universidade Federal do Cará** | ***cvidal@dc.ufc.br***]
Joaquim B. Cavalcante-Neto [**Universidade Federal do Cará** | ***joaquimb@dc.ufc.br***]
Yuri L. B. Nogueira [**Universidade Federal do Cará** | ***yuri@dc.ufc.br***]

Abstract We have developed an autonomous virtual character guided by emotions. The agent is a virtual character who lives in a three-dimensional maze world. Results show that emotion drivers can induce the behavior of a trained agent. Our approach is a case of goal parameterized reinforcement learning, which creates the proper conditioning between emotion drivers and a set of goals that determine the behavioral profile of a virtual character. We train agents who can randomly assume these goals while maximizing a reward function based on intrinsic and extrinsic motivations. A mapping between motivation and emotion was carried out. So, the agent learned a behavior profile as a training goal. The developed approach was integrated with the Advantage Actor-Critic (A3C) algorithm. Experiments showed that this approach produces behaviors consistent with the goals given to agents, and has potential for the development of believable virtual characters.

Keywords: *Autonomous Virtual Characters, Emotion, Motivation, Deep Reinforcement Learning*

1 Introduction

Emotion and autonomy of virtual characters are first-class elements of the virtual reality scenario's plausibility. An emotionally guided virtual character can respond more convincingly to interactions with users and promotes user engagement. Therefore, developing a believable autonomous virtual character is still a challenging issue (Gillies, 2018). Moreover, simultaneously getting autonomy in some task and behaviors with emotion and personality traits require meeting multiples goals.

We take a viewpoint that autonomy means agents act by themselves, without external control. Artificial intelligence techniques provide methods to get autonomous agents. Of these techniques, Reinforcement Learning (RL) has a long story with computational theories of emotion and motivations (Moerland et al., 2018). It is doubly hard to find an autonomy that shows emotion and other required properties of the virtual world. Recent work on the development of autonomous Non-Player Characters (NPCs) based on deep reinforcement learning focuses primarily on maximizing a reward function based on the overall game score. The VR emphasis goes beyond the maximization of the overall game score. Specifically, interaction with virtual characters can be a challenge to engage in virtual reality.

Generating a behavior profile according to a writer's specifications of the autonomous characters is a problem in narratives of video games or other virtual reality applications. Justesen et al. (2017) perceive there is currently a lack of tools for designers to efficiently train Non-Player Characters. Therefore, a tool that allows designers to specify desired NPC behaviors (while assuring a certain level of control over the final trained behavior) would accelerate the uptake of these new methods in the game industry (Justesen et al., 2017).

In order to show that an artificial agent can achieve autonomy while emotionally motivated, Gomes et al. (2019) developed an approach based on homeostasis. When homeostatic variables are outside a given range, two things happen. First, the neural component of the decision model is disturbed. Second, a secondary reward for the agent is produced. This approach is a particular type of Goal Parametrized Reinforcement Learning (GPRL) despite the biological and psychological inspiration of these works. Thus, in this paper, we show not only the relationship between GPRL and our approach (Gomes et al., 2019), but also how to obtain character behaviors guided by intrinsic emotions and motivations.

This paper is organized in seven sections, starting with the present introduction. In Section 2, we present some background regarding the topics approached in this work, namely deep neural network, and GPRL. In Section 3, we show state of the art related to believable virtual characters (or agents) with emotions and intrinsic motivations based on reinforcement learning. In Section 4, we describe our approach based on reinforcement learning and deep learning. In Section 5, we describe our main results and findings organized according to our approach. In Section 6, we present and discuss our contributions in the face of our main findings. Finally, we conclude the paper in Section 7.

2 Background

Developing a tool that allows designers to easily specify desired NPC behaviors (and undesired ones) while assuring a certain level of control over the final trained outcomes is challenging problem. To address this problem, we use deep learning and goal-parameterized reinforcement learning. In this section, a brief discussion on the theoretical basis for those two research areas is presented.

2.1 Deep Neural Networks

A classical artificial neural network (ANN) architecture is the Multi-Layer Perceptron (MLP). Traditionally, MLP has up to three layers. Gradient-based optimization struggles to achieve convergence in the training phase of ANNs with more than three layers. However, current progress in algorithms and hardware made gradient-based optimization of RNAs surpass traditional methods in computer vision (Goodfellow et al., 2016) and other fields.

It is easy to determine the structure of neural networks' input and output because it depends only on the nature of the subjacent functions to learn. However, determining the number and the nature of the hidden layers requires more expertise. Although the power of representation of a deep neural network is proportional to the number of layers in its structure, to train such a network is a hard task. Fortunately, there are several heuristics in the literature for the definition of the number of neurons in the hidden layers of neural networks. For example, Hecht-Nielsen (1987) demonstrated, based on Kolmogorov's theorem, that any function of n variables can be represented by $2n + 1$ functions of a given variable. So, the number of neurons in hidden layers can be defined as

$$N_{hidden} = 2 \cdot N_{in} + 1$$

where N_{hidden} is the number of hidden neurons and N_{in} is the number of entries.

One of the major bottlenecks in neural networks was pattern recognition in image data and other types of multidimensional data. Convolutional neural networks efficiently solve this problem. Convolutional networks (ConvNets) are a trainable, biologically inspired architecture that can learn invariant resources. Each stage in a ConvNets is composed of a filter bank, some nonlinearities, and feature pooling layers. With multiple stages, a ConvNet can learn multi-level hierarchies of features. While ConvNets have been successfully deployed in many commercial applications from OCR to video surveillance, they require large amounts of labeled training samples (LeCun et al., 2010). In Reinforcement Learning, Mnih et al. (2016) use two convolutional layers to deal with three-dimensional environments.

2.2 Reinforcement Learning

Mathematically, RL is idealized as a Markov Decision Process (MDP). As the agent interacts with the environment, in a given instant t, it perceives a state s_t and executes an action a_t. The state s_t and the action a_t determine the next state s_{t+1} uniquely. For every action a_t, the agent receives a reward $r_t \in \mathbb{R}$. The cycle perception-action-reward progresses with time.

The function shown on Equation 1 represents the agent's performance, which is the discounted reward R_t from time step t, where r_t is the reward received after the transition from time step t to time step $t + 1$, and γ is a discount term. The term $\gamma \in [0, 1]$ adjusts the importance of the long-term consequences of the agent's actions. The reinforcement signal can be positive, negative, or zero.

$$R_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k}. \tag{1}$$

The agent's policy consists in selecting an action based on a state value function $V : S \rightarrow \mathbb{R}$ or on the $(state, action)$ value function $Q : S \times A \rightarrow \mathbb{R}$ to maximize R_t.

In Approximate Reinforcement Learning, value functions are approximated by parameters θ. The representation of the value functions can be a non-linear model, as several types of neural networks. Mnih et al. (2015) shown that reinforcement learning with deep neural networks gets a superhuman performance in several Atari games. In this approach, the agent selects actions based only on high dimensional inputs representation (as the pixels of video-game frames). This achievement paved the way for the emergence of Deep Reinforcement Learning (DRL).

In DRL, θ is a set of weights of the neural network and, usually, gradient-based optimization is used to maximize the goal function shown in Equation 1. Therefore, in DRL the state value function is $V_\pi(s; \theta)$, which means the expected value from state s using the policy π based on parameters θ.

Asynchronous Advantage Actor-Critic (A3C) is a successful algorithm for many tasks in virtual worlds, and is a baseline for more sophisticated algorithms. Thus, we chose A3C as a baseline for the training of autonomous virtual characters with emotions.

2.3 Asynchronous Advantage Actor-Critic

A3C is a model-free and on-policy reinforcement learning algorithm, which determines a π policy by directly parameterizing it as $\pi(a|s; \theta)$ and using an estimating function $V(s_t; \theta)$. The function $\pi(a|s; \theta)$ means the probability that the agent selects the action a, given the state s in accordance with a parameter θ.

The θ parameters are updated using gradient methods of a function $\mathbb{E}(R_t)$, which is the expected return. The elements that form the basis of the A3C algorithm, the actor, and the critic originate from the family of algorithms REINFORCE (Williams, 1992), whose standard version updates the parameters θ of the policy π in the direction of $\nabla_\theta log\pi(a_t|s_t; \theta)R_t$, which is an unbiased estimate of the expected reward, given by $\nabla_\theta \mathbb{E}[R_t]$. To reduce the variance of this estimate, one chooses the modern approach of gradient-based actor and critic:

$$\nabla_\theta log\pi(a_t|s_t; \theta)(R_t - V(s_t; \theta)), \tag{2}$$

where R_t is an estimate $Q^\pi(a_t, s_t)$ of the policy value function π; and $V(s_t; \theta)$ is an estimate of $V^\pi(s_t)$. Thus, $\pi(a_t|s_t; \theta)$ is an actor that determines the action to perform. The function $V(S_t; \theta)$ is a critic, which estimates the value of a state. The difference between the value of the action in the current state and the estimated value of the state is the advantage of the action in the considered state.

The A3C algorithm runs m copies of both the agent and the environment in parallel. Figure 1 shows the architecture of the A3C algorithm. Instances of the environment and instances of the agents' policies are potentially different from

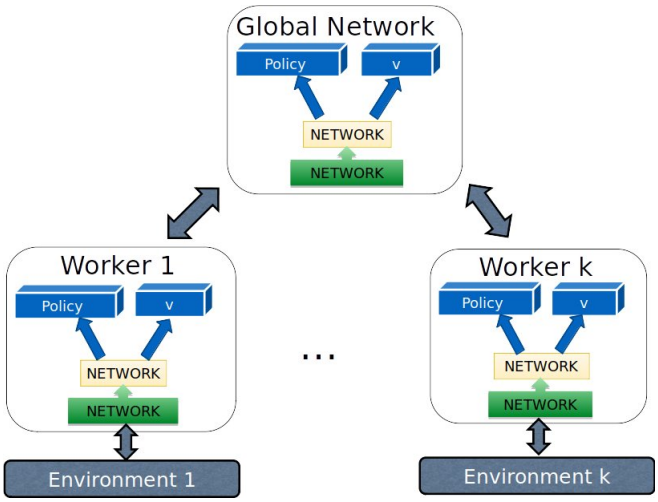


Figure 1. A3C architecture. A worker runs in its own thread or process, interacting with a copy of the environment. A separate process collects each worker's gradients separately and applies them to a global neural network model. Although each worker has its copy of the neural network and runs in its simulation, after a finite number of time steps, the worker's neural network is synchronized with the global neural network.

one another. Each agent has its actor as a $\pi(a_t|s_t;\theta)$ policy and its critic as an estimate of the $V(S_t|\theta_v)$ function. Notice that the policy and the state value function share some of their parameters. Mnih et al. (2016) update both the policy and the value function every t_{max} actions or when the state is terminal. The update performed by the algorithm is calculated as $\nabla_{\theta'} log\pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$, where $A(S_t,a_t;\theta,\theta_v)$ is an estimate of the advantage function given by $\sum_{i=0}^{k-1}(\gamma^i r_{t+i}) + \gamma^k V(S_{t+k};\theta_v) - V(S_t;\theta_v)$ and k can vary from state to state, having an upper bound equal to t_{max}.

Implementations of the A3C algorithm usually use feed-forward or recurrent neural networks with the sharing of parameters by two output-layer branches, one that approximates the policy π and another that approximates the state-value function V that guides the search of π.

2.4 Goal-Parameterized Reinforcement Learning

Goal-Parameterized Reinforcement Learning (GPRL) generalizes Deep Reinforcement Learning (DRL) to multi-goal learning. Although there are many ways of using multi-goals in RL, only two approaches that are relevant to this work are presented.

In GPRL, The common viewpoint is that there exists a vector space where a goal has a representation. Goals' representation is concatenated with the state to shape an expanded state. The value function $V(s;\theta)$ is a single function approximator that estimates a long-term reward from any state s, using parameters θ. Schaul et al. (2015) introduce an approach named *Universal Value Function Approximator* (UVFA). Based on UVFA, the value function becomes $V(s,g;\theta)$, which, now, depends not only on the states s but also on the goals g. Similarly, the policy $\pi(a|s;\theta)$ becomes $\pi(a|s,g;\theta)$, which, now, is conditioned by the goal g.

Andrychowicz et al. (2017) show a way to learn multiple goals in only one interaction. Thus, the agent uses interaction to achieve a goal by learning another goal through modification of the reward function. For example, the agent acts in the environment, producing a transition (s,s',r_g,a,g) where r_g is the reward associated with the goal g. The agent can learn from this transition, but can also use this interaction to learn other goals; to do so, it can change the goal into a new one and recompute the reward, resulting in a new transition $(s,s',r_{g'},a,g')$. The only constraint for doing that is the reward function $R(s,a,s',g')$ to be known. Typically, an agent can have a goal state and a reward function which is 1 if it is into that state and 0 otherwise. At every interaction, it can change its true goal state for its current state and learn with a positive reward.

2.5 Psychological Theories of Emotion

The psychological theories of emotion are generally grounded in current psychological theories. The three main groups of theories of emotion are the categorical theory, the dimensional theories (Russell, 1978), and the componential theories (Lazarus, 1991).

The categorical theory considers a set of discrete emotions. Ekman et al. (1987) identified a pattern of facial expressions across various cultures. These combinations produce a more complex emotional state. Anger, fear, joy, sadness, surprise, and disgust compound the set of basic expressions. A basic emotion is a pattern of response or the tendency of action from an evolutionary viewpoint. Therefore, models of categorical emotions reveal tendencies of actions.

The dimensional theory considers an underlying affective space, which involves two dimensions: valence and arousal.

According to componential theory, emotion is the appraisal of input stimuli in accordance with personal relevance. This theory is about the elicitation of emotion. Examples of appraisal dimensions are valence, novelty, relevance, congruence, and potential coupling.

It is advocated that emotions always precede actions. In a broader viewpoint, emotions are considered in a loop of reward that involves the function and elicitation of emotion.

2.6 Neurophysiological Evidences and Emotional Model

Noradreline's role in emotion and motivation is highlighted in several works. Galvin (1985) concluded that noradrenaline is involved in stress responses, stress pathology, and consequences of stress exposure. Particularly, Harley (1987) highlight the role of noradrenaline in increasing or inhibiting synaptic connections in some regions of the brain.

Specifically, Doya (2002) identified that dopamine signals the error in reward prediction, serotonin controls the time scale of reward prediction, noradrenaline controls the randomness in action selection, and acetylcholine controls the speed of memory update.

2.7 Extrinsic and Intrinsic Motivations

Singh et al. (2010) identified extrinsic and intrinsic rewards. Extrinsic rewards refer to stimuli in the external world, such as food, possibly influenced by internal variables, such as insulin level. Extrinsic motivation is studied together with

homeostasis, while intrinsic motivation shows a strong overlap with the theory of emotion assessment. Our perspective on intrinsic motivation is that adopted by Singh et al. (2010) that intrinsic motivation is independent of the specific environment. For example, mating depends on the environment in which physical contact between two agents is important for the reproduction of the agent. Curiosity, on the other hand, can be implemented in all environments as a more informative state search.

2.8 Homeostasis

Homeostasis, a concept borrowed from biology that inspires many works in computing, can be defined as the ability to maintain the internal environment in an almost constant balance, regardless of the changes that occur in the external environment. In response to internal imbalances resulting from external factors, the agent seeks actions that bring the internal state back into balance. More formally, considering h_t the homeostasis at time, t, the physiological state of the agent at t is described as $H_t = \{h_{1,t}, h_{2,t}, ..., h_{N,t}\}$, where $h_{i,t}$ is the i-th homeostatic variable at t. A homeostatic variable i has a tolerance interval h_i^*. That is, $h_i^* = (min, max)$, where usually $min, max >= 0$ and some cases $max = \infty$.

The agent has no direct control over the value of a homeostatic variable. However, there must be a causal link between the agent's actions and the current or future level of the variables. The variable level is modified by resources that are consumed. The consumption of these resources changes the values of the homeostatic variables. For example, a specific homeostatic variable can increase its value as a result of resource consumption. Other actions or time may decrease the value of the variable. Let $\bar{a}$ be the set of the actions that lead to the consumption of resources and let $\bar{s}$ be a set of states that contains resources. A homeostatic dynamics (Konidaris and Barto, 2006) is usually described as

$$h_{i,t+1} = \begin{cases} h_{i,t} + \psi(s_t, a_t) \text{ if } \mathrm{a_t} \in \bar{\mathrm{a}}, \mathrm{s_t} \in \bar{\mathrm{s}} \\ h_{i,t} - \epsilon, \text{ otherwise,} \end{cases} \quad (3)$$

where $\psi(s_t, a_t)$ is the effect of the presence of the resource and ϵ is a decrease value of a variable whenever resources are not collected. An emotional impulse is explicitly identified as the difference between the current value and the level of the homeostatic variable (for example, $d_{i,t} = 1$ if $h_{i,t} \in h_i^*$, else $d_{i,t} = 0$). The total emotional impulse in the system is defined as in (Cos et al., 2013):

$$D_t = \sum_{i=1}^{N} w_i d_{i,t}, \quad (4)$$

where w_i is the weight of the i-th homeostatic variable.

3 Related Works

There is comprehensive literature on deep reinforcement learning and believable virtual agents or characters. In this section, one selects the works that most closely relates to our approach. For better contextualization, we group these works as follows: those with an emphasis on believable characters, and those with emphasis on reinforcement learning, emotion models, and intrinsic motivation.

3.1 Believable Agents and Virtual Characters

In the context of Virtual Reality, Moussa and Magnenat-Thalmann (2013) use the emotional attachment as an impulse to determine the intensity of decision-making emotions of virtual humans interacting with the user. Their approach uses emotions to calculate rewards for the reinforcement learning mechanism, but they use only emotion appraisal. Furthermore, they do not provide emotional stimulus directly as an input to the decision-making process. Our approach supports virtual humans who interact with the user. Also, we use emotion stimuli directly in the agent decision-making process together with function approximation through neural networks to represent action selection policy. The neural model takes inputs from the emotional model to guide the action selection.

Asensio et al. (2014) presented an approach in which a virtual character, developed with reinforcement learning, exhibits behavior that is guided by intrinsic motivations in accordance with explicit models of human needs. The environment used by those authors was simpler than the one presented in our work. However, it is not clear how reinforcement learning produced the agent's behavior.

Wang et al. (2017) investigated the effectiveness of Q-network based in DRL for interactive narrative personalization. They introduced a bipartite player simulation model that uses a pair of validated classifiers to generate synthetic data on sequences of player actions and outcomes.

In the field of Intelligent Virtual Agents (IVAs), Shvo et al. (2019) propose a computational model of affect that incorporates an empirically-based interplay between its components of personality, motivation, emotion, and mood. Although this approach allows specifying intelligent agents with personality traits, the agent's decision-making is based on classical search and planning. The authors use their approach in a simple quiz problem. We believe that Deep Reinforcement Learning (DLR) is better suited to complex problems.

3.2 Reinforcement Learning, Emotion Models and Intrinsic Motivation

Merrick and Maher (2006) have presented motivated reinforcement learning agents as a means of creating non-player characters that can both evolve and adapt. The author's emphasis is on adaptation to environments in evolution. Also, the authors use Q-Learning for training their agents. Although this publication is old, it has many similarities with our approach. The main differences are that we use the A3C algorithm for training agents with deep neural networks, and our emphasis is on getting emotional guided behaviors with different behavior profiles.

Moerland et al. (2016) implemented the elicitation of fear and hope with reinforcement learning. Their model allows agents to elicit fear and hope during the agent's training and also explains which previous event caused these emotions.

The authors' approach was based on tabular learning, which makes it inappropriate for large state spaces.

Collenette et al. (2017) show how to integrate a humor model into the classic SARSA reinforcement learning algorithm. Moreover, the authors show how this integration can enable self-interested agents to be successful in a multi-agent environment. The apparent model improves the level of cooperation within the multi-agent system compared to the standard SARSA algorithm. Still, the agent interaction model and the agents' ability to differentiate opponents influence the convergence of learning.

Moerland et al. (2018) present several models of emotional agents based on reinforcement learning. Their paper derived some aspects of the reinforcement learning process. The authors found that emotional behavior can persist after training. Many other works deal with virtual characters with reinforcement learning, such as Glavin and Madden (2015), Justesen et al. (2017), and Wang et al. (2017). However, these works usually do not use an emotional model, and do not allow controlled modifications of the character's behavior after training.

Research on intrinsic motivations and reinforcement learning is very comprehensive. In many settings, researchers use curiosity as an intrinsic motivation that generates rewards for encouraging agents to explore the environment and learn useful skills. Pathak et al. (2017) define curiosity as the error of the agent's prediction of his actions' results (Pathak et al., 2017). For Stanton and Clune (2018), curiosity promotes intra-life exploration, rewarding agents to visit as many states as possible in the episode. Gomes et al. (2019) use the elicitation of emotion to motivate an agent to explore its environment. That is the approach we take in this work.

4 Methods

Our agent's architecture is based on reinforcement learning with intrinsic and extrinsic motivations (Singh et al., 2010). Two algorithms used as the basis for new reinforcement learning approaches are DQN (Deep Q-Network) (Mnih et al., 2015) and A3C (Mnih et al., 2016). The A3C algorithm has the advantage of producing significant results even in non-specialized hardware. Using an A3C-derived approach, such as the UNREAL and IMPALA algorithms (Hernandez-Leal et al., 2019), would add an extra layer of complexity that would not bring clear benefits to the current stage of this work. Therefore, A3C was the natural choice for the development of the approaches presented herein.

The agent makes a decision based on the state of its internal environment. Notice, however, that the agent's decision is restricted to a discrete set of actions, and that the agent's internal state is a set of variables driving the agent's emotional state. Hence, in this work, the development of a virtual character is realized in five stages (see Figure 2):

1. Mapping of internal state levels to an emotional state,
2. Definition of the agent's behavioral profiles,
3. Training of the agent,
4. Evaluation of the agent's general behavior in the environment, and
5. Validation of the behavior profiles.

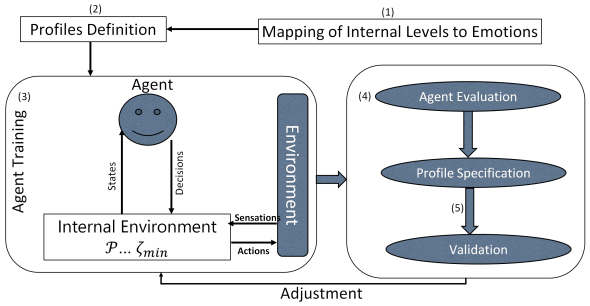


Figure 2. Stages to obtain the agents' profiles in the explicit motivation model. The left block shows the architecture of the reinforcement learning agent with intrinsic motivation given by Singh et al. (2010).

In stage (1), one uses two variables that drive agent emotions. First, the restitution of the agent's energy at the predefined level drives the agent to high arousal and positive affective states. However, the energy level out of a specific energy range drives agents to high arousal but negative affective states. This negative state can result in an emotional state of dismay or despair. The resulting emotional state depends on the agent's behavior profile.

Direct agent interaction with objects of the external environment changes energy level, and that change produces an internal reward. Therefore, the energy level drives extrinsic motivation. Also, when the energy level is outside adequate ranges, it perturbs the agent's decision making. Patience is another factor whose value results from the evaluation of state sequences seen by the agent. That factor produces an intrinsic motivation because changes in its value generate rewards based on the history of the expected state's value. We made the mapping of internal variable values to emotional states before the agent's training.

In stage (2), energy levels and patience levels determine the agent's behavioral profile. Agent Behavioral Profile is a set B of ranges of internal variables. Each subset $b \in B$ determines a list of ranges or limits for each internal variable. Behavior profiles determine a pattern of behavior. For example, an agent with a high level of patience may take longer to respond to negative states. Each behavior profile has a goal because of the reward function change for each behavior profile. So, one uses Goal-Parameterized Reinforcement Learning (GPRL) to address this issue.

In stage (3), during agent training, multiple goals are selected during each agent's lifespan, and the reward function associated with each goal is calculated. So, we get the conditioning between the agent's profile and the agent's policy function. We evaluate three strategies of GPRL (Andrychowicz et al., 2017). In Section 4.3, we show more details about this.

In stage 4, we evaluate whether or not the behavior of the agent is consistent with the target environment. Our approach is coupled with a decision model based on neural networks. Before detailing each stage separately, we show an overview of this decision model.

4.1 Agent's Decision Model

The developed agent's decision-making model consists of a feed-forward neural network that receives two groups of inputs. The first input group is a sequence I_k =

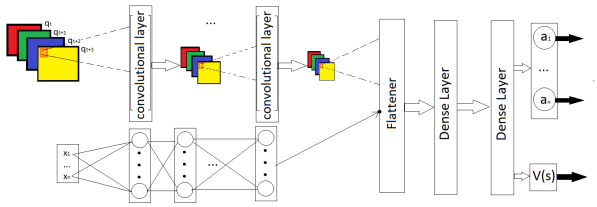


Figure 3. The general architecture for training agents with intrinsic motivations.

$(q_t, q_{t+1}, ..., q_{t+k-1})$ of k images captured by the agent's virtual camera from time t to time $t+k-1$ (one frame for each discrete time instant). The second input group is a vector of real numbers containing the state variables that model the state of the agent's internal environment. The neural network has two output groups. The first one represents the agent's policy, giving the probability of each action to be selected for execution. The second group consists of a scalar indicating the expected future value of the state perceived by the neural network. More details are presented In the next section, one shows more detail about the network's architecture.

4.1.1 The Agent's Neural Network

The agent's decision model is a multi-layer feed-forward neural network with two types of input and two types of output. Therefore, this network has a two-stream architecture. Two stream deep neural networks were applied in other domains as action detection in video (Simonyan and Zisserman, 2014). We adapted this idea to select an action based on an environment signal containing different input types. The first input is a sequence of frames from the agent's virtual camera. The second input is a vector of real numbers from the state of the agent's internal environment. A set of convolutional layers is attached to the first input because convolution is an efficient operation for extracting features from images. Another independent set of fully-connected layers is attached to the second input. Those layers are called dense layers.

The architecture of the network used in this work is shown in Figure 3. We show an instance of that architecture, which works for two different environments, and, in Section 5, that instance is put to work.

The agent's neural network parameters were optimized to get maximum the Function 1. So, we present the agent's learning task in the next section.

4.1.2 The Agent's Learning Task

The tasks considered in this work are those in which an agent interacts with the environment E through a sequence of actions, observations, and rewards.

In each interaction t, in response to an observation x_t, the agent selects an action a_t from a set of possible actions A. The action modifies the environment, which can be a virtual world simulator. The agent can receive a reward r_t for the execution of the selected action. In general, E may be stochastic.

The agent perceives the external environment, its own internal environment, and its own goals. Goals indicate what behavior profile the agent should generate. That profile defines the current internal reward. Thus, an observation is a tuple $x = (x_e, x_p, x_g)$, where x_e is the agent's view of the external environment, x_p are the proprioceptions and internal levels of the agent, and x_g is a goal given to the agent.

An observation x_e of the external environment occurs through the agent's artificial vision. The considered environment is partially observable and noisy, which makes it difficult for the agent to understand the current situation based only on the current state. Therefore, an observation x_e can be expressed in the form of a sequence $I_k = (q_t, q_{t+1}, ..., q_{t+k-1})$ of k frames. Since an observation x_e is of the form I_k, one of the agent's inputs is a tensor of the form $k \times W \times H$, where k is the size of the sequence, W and H are the width and height of the frames of the sequence. The agent's proprioceptions are real-number vectors that indicate the internal state of the agent. Behavior profiles are ranges within which the variables must be maintained. The goals are inputs to the agent's neural network indicating which behavior profile must be produced. Also, the conditioning between the policy and the value function of the agent with its goals was generated. These are represented by $\pi(s, g; \theta)$ and $V_\pi(s, g; \theta)$ where s is the current state, g is the current goal, and θ is a set of weights of the agent's neural network. So, one preserves the Markovian property of the agent's decision making.

Therefore, the agent takes sequences of actions and observations represented by $s_t = x_1, a_1, x_2, a_2, ..., x_t, a_t$, and learns strategies of action in the environment that depend on those sequences, which have a finite number of time steps (Mnih et al., 2015). This formalism results in a finite Markovian Decision Process (MPD - Markov Decision Process) in with each sequence represents a distinct state. Thus, we can use reinforced learning techniques for MDPs by simply applying the complete sequence s_t as a state representation at an instant t. In this work, one used A3C algorithm (Mnih et al., 2016) for this. One of the main advantages of the A3C algorithm is the ability to run on non-specialized hardware. Another important advantage is that the algorithm is the baseline to faster state-of-the-art algorithms on deep reinforcement learning when applied to bi-dimensional and three-dimensional scenarios.

4.2 Mapping of Internal Levels to Emotions

The emotion elicitation is closely linked to the generation of internal rewards for the agent. Changes to the agent's energy levels and changes to the expected state value produce the agent's internal reward. This section presents how our model of intrinsic and extrinsic motivations is used to get the agent's internal reward function.

4.2.1 Reward Shaping and Emotion Elicitation

In this work, the general reward as a function of t (time step) is represented as

$$R(t) = r_t + r_t^{\Delta}, \tag{5}$$

where r_t is an environment-dependent external reward function, and r_t^{Δ} is an internal reward based on the emotional engine. The value of that internal reward usually results from the emotional model's evaluation. Therefore, the emotional

rewardr_t^{Δ}, which is zero on the outset of step t of an episode, increases in certain emotional states elicited at time t.

Emotion elicitation stimulates the agent's decision process while changing the reward function. After training, it is expected that the conditioning caused by the **emotion-reward-emotion** loop will arise. There are two sources of internal rewards: extrinsic motivations and intrinsic motivations.

4.2.2 Reward Resulting from the Extrinsic Motivation

The **energy** variable relates to extrinsic motivation because the emotion associated with it depends on external environment resources. Depending upon the type of environment resource the agent gets, its energy may increase or decrease. However, the agent's always lose energy over time. In this work, we use a homeostatic variable called $\mathcal{E}$, which indicates the agent's energy level. The range $E_r = (\mathcal{E}_{min}, \mathcal{E}_{max})$ of this variable is taken from the current profile b.

When $\mathcal{E}$ is outside E_r, three things happen: (1) r_t^{Δ} decreases; (2) connections to the output layer of the agent's network are strengthened or weakened; and (3) the goal variable D changes its value to indicate that the current goal is to bring $\mathcal{E}$ inside E_r. The strengthening or weakening of the connections to the output layer of the agent's network is done through a multiplier term $\mathcal{M}_f$, which is applied to the layer immediately preceding the output of the neural network. $\mathcal{M}_f$ is taken from the current profile b. Its value can be:

$$\begin{cases} \mathcal{M}_f = 1 \text{ (neutral)}, & \text{if } \mathcal{E} \in \mathrm{E_r}, \\ \mathcal{M}_f \in (0,1) \text{ or } \mathcal{M}_f > 1, & \text{if } \mathcal{E} \notin \mathrm{E_r}. \end{cases}$$

Whether $\mathcal{M}_f \in (0,1)$ or $\mathcal{M}_f > 1$, depends upon the agent's profile b.

4.2.3 Intrinsic Motivations Model

We derive another control variable $\mathcal{P}$ – **Patience** – associated with the variable $\mathcal{C}_p$. The variable $\mathcal{C}_p$ counts the number of state transitions without increasing the expected state value. In this work, $\mathcal{C}_p$ relates to intrinsic motivation because the emotion associated with it is elicited from the variation of the function V_π. Let $g_t = \{s_0, a_0, ..., s_i, a_i, ..., s_t, a_t\}$ be the complete history of agent-environment interaction at time t, the sensory stimulus over g_t in time $t > 0$, named I_t, is equal to $V^\pi(s_t, b; \theta) - V^\pi(s_{t-1}, b; \theta)$. Now, let $\mathcal{P}$ be the maximum interval without realizing positive stimuli until the agent enters into a state of impatience. The variable $\mathcal{C}_p$ is incremented whenever an agent fails to receive positive stimulus. If $\mathcal{C}_p$ is greater than a threshold $\mathcal{P}$, the agent's emotional state changes to impatience. If the agent receives a positive sensory stimulus, then $\mathcal{C}_p$ is defined as equal to $zero$.

Satisfaction is a result of the evaluation of $\mathcal{C}_p$. Thus, if $\mathcal{C}_p > \mathcal{P}$, one defines $\Omega_i = -1$; otherwise, $\Omega_i = 1$. Along with the limits of other variables, $\mathcal{P}$ is defined at the beginning of an episode as a result of selecting an agent's behavior profile b. In this case, Ω_i indicates whether or not the goal of learning the associated homeostatic behavior deserves attention. Therefore, Ω_i is equivalent to a goal definition to satisfy the pattern of behavior associated with $\mathcal{C}_p$.

4.2.4 Reward Resulting from the Intrinsic Motivation

If $\Omega_i = -1$, the mechanism of elicitation of an evaluation variable ζ is performed. In this work, the value of ζ was computed as the average entropy of the output of the agent's policy $\pi(s, g; \theta)$ in the last k steps of the current episode (one used $k = 30$).

So, if $\Omega_i = -1$, the entropy of the agent's decisions ζ must be within a range $\zeta_r = [\zeta_{min}, \zeta_{max}] \in b$. If the agent manages to keep the entropy of his actions within that range, it is rewarded. This interval is previously chosen between a set of predefined real intervals $\{[0.1, \infty], [0.5, \infty], [0.9, \infty]\}$.

4.2.5 Conditioned Emotional Response

Conditioned Emotional Response is the learned emotional reaction or the response to certain conditioning stimuli (Estes and Skinner, 1941). We used conditioning during the training stage. In this way, after the agent is trained, a user can change the behavior of the agent by changing the tolerance limits under which the variables operate. Concerning the energy $\mathcal{E}$, one uses the multiplier term $\mathcal{M}_f$ induced by the internal energy levels of the agent. The energy levels produce an agent's emotional response from a conditioning principle inspired by biology. More specifically, this strategy was inspired by the idea that *norepinephrine* increases the chemical energy in the body to give quick responses in a stressful situation (Doya, 2002). Thus, along with the scale change in the weights of connections arriving at the network's output layer, a signal indicating the agent's energy level (D), of three possible levels, was sent as network input: low, medium, and high. The D level is calculated as follows:

$$\text{if } \begin{cases} \mathcal{E} < \mathcal{E}_{min} \ \rightarrow \mathrm{D} = -0.5, \text{ low energy}, \\ \mathcal{E}_{min} \leq \mathcal{E} \leq \mathcal{E}_{max} \rightarrow \mathrm{D} = 0.5, \text{ normal energy}, \\ \mathcal{E} > \mathcal{E}_{max} \ \rightarrow \mathrm{D} = 1, \text{ high energy}. \end{cases}$$

The values of D are input to the agent's neural network.

The variables $\mathcal{E}_{min}, \mathcal{E}_{max}$ and $\mathcal{E}$ are part of the agent's perceptual input. Thus, after the agent's training, if conditioning is successful, the agent acts in accordance with a user-defined range.

With respect to Ω_i, we chose these intervals to match three different levels of emotional responses. As part of the agent's perceptual input, one defines the list $[\zeta_{min}, \zeta_{max}, \zeta, \Omega_i]$ establishing the conditioning of the user's input with the expected behavior of the agent. Thus, if $\Omega_i = -1$, its reward r_t^{Δ} is incremented by a value greater than zero (according to Equation 5), so that conditioning is generated only when the agent is disappointed.

The agent's neural network receives an input vector $[\mathcal{E}_{min}, \mathcal{E}_{max}, \mathcal{E}, \zeta_{min}, \zeta_{max}, \mathcal{C}_{pmax}, \Omega_i, D]$ whose components are taken form the agent's internal state, profile definition and current goal. Some variables in that vector define the behavior profile, while other variables define goals associated with behavior profiles. The choice of appropriate values is made through a discretization of the input space, which we describe in the next section.

4.3 Definition of the Profiles

The agent's current profile b in an episode is denoted by $b = [(\mathcal{E}_{min}, \mathcal{E}_{max}), (\zeta_{min}, \zeta_{max}), \mathcal{C}_{pmax}]$. Profile b is take from a distribution B of profiles.

The list of all profiles is largely dependent upon the type and characteristics of the simulated environment. The general idea is that ranges are determined in the domains of the emotional model's variables. The limits of those ranges determine a certain pattern of behavior. For instance, if $\mathcal{E} \in [0, 1000]$, examples of limits are $E_r^1 = [0, 30], E_r^2 = [10, 20], E_r^3 = [10, 40]$, such that $E_r^* = \{E_r^1, E_r^2, E_r^3\}$; or, if $P \in [0, 1000]$, P_r^* could be $P_r^* = [30, 50, 100, 300, 500]$. If the emotional model has only two variables, $\mathcal{E}$ and P, set B is defined as $\{b : b = (E_{min}, E_{max}, P_i) \forall [E_{min}, E_{max}] \in E_r^* \wedge \forall P_i \in P_r^*\}$.

The agent's profile can be defined, after its training, through a Graphical User Interface (GUI). Figure 4 shows an example of a GUI to configure the agent's behavior profile.

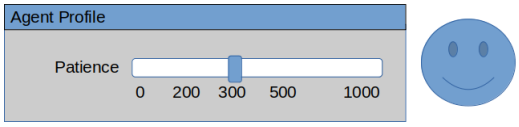


Figure 4. A partial view of the Graphical User Interface (GUI) used to determine how the agent reacts to its own emotions.

4.3.1 Agent's Goals

The agent's current goal g in an episode is denoted by $g = [D, \Omega_i]$. Goals indicate what behavior profiles agents must get. If $D = 1$ and $\Omega_i = 1$, the agent's reward is generated mainly from external reward sources. In this case, the agent's focus can be to select actions that maximize external reward. If $D <= 0.5$ or $\Omega_i = -1$, the agent's focus can be to select actions that maximize internal reward.

Thus, this approach is equivalent to goal-parameterized reinforcement learning eliciting emotional states while maximizing an external reward.

4.4 Agent Training

During training, the agent assumes a behavior profile (b) from the set B of possible profiles. A profile is selected from B in accordance with a uniform distribution before the next episode starts.

The internal reward r_t^{Δ} is produced according to the agent's current profile $b \in B$, which is selected at the outset of an episode.

In this work, one uses the A3C algorithm to maximize the future discounted reward. We tested the three following approaches for the A3C algorithm.

4.4.1 Agent with Explicit Goals

The main approach used to select state transitions in the A3C algorithm was to maintain the properties of the original A3C algorithm, but use the concept of extended state. Extended state is a signal (x_e) from the external environment, plus a signal (x_p) from the internal environment and a description (x_g) of the agent's current goal. Transitions are generated from the extended state $s_{t,T} = (x_q(t, T), x_p(t, T), x_g(T))$ in time step t to the extended state $s_{t+1,T} = (x_e(t + 1, T), x_p(t + 1, T), x_g(T))$ in time step $t + 1$. For simplicity, since the episodes are independent of one another, we omit the episode's index T. For example, one mentions the reward at time t as r_t and not as $r_{t,T}$. When the agent performs an action a at time step t, it receives a reward r_t and gets a new extended state s_{t+1}. The state transition is represented by $Tr(t, t + 1) = (s_t, a_t, r_t, s_{t+1})$. Other characteristics of the A3C algorithm are maintained according with their original description in (Mnih et al., 2016). Note that $x_g(T) = (D, \Omega_i)$ and $x_g(T)$ does not change during episode T.

Although we have separate goals for each behavior profile, in this approach, the total reward of the agent is always the sum of the external and internal rewards. Algorithm 1 (Apprendix A) shows the strategy of behavior profile and goal selection for this approach.

4.4.2 Agent without Explicit Goals

Another approach used to select state transitions also uses the concept of the extended state. In this case, transitions are generated from the extended state $s_t = (x_q(t), x_p(t))$ in time step t to the extended state $s_{t+1} = (x_e(t + 1), x_p(t + 1))$ in time step $t + 1$. Thus, no explicit goal is given in the agent's transitions. Algorithm 2 (Appendix A) shows the reward calculation for this approach.

4.4.3 Agent with Explicit Randomic Goals

Despite some similarities with the latter approach, this approach has two main differences. First, maximizing external reward is now an explicit goal. Second, only one goal is selected at a time to add up to the total reward. Algorithm 3 (Apprendix A) shows the reward calculation for this approach.

4.5 Agent Evaluation and Profile Validation

The two last stages of virtual character development are agent evaluation and profile validation. In agent validation, one analyzes, visually, whether or not the general agent's behavior converges to coherent actions in the virtual environment. For example, if the agent sees a fruit, it must get it when the agent's level of energy is low.

In profile validations, one analyzes, visually, whether or not the agent's behavior is coherent with the currently selected behavior profile, and whether or not the agent's overall behavior is compatible with the allowable actions in the environment.

If errors or inconsistencies are noted in the last two stages of the agent's development, one returns to the first stage.

5 Experiments

We developed two environments as testbeds for the ideas in this paper. The first one is a traditional grid world, used

to select good ideas before they are tested in a more complex environment. The more complex environment is a three-dimensional virtual maze. An agent explores this maze with the help of first-person view and touch information. Therefore, in this section, one describes both environments and the agents used in the developed tests.

5.1 The Grid World

We used an instance of the traditional grid world environment. Our version of this environment consists of a grid with one hundred cells arranged in a matrix of ten lines by ten columns (Figure 5). The cells are indexed by a pair (l, c) of two non-negative integer coordinates: l is the line index and c is the column index, where $0 \leq l, c < 10$. The grid world is a discrete, completely observable, static, episodic environment.

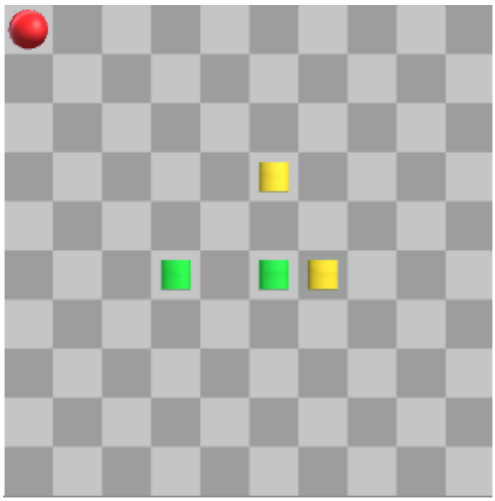


Figure 5. The agent's environment is a grid world environment. The red circle represents the agent. A cell with a green cylinder increases the life of the agent if it reaches that cell. A cell with a yellow cylinder decreases the life of the agent if it reaches that cell.

An agent can move around the grid, one block at a time, in two orthogonal directions (keeping l of c constant). The actions for this case are: move to the right, move to the left, move up and move down. The agent is blocked from getting off the limits of the world.

The cells of the environment can be occupied by two kinds of cylinders: green cylinders and yellow cylinders. If the agent reaches a cell with a green cylinder, it earns one life; on the other hand,if it reaches a cell with a yellow cylinder, it loses one life. The life span of an agent is approximately ten seconds. There are two ways an episode finishes: when the agent's lifespan expires, or when the agent loses all its lives (initially the agent gets three lives).

5.1.1 The Grid World's Agent

At time step t, the external input to the agent in this experiment is a matrix q_t that represents the state of the environment at that time step, i.e., $x_e^t = (q_t)$. Each cell of the input matrix stores the code of the object that occupies that cell (see the objects' codes in Table 1).

In addition, one uses a simplified motivation model for this agent. The agent has only one internal variable $\mathcal{E}$ that

Table 1. Object's code in the grid world.

Object	Code
Agent	1
Green Cylinder	3
Yellow Cylinder	4
Empty Cell	0

represents the energy level of the agent. The agent starts an episode with an energy level $\mathcal{E} = \frac{1}{2}(\mathcal{E}_{min} + \mathcal{E}_{max})$. The agent's energy decreases by 1 percent of its current energy level for each movement the agent makes between time steps t and $t+1$. The energy level increases by 10 percent of the maximum energy level at each time step t if the agent stops. The values $\mathcal{E}_{min}$ and $\mathcal{E}_{max}$ are chosen at the beginning of an episode from a set of values of predefined intervals, which are selected according with an uniform distribution. Thus, the vector of the agent's internal levels is composed of $\mathcal{E}_{min}$, $\mathcal{E}_{max}$, $\mathcal{E}$ and D. Consider D_t to be the value of D at time step t. Thus, $D_t = -1$ if the energy level is outside the range $[\mathcal{E}_{min}, \mathcal{E}_{max}]$ and $D_t = 1$, otherwise. Therefore, $x_p^t = (\mathcal{E}_{min}, \mathcal{E}_{max}, \mathcal{E})$ and $x_g^t = (D_t)$.

The agent actions are up, down, right, left, and no operation (NoOp). The final score of the agent in each episode is the agent's number of lives left.

5.1.2 The Grid World Reward

Reward in the grid world results from the agent interaction with the environment (external reward) and from the control of the agent's internal energy level (internal reward). Reward can be considered a guide of intrinsic motivation because it does not directly affect the actions of the agent in the external environment but, rather, depends on the history of agent-environment interaction. At the beginning of a time step t in a given episode T, both the internal reward (r_t) and the external reward (r_t^{Δ}) are equal to zero. Those rewards are modified according with the actions that occurred in time step $t-1$ on state s_{t-1} and that produced the current state s_t.

The external reward at time t is equal to 1, if the agent reaches a cell with a green cylinder, and, -1, if the agent reaches a cell with a yellow cylinder. The agent's external reward on the last time step of an episode is one (1) if the agent's number of lives is greater than zero (0), otherwise it is zero (0).

The internal reward at time step t is calculated as

$$\text{if } \begin{cases} \mathcal{E} < \mathcal{E}_{min} & \rightarrow \mathrm{r}_\mathrm{t}^{\Delta} = \mathrm{r}_\mathrm{t}^{\Delta} - 0.01, \\ \mathcal{E}_{min} \leq \mathcal{E} \leq \mathcal{E}_{max} & \rightarrow \mathrm{r}_\mathrm{t}^{\Delta} = \mathrm{r}_\mathrm{t}^{\Delta} + 0.01, \\ \mathcal{E} > \mathcal{E}_{max} & \rightarrow \mathrm{r}_\mathrm{t}^{\Delta} = \mathrm{r}_\mathrm{t}^{\Delta} - 0.01. \end{cases}$$

The total reward at time step t is calculated according to Equation 5.

5.2 The Three-Dimensional Virtual Maze World

The environment of the Maze World contains nutritious and poisonous fruits, which are randomly scattered throughout the maze. The environment is episodic, which means that an

episode starts when the agent enters the maze by a fixed gate, and ends either when the agent dies of starvation or when the agent gets out of the maze. The agent successfully exits the maze when it manages to find a key and returns to the portal. The topology of the maze is shown in Figure 14(a).

Environment-agent interactions define the external component r_t of the reward function in Equation 5. The term r_t is zero at the start of an episode and changes in accordance with the agent's actions. Every time an agent picks up a nutritious fruit, the variable r_t increases by 10 (ten) units. On the other hand, every time an agent picks up a poisonous fruit, r_t decreases by 10 (ten) units. Yet, if an agent picks up the gold key, r_t increases by 300 units. Thus, Equation 5 has a direct dependence on the agent's actions and on its emotional state.

5.2.1 The Maze World's Agent

The agent has a physical body as shown in Figure 6(a). Agent's physical body enables it to interact with the environment and to navigate through the environment, performing several actions. Table 2 shows the codes associated with common actions.

Table 2. Action codes.

Action	Code
Walk	0
Flipping	1
Run	2
Look to the left	3
Look to the Right	4
Pick up (touched object)	5

The agent's perceptions are the pair (O, P), where O is a sequence of images that represents the agent's view of the world, and P is a vector with values from your internal state, from your profile definition and your current goal. The agent's artificial vision is based on *ray casting* and produces the sequence of images O. In our model, k images with 20×20 pixels are generated by casting rays from the agent's eye position. For this, one uses asymmetrical frustum with a 90-degree aperture angle, as shown in Figure 6(b). The distance from the eye to the image plane is d. The vector P contains the behavior profile, the current goal definition, and the agent's proprioceptions. Proprioception information has two values. The first value is either 0 (indicating no touch) or 1 (indicating touch). If an object is touched, the second value of the pair is the identifier of that object, otherwise (no touch), it is 0 (zero). The identifier of a poisonous fruit is -1, and the id value of a nutritious fruit is 1.

5.3 The Agent's Neural Network

We use a $feedforward$ neural network to represent the agent's action selection policy. Biologic evidence indicates that different input signals are processed by different neural organizations (Lodish et al., 2000). Thus, as shown in Figure 7, our neural network architecture has two specialized streams: one is specialized in the processing of emotions, and another is specialized in visual processing. The output of this neural network is compatible with the expected outputs in the

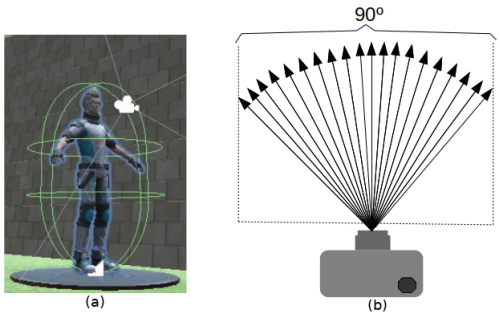


Figure 6. (a) Agent's physical body and (b) ray casting with an aperture angle of 90 degrees and symmetrical perspective projection.

A3C algorithm. Therefore, our network has an output array for the policy $\pi(s, g; \theta)$ and an output value for the function $V_\pi(s, g; \theta)$.

The visual stream of the neural network architecture was based on the architecture of Mnih et al. (2016) and Mnih et al. (2015). The network used a convolutional layer with 16 filters of size 3×3 with stride 1, followed by a convolutional layer with 16 filters of size 3×3 with stride 2. Convolutional layers were connected to a fully-connected layer with 512 hidden neurons. This hidden layer is shared with the emotional stream of the agent's neural network. All three hidden layers were followed by a rectifier for non-linearity. The model used by actor-critic agents had two sets of outputs. The first output is a softmax output with one entry per action representing the probability of choosing the action. Second is a single linear output representing the value function.

The stream for processing the emotional model contains two fully connected (dense connections) layers ($E1$ and $E2$), each with thirty neurons. In the maze world scenario, the input of the emotional model consists of internal levels and limits ($[\mathcal{E}_{min}, \mathcal{E}_{max}, \mathcal{E}, \zeta_{min}, \zeta_{max}, \zeta, \mathcal{C}_{pmax}]$); goal parameters $[D, \Omega_i]$; and touch sensory inputs $\mathcal{T}$ and $\mathcal{V}$. Therefore, according to the heuristic proposed by Hecht-Nielsen (1987), thirty neurons are more than sufficient to learn emotional patterns with an input size of eleven (11) values. In the grid world, the entry corresponding to the internal levels consists of $[\mathcal{E}_{min}, \mathcal{E}_{max}, \mathcal{E}]$. The goal entry consists of only one variable, D. Even though the entry is simpler, in this problem, one used the same number of neurons in the maze world experiment.

The output of the second convolutional layer was concatenated with the $E2$ layer. The resulting list of neurons was connected to the shared layer of hidden neurons (H) employing dense connections. The H layer has connections to an output layer and the neuron V_s. The neuron V_s predicts the expected value of the current state. Usually, the neuron V_s is used only during training because it is used in the algorithm A3C to calculate the advantage of the current action. However, one uses the value of V_s as input from the agent's emotional model. Thus, not rule out the value of V_s after training in our emotional approach. Output of the V_s neuron is the output of the function $V_\pi(s, g; \theta)$.

5.4 Training Parameters and Simulations

One uses the RMSProp gradient optimization algorithm since this optimizer has obtained good results in maze exploration (Mnih et al., 2016). The parameters' values are equal

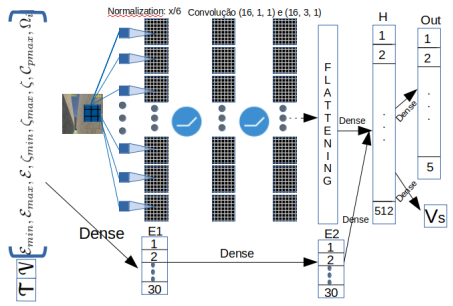


Figure 7. Agent neural network model.

to those set by Mnih et al. (2016), except for the number of ignored frames, which we defined as eight, since values smaller than eight did not produce significant differences between neighboring states. Moreover, the maximum amount of steps after which the current neural network of a thread is updated was set to thirty, since we realized that the commonly used value in (Mnih et al., 2016) takes longer to converge. Table 3 shows a summary of the parameters' values. All experiments used a discount of $\gamma = 0.99$ and an RMSProp decay factor of $\alpha = 0.99$.

Table 3. Learning Parameters (Mnih et al., 2016).

Parameter	Value
Number of *threads*.	8
Number of frames in which a selected action is repeated.	8
Size of the frame sequence in each step.	4
Initial learning rate	10^{-4}
A3C update steps (*tmax*)	30
normalized maximum gradient	5.0
Discount rate of the A3C algorithm (γ)	0.99
Decay factor of the RMSProp algorithm	0.99
Entropy bonus (β)	0.001

We performed experiments using the same network architecture, the same optimization algorithm, and the same algorithm settings for the two types of environments analyzed. The only difference between the two experiments is that the neural network input to the grid world agent is one frame with shape $M \times M$ where $M = 10$.

The experiments for the grid world analyze strategies to be used in a more complex environment. The different strategies analyzed were those described in algorithms 1, 2, and 3. As the best result obtained was with the strategy shown in Algorithm 1, only this approach was evaluated in the three-dimensional maze world. One performed six simulations for all strategies used in the grid world, and five simulations with the best strategy in the maze world.

All simulations were performed with the parameters specified in Table 3, using the RMSProp optimization algorithm (Mnih et al., 2016). To define the number of training hours in both experiments, we observed the learning curve. For this, initial training was carried out to detect how long the agents' average performance would take to stabilize. In the simulations of the grid world, that stabilization time was about two hours; and, in the simulation of the three-dimensional world of the maze, the stabilization time was about eight hours.

6 Results and Discussions

In this section, the presentation is organized in accordance with the performed simulations: the grid world; and the three-dimensional maze world.

6.1 Grid World's Simulations

We analyzed the agent in the grid world to verify whether the agent's behavior was consistent with the environment (through the external reward maximization), and to verify whether the goals given as behavior profiles were achieved. For that, four groups of six simulations were performed. In the first group of simulations, the original A3C algorithm (Mnih et al., 2016) was used, with the parameters shown in Section 5.4. The purpose of this simulation is to compare the results obtained with the approaches shown in algorithms 1, 2, and 3 with the results obtained with the standard A3C algorithm. In all simulations, the same agent settings were used. In the other three groups of simulations, we obtain the average performance of the following algorithms: A3C with Explicit Goal (A3CEG), A3C with Explicit Randomic Goal (A3CERG), and A3C without Explicit Goal (A3CWEG).

6.1.1 Training phase

The training lasted eight hundred episodes. The average sum of rewards in the last one hundred registered training episodes in six simulations is shown in Table 4. One registers one episode every other one hundred episodes.

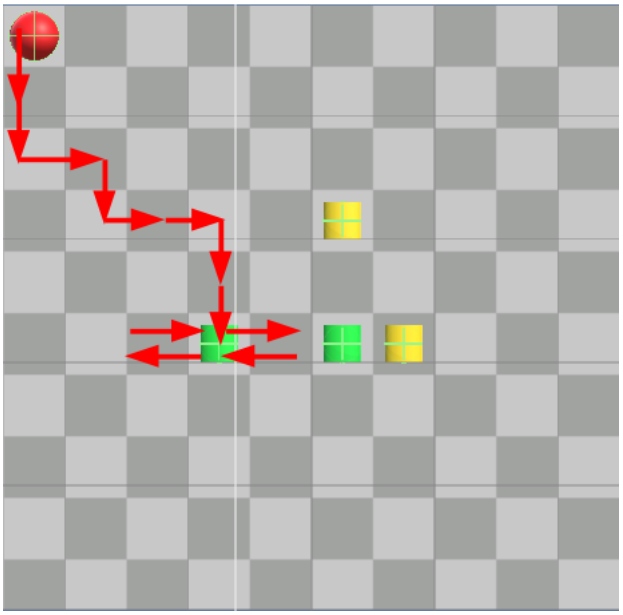
Table 4. The average sum of rewards in the last 100 registered training episodes over six simulations. We register one episode every other one hundred episodes.

Agent Strategy	Sum of Rewards
A3C	240
A3CEG	70
A3CERG	120
A3CWEG	219

The agent trained with the standard A3C algorithm receives a higher sum of rewards than the agents trained with the other three A3C algorithms. The agent trained with the A3CEG algorithm receives the lowest sum of rewards per episode.

Therefore, the default A3C agent has optimal behavior if we consider only the maximization of the external reward as a goal. In this case, optimal behavior is when the agent gets in and out of cells occupied by green cylinders. This is the behavior of the agent with the default A3C algorithm. Figure 8 shows an example of the trajectory generated by the standard A3C agent. However, the default A3C algorithm does not allow agents with different behavior profiles. Thus, the behaviors of the trained agents A3CEG, A3CERG, and A3CWEG were analyzed qualitatively after the agent's training, focusing on the learning of behavior profiles and on the influence of goals in the agent's decision.

Figure 8. An example of a trajectory obtained in an execution of the grid world agent with the standard A3C algorithm.



6.1.2 Agents' Behavior Profile

We analyzed the coherence of the agents' behavior regarding the homeostatic goal of maintaining the internal level within the range of the corresponding variable. For each behavioral profile of an agent, one shows a graph with the average evolution per episode of the variable's value and the limits within which the variable must be kept.

First, one tried the **A3CWEG** agent. The convergence to the expected behavior in the environment occurred in approximately one hour and a half. However, all training sessions ran for two hours. Although this agent has achieved consistency with the goal in the external environment, Figure 9 shows that the A3CWEG agent does not respond to the change in the behavior profile. Regardless of the behavior profiles analyzed, the behavior of the variable (green line) was the same: the agent's energy always drops to zero.

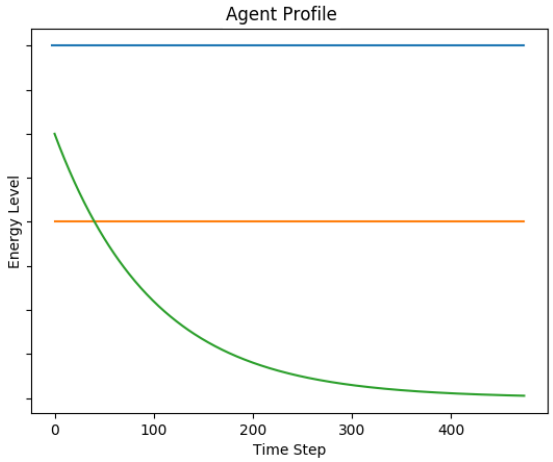


Figure 9. Typical behavior of the variable controlled by the agent A3CWEG. The green line indicates the variable evolution. The orange and blue lines indicate the variable's target range (min, max). The variable's behavior was the same consolidated in this figure, regardless of the target range selected.

Next, given that the A3CWEG agent failed to obtain different behavior profiles after training, we attempted to design an agent with explicit goals, adopting two different strategies.

In the first strategy, named A3CERG, the agents select goals at random at the outset of an episode. However, the agents also have some explicit goals that define the computing of the reward function. i.e., to maximize the external reward function. Figure 10 shows the agent's average sum of rewards about one hundred episodes after the learning stage; and Figure 11 shows the performance of the A3CERG agent after the learning stage. The result is an average of about one hundred episodes, and the purpose is to maximize the internal reward function. One can observe that the agent's internal variable is sensitive to the chosen interval. Besides, in all cases, when the variable goes outside its limits, the agent tends to change its behavior to bring it back to an appropriate level. The problem here is that the goal must be given manually to the agent after the learning step to obtain autonomous behavior.

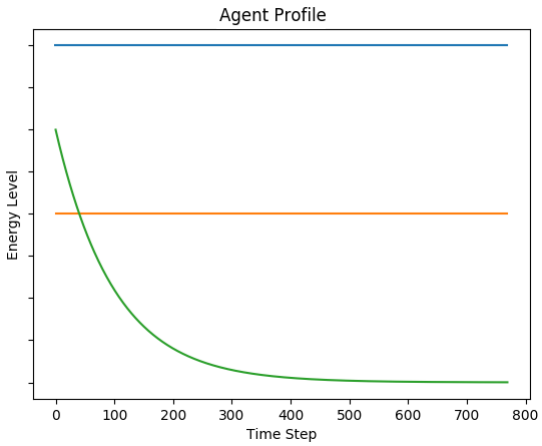


Figure 10. Typical behavior of the variable controlled by the agent A3CERG. The green line indicates the variable evolution. The orange and blue lines indicate the variable's target range (min, max). The variable's behavior was the same consolidated in this figure, regardless of the target range selected.

In the second strategy, named A3CEG, the goals were set explicitly, with the simultaneous computation of the internal reward and the external reward. However, each time a goal is selected, a new episode is created. Figure 12 shows the result of the analyzed behavior profiles.

It is observed that, for a different range of the variable, the agent's behavior is coherent with the its internal goal. The agent is also consistent with the goal derived from the external reward, although it performs much fewer movements than the other agents. The agent's trajectory is compatible with the path of the standard A3C agent (Figure 8).

6.2 Maze World's Simulations

6.2.1 Overall description

The behavioral analysis of the maze world agents after training was done based on their respective trajectories. Although the general strategy is based on the A3CEG agent's strategy, it is more complex because it involves more than one internal variable. The analysis of trajectories is widely used

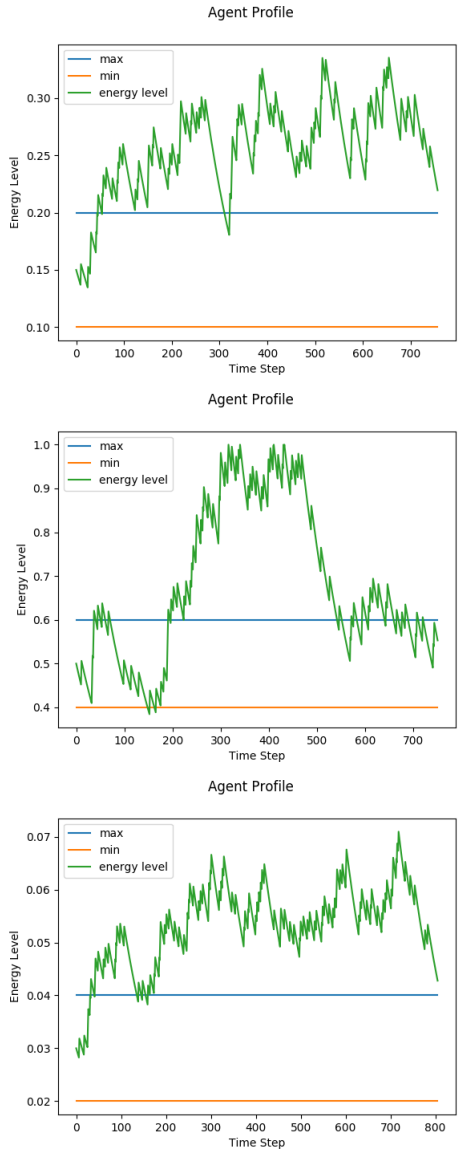


Figure 11. In these three graphs, the control of the A3CERG agent is shown over the current value of the internal variable (green line). It is observed that for different intervals of the variable, the behavior of this variable changes because the goal is to maximize the internal reward.

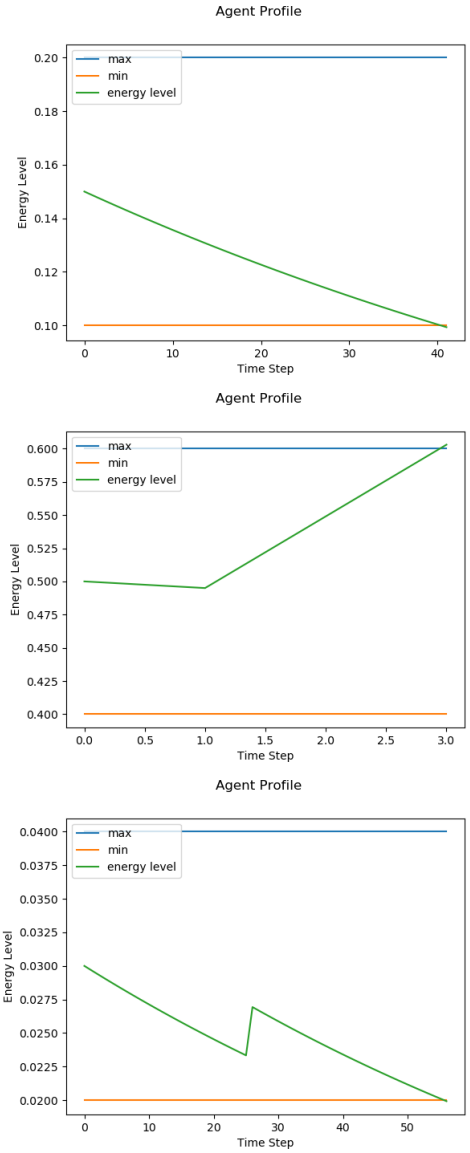


Figure 12. In these three graphs, the control of the A3CEG agent is shown over the current value of the internal variable (green line). It is observed that for different intervals of the variable, the agent's behavior is coherent with external and internal goals.

to visualize the behavior of agents in virtual environments, such as mazes. Therefore, to analyze the data generated in the simulations of this work, a visual exploratory analysis of the agent's trajectories inspired by Bechtold et al. (2018) was carried out. The trajectories show the agent's emotional state, which is summarized as a concept of happiness. Thus, looking at the agent's ordered pair (excitation-discouragement, satisfaction-disappointment), the agent is:

- happy, if ($\mathcal{E}_{min} \leq \mathcal{E} \leq \mathcal{E}_{max}$ and $\mathcal{C}_p \leq \mathcal{P}$);
- unhappy, if ($\mathcal{E} \notin [\mathcal{E}_{min}, \mathcal{E}_{max}]$ and $\mathcal{C}_p > \mathcal{P}$); or
- emotionally neutral, in other cases.

We represent the agent's steps by the lengths of line segments; and its emotional state by the colors of line segments. Thus, green segments represent states of happiness; blue segments represent states of neutral emotion, and red segments represent states of sadness. There might be an overlap between two colors when the agent passes twice through a given position, each in a different emotional state. So, if C_a is the previous line color and C_b is the recent line color, then the new line color C_n is calculated as: $C_n = 99\%C_a + 1\%C_b$. The maze's floor is colored white, and the maze's walls are black. The lines are the tracks that show the agent's trajectory. Since the amount of emotional state combinations is large, only a subset of the configurations was considered. This subset contains variables that have led to a substantial change in the agent's behavior. So one condensed the emotional information of the agent in a single variable called happiness.

Also, one analyzed the average speed of the agent in all configurations of emotional variables. One presents only the subset of the results with the most considerable difference in the agent's final behavior. The configurations of this subset are shown in Table 5. Each row in this table corresponds to a different controller.

Table 5. Agent emotional parameters.

#	$\mathcal{E}_{min}$	$\mathcal{E}_{max}$	ζ_{min}	$\mathcal{M}_{\int}$	$\mathcal{C}_{pmax}$
1	10	40	0.9	1.9	$3k$
2	30	40	0.9	1.9	$3k$
3	30	40	0.9	0.1	$3k$
4	5	40	0.1	1.9	100
5	5	40	0.9	1.9	100
6	5	40	0.9	1.9	500

6.2.2 Discussions

We present the results in two steps: exhibition of the agent's average reward during five training stages; and analysis of the higher scored emotional controller.

In Figure 13 (a), one can see the evolution of the average reward per episode. An examination of the controller generated after the training shows that, in fact, the learned controller makes the agent avoid the poisonous fruits (red spheres) and seek the nutritious fruits (white spheres). Besides, Figure 13 (b) shows that the value of the states increases with time. Finally, in figure 13 (c), the number of steps per episode increases substantially throughout the training. These three observations show that learning was effective.

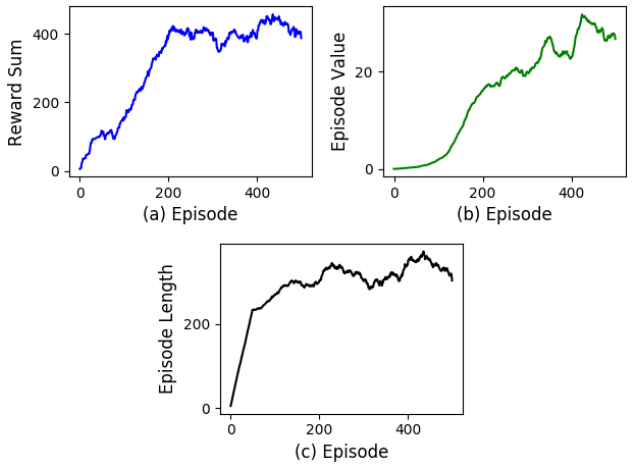


Figure 13. In these figures, the horizontal axes show the episode and vertical axis show (a) the average reward, (b) the state value, and (c) the average amount of time steps by episode. One thousand episodes take eight hours approximately.

We analyzed that the trained controller obtained the highest sum of rewards for different configurations of the emotional model's variables. In figures 14, 15 and 16, one shows the effect of changing the variables $\mathcal{E}_{min}$ and $\mathcal{M}_{\int}$. The controllers corresponding to the configurations of line 1 to line 2 of Table 5, one only changed the value of $\mathcal{E}_{min}$. So we verified how $\mathcal{M}_{\int}$ affects the agent's behavior. Thus, the controller corresponding to line 2 discouraged earlier. The result was a shortening of the agent's trajectory. In addition to that, the controller with the configuration of line 1 spent more time in the happy state than the controller with the configuration of line 2.

We perceived that the agent's speed with the controller configuration 2 is higher than that of configuration 1 when the multiplier is greater than 1 (one). That difference is generated because the minimum level on line 2 causes $\mathcal{M}_{\int}$ to act earlier on the efferent channel. In this case, the value of $\mathcal{M}_{\int} > 1$ was associated with a higher average speed of the agent. Although that result may look inconsistent at first, it can be understood from the analogy that people act according to their personalities in certain situations. In danger, some people become desperate and act on impulse, and others become discouraged and barely able to move.

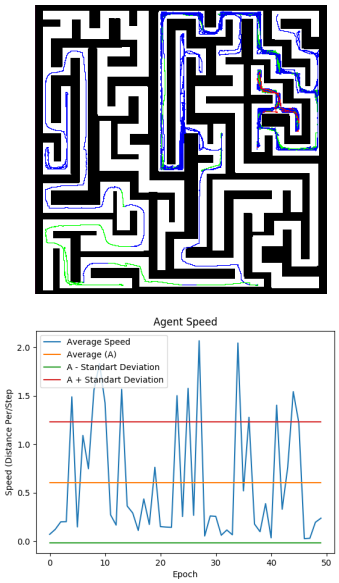


Figure 14. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in row 1 of Table 5.

With the configurations of lines 1 and 2 in Table 5, one noticed that the change in the minimum energy threshold of the agent indicates coherent changes in its behavior. In Figure 16, one shown the results corresponding to the controller with all parameters copied from line 2 of Table 5, except $\mathcal{M}_{\int}$, which was set to 10%. Thus, we verified if the response to the low level of energy was coherent with the type of response defined by the choice of the multiplier term. We perceived that the agent's speed was significantly lower with this configuration. Therefore, the agent's behavior changes coherently according to the choice of the multiplier value. The multiplier application generates a shortening in the trajectory of the agent. If $\mathcal{M}_{\int} > 1$, the agent increases his speed and consequently the probability of ignoring fruits on its way. Thus, the increase in speed was not triggered by a rational decision, but rather by term $\mathcal{M}_{\int} > 1$. If $0 < \mathcal{M}_{\int} < 1$, the trajectory of the agent is affected (shortened) due to a reduction of the agent's speed. In this way, the agent behaves consistently with the choice $\mathcal{M}_{\int}$ when its energy level is below a certain predefined value.

In figures 17 and 18, we show the effect of the change of value of the variable ζ_{min} when the emotion is elicited according to the agent's energy. We kept all properties equal between the two configurations except for the uncertainty of the agent through high energy level. As a result, the color difference between the trajectories of the two figures is noticeable. If the agent receives a positive reward to increase the uncertainty about his actions, its trajectory increases because it tries to take more actions in low-level states. We perceive that in Figure 20, where the agent's uncertainty is the entropy of the choice of its actions calculated based on the history of

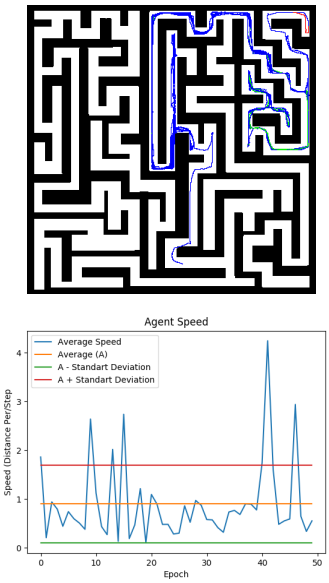


Figure 15. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in row 2 of Table 5.

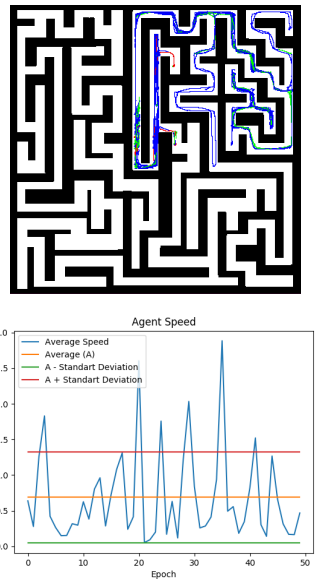


Figure 17. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in row 4 of Table 5.

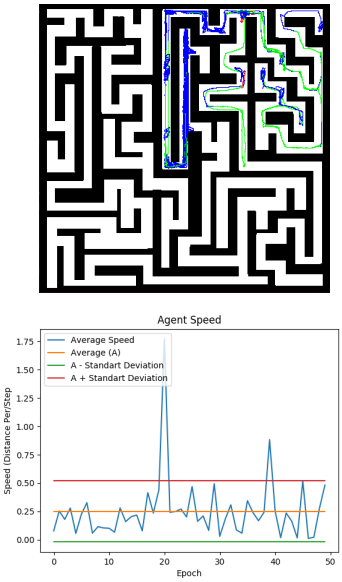


Figure 16. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in row 3 of Table 5.

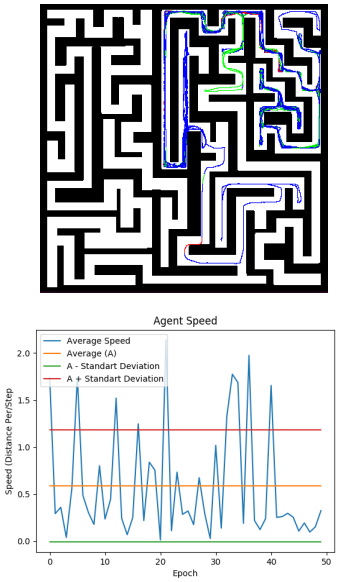


Figure 18. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in row 5 of Table 5.

actions of the last thirty simulation steps in emotional states of *dismay*.

Figure 19 shows the result obtained when using the agent's configuration in line 6 of Table 5. Here, concerning line 5, only the agent's patience changed, from 300 to 500 non-positive interactions. A substantial difference between the two configurations is that the agent with the configuration of line 6 is in the emotional state of *sadness* fewer times than the agent with configuration of line 5.

The results showed that the agent's behavior changes in a way that is consistent with the configurations of its emotional model. However, as seen in Figure 20, the agent fails to take the variable that represents the entropy of its own decisions to stay above the minimum threshold of 0.9. Our hypothesis here is that a strict markovian model is not sufficient to adequately capture conditions that are not explicitly represented in the current state of the agent.

7 Conclusions

We have developed agents with a configurable emotional model based on A3C algorithm (Asynchronous Advantage Actor-Critic). Thus, after the agent's training, predefined changes in control variables led the agent to obtain behaviors that were consistent with the choice of the settings made *a priori*. We show that this approach is a type of goal-parameterized reinforcement learning. In this case, the parameterization was designed to project different emotional responses in accordance with a given behavior profile. To show the equivalence between the developed approach and the goal parameterized reinforcement learning, we designed a simple grid world, and the results were consistent with the initial hypotheses suggested.

Therefore, we obtained an emotional agent based on reinforcement learning. More refinements, however, are wel-

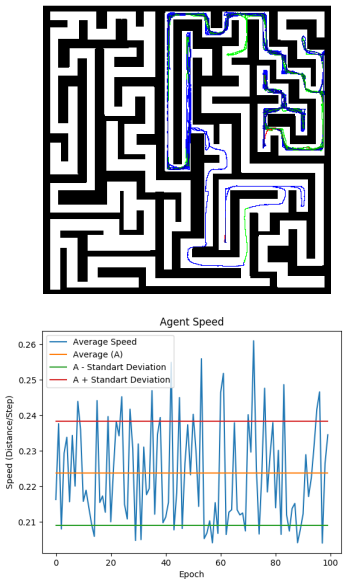


Figure 19. (a) The agent emotional state, and (b) the agent speed with the emotional model's configurations equal to that shown in line in line 6 of Table 5.

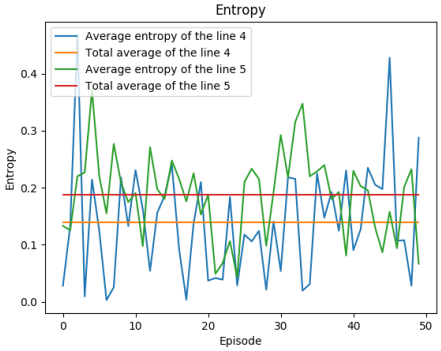


Figure 20. Comparison of the average agent entropy with the configurations shown in lines 4 and 5 of Table 5.

comed to achieve greater accuracy in customizing NPC behaviors. This result is interesting for virtual reality applications because the elicitation of emotions can be used to configure avatar animations consistent with their emotional states. Thus, possible applications of this work are the development of autonomous virtual characters with different emotional profiles for video games, virtual characters that mimic human emotion to virtual training of medical clinic attendants in stress situation and to training autonomous vehicles to deal with human emotion in simulated environments as the Carla Simulator (Dosovitskiy et al., 2017).

The next step of this research is an in-depth analysis of all the components of the developed model and determining empirically and theoretically how these components affect the agent's behavior. Then, an improvement of the model can be performed to obtain more precision in the generated behaviors. The ultimate goal is to get a robust approach for generating believable virtual character behaviors that look autonomous.

Acknowledgment

The authors would like to thank CNPq, the State University of Acarau Valley, and the Graduate Program in Computer Science of the Federal University of Ceará (MDCC/UFC) for their financial support.

This paper is an extended and revised version of the paper (Gomes et al., 2019).

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5055–5065, Red Hook, NY, USA. Curran Associates Inc.

Asensio, J. M. L., Peralta, J., Arrabales, R., Bedia, M. G., Cortez, P., and Peña, A. L. (2014). Artificial intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters. *Expert Systems with Applications*, 41(16):7281 – 7290.

Bechtold, F., Splechtna, R., and Matkovic, K. (2018). Visual Exploratory Analysis for Multiple T-Maze Studies. In Puig Puig, A., Schultz, T., Vilanova, A., Hotz, I., Kozlikova, B., and Vázquez, P.-P., editors, *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 203–213. The Eurographics Association.

Collenette, J., Atkinson, K., Bloembergen, D., and Tuyls, K. (2017). Mood modelling within reinforcement learning. *Artificial Life Conference Proceedings*, 14(29):106–113.

Cos, I., Cañamero, L., Hayes, G. M., and Gillies, A. (2013). Hedonic value: Enhancing adaptation for motivated agents. *Adaptive Behavior*, 21(6):465–483.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Doya, K. (2002). Metalearning and neuromodulation. *Neural Netw.*, 15(4):495–506.

Ekman, P., Friesen, W. V., O'Sullivan, M., Chan, M., Diacoyanni-Tarlatzis, I., Heider, K., R., et al. (1987). Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of Personality and Social Psychology*, 53(4):712–717.

Estes, W. K. and Skinner, B. F. (1941). Some quantitative properties of anxiety. *Journal of Experimental Psychology*, 29(5):390–400.

Galvin, G. (1985). Stress and brain noradrenaline: a review. *Neurosci. Biobehav. Rev*, 9:233–243.

Gillies, M. (2018). Creating virtual characters. In *Proceedings of the 5th International Conference on Movement and Computing*, MOCO '18, New York, NY, USA. Association for Computing Machinery.

Glavin, F. G. and Madden, M. G. (2015). Learning to shoot in first person shooter games by stabilizing actions and clustering rewards for reinforcement learning. In *2015 IEEE*

Conference on Computational Intelligence and Games (CIG), pages 344–351.

Gomes, G., Vidal, C. A., Cavalcante Neto, J. B., and Nogueira, Y. L. B. (2019). An emotional virtual character: A deep learning approach with reinforcement learning. In *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, pages 223–231.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Harley, C. (1987). A role for norepinephrine in arousal, emotion and learning?: Limbic modulation by norepinephrine and the kety hypothesis. *Progress in neuropsychopharmacology biological psychiatry*, 11:419–58.

Hecht-Nielsen, R. (1987). Kolmogorov's mapping neural network existence theorem. In *Proceedings of the international conference on Neural Networks*, volume 3, pages 11–14. IEEE Press New York.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797.

Justesen, N., Bontrager, P., Togelius, J., and Risi, S. (2017). Deep learning for video game playing. *CoRR*, abs/1708.07902.

Konidaris, G. and Barto, A. (2006). An adaptive robot motivational system. In *From Animals to Animats 9*, pages 346–356. Springer.

Lazarus, R. S. (1991). Cognition and motivation in emotion. *American Psychologist*, 46(4):352–367.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE.

Lodish, H., Berk, A., Zipursky, S. L., and et al. (2000). *Molecular Cell Biology*, chapter 4. W. H. Freeman, 4 edition.

Merrick, K. and Maher, M. L. (2006). Motivated reinforcement learning for non-player characters in persistent computer game worlds. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '06, page 3–es, New York, NY, USA. Association for Computing Machinery.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., G Bellemare, M., Graves, A., Riedmiller, M., K Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–33.

Moerland, T. M., Broekens, J., and Jonker, C. M. (2016). Fear and hope emerge from anticipation in model-based reinforcement learning. In *IJCAI*, pages 848–854.

Moerland, T. M., Broekens, J., and Jonker, C. M. (2018). Emotion in reinforcement learning agents and robots: a survey. *Machine Learning*, 107(2):443–480.

Moussa, M. B. and Magnenat-Thalmann, N. (2013). Toward socially responsible agents: integrating attachment and learning in emotional decision-making. *Computer Animation and Virtual Worlds*, 24(3-4):327–334.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2778–2787. JMLR.org.

Russell, J. A. (1978). Evidence of convergent validity on the dimensions of affect. *Journal of Personality and Social Psicology*, 36(10):1152 – 1168.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320.

Shvo, M., Buhmann, J., and Kapadia, M. (2019). An interdependent model of personality, motivation, emotion, and mood for intelligent virtual agents. In *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, pages 65–72.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Trans. on Auton. Ment. Dev.*, 2(2):70–82.

Stanton, C. and Clune, J. (2018). Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. *arXiv preprint arXiv:1806.00553*.

Wang, P., Rowe, J., Min, W., Mott, B., and Lester, J. (2017). Interactive narrative personalization with deep reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3852–3858.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Appendices

Appendix A: Algorithms

Algorithm 1 Agent-based on Explicit Goals (A3CEG). Agent, environment, and profiles are objects that encapsulate agent, environment, and agent's profile.

procedure AgentStep($agent, environment, profiles$)
  $goals \leftarrow [\,]$ ▷ A list of goals
  $internalreward \leftarrow 0$ ▷ Sum of internal rewards
  if new episode **then**
    $x_e^i, x_p^i, x_g^i \leftarrow initial\_state$
  for all $profile \in profiles$ **do**
    $a \leftarrow profile.min, b \leftarrow profile.max$
    $m \leftarrow a + (b-a)/2$
    $v \leftarrow profile.current\_value$
    $pv \leftarrow profile.preview\_value$
    $r_t^\Delta \leftarrow 0$
    if $(a \leq v \leq b)\ and\ (pv < a\ or\ pv > b)$ **then**
      $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
      $profile.goal \leftarrow neutral$ ▷ usually 0
    else if $v < a$ **then**
      if $|v-m| < |pv-m|$ **then**
        $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
      else
        $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
      $profile.goal \leftarrow low$ ▷ usually -1
    else
      if $|v-m| < |pv-m|$ **then**
        $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
      else
        $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
      $profile.goal \leftarrow high$ ▷ usually 1
    $internalreward \leftarrow internalreward + r_t^\Delta$
    $goals.append(profiles.goal)$
  $x_e \leftarrow frames\_of\_the\_world$
  $x_p, x_g \leftarrow agent.get\_internal(profiles)$ ▷ The agent returns your internal state
  $action \leftarrow agent.act(x_e, x_p, x_g)$ ▷ The agent chooses an action.
  $final\_x_e, reward, done = environment.step(action)$ ▷ It performs a simulation step.
  $Transition \leftarrow ((x_e^i, x_p^i, x_g^i), action, reward + internalreward, (x_e, x_p, x_g))$
  $A3C\_learning\_from(Transiction)$
  if not $done$ **then**
    $x_e^i \leftarrow x_e, x_p^i \leftarrow x_p, x_g^i \leftarrow x_g$

Algorithm 2 Agent Without Explicit Goals (A3CWEG). Agent, environment, and profiles are objects that encapsulate agent, environment, and agent's profile.

procedure AgentStep($agent, environment, profiles$)
  $internalreward \leftarrow 0$ ▷ Sum of internal rewards
  if new episode **then**
    $x_e^i, x_p^i \leftarrow initial\_state$
  for all $profile \in profiles$ **do**
    $a \leftarrow profile.min, b \leftarrow profile.max$
    $m \leftarrow a + (b-a)/2$
    $v \leftarrow profile.current\_value$
    $pv \leftarrow profile.preview\_value$
    $r_t^\Delta \leftarrow 0$
    if $(a \leq v \leq b)\ and\ (pv < a\ or\ pv > b)$ **then**
      $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
    else if $v < a$ **then**
      if $|v-m| < |pv-m|$ **then**
        $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
      else
        $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
    else
      if $|v-m| < |pv-m|$ **then**
        $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
      else
        $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
    $internalreward \leftarrow internalreward + r_t^\Delta$
  $x_e \leftarrow frames\_of\_the\_world$
  $x_p \leftarrow agent.get\_internal(profiles)$
  $action \leftarrow agent.act(x_e, x_p)$
  $final\_x_e, reward, done \leftarrow environment.step(action)$
  $Transition \leftarrow ((x_e^i, x_p^i), action, reward + internalreward, (x_e, x_p))$
  $A3C\_learning\_from(Transiction)$
  if not $done$ **then**
    $x_e^i \leftarrow x_e, x_p^i \leftarrow x_p$

Algorithm 3 Agent-based on Explicit Randomic Goals (A3CERG). Agent, environment, and profiles are objects that encapsulate agent, environment, and agent's profile.

procedure AgentStep($agent, environment, profiles$)
  $goals \leftarrow [\,]$ ▷ A list of goals
  $internalreward \leftarrow 0$ ▷ Sum of internal rewards
  if new episode **then**
    $x_e^i, x_p^i, x_g^i \leftarrow initial\_state$
  $profile \leftarrow uniform(profiles)$ ▷ Gets a random access to a list of profiles uniformly distributed.
  $a \leftarrow profile.min, b \leftarrow profile.max$
  $m \leftarrow a + (b-a)/2$
  $v \leftarrow profile.current\_value$
  $pv \leftarrow profile.preview\_value$
  $r_t^\Delta \leftarrow 0$
  if $(a \leq v \leq b)\ and\ (pv < a\ or\ pv > b)$ **then**
    $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
    $profile.goal \leftarrow neutral$ ▷ usually 0
  else if $v < a$ **then**
    if $|v-m| < |pv-m|$ **then**
      $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
    else
      $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
    $profile.goal \leftarrow low$ ▷ usually -1
  else
    if $|v-m| < |pv-m|$ **then**
      $r_t^\Delta \leftarrow r_t^\Delta + \epsilon$ ▷ $\epsilon > 0$
    else
      $r_t^\Delta \leftarrow r_t^\Delta - \epsilon$ ▷ $\epsilon > 0$
    $profile.goal \leftarrow high$ ▷ usually 1
  $internalreward \leftarrow internalreward + r_t^\Delta$
  $x_e \leftarrow frames\_of\_the\_world$
  $x_p, x_g \leftarrow agent.get\_internal(profile)$
  $action \leftarrow agent.act(x_e, x_p, x_g)$
  $final\_x_e, reward, done = environment.step(action)$
  $Transition \leftarrow ((x_e^i, x_p^i, x_g^i), action, reward + internalreward, (x_e, x_p, x_g))$
  $A3C\_learning\_from(Transiction)$
  if not $done$ **then**
    $x_e^i \leftarrow x_e, x_p^i \leftarrow x_p, x_g^i \leftarrow x_g$