(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 May 2022 (19.05.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/103676 A1

(51) International Patent Classification:
A61N 1/18 (2006.01)

(21) International Application Number:
PCT/US2021/058390

(22) International Filing Date:
08 November 2021 (08.11.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17/095,617 11 November 2020 (11.11.2020) US

(71) Applicant: **SONY INTERACTIVE ENTERTAINMENT INC.** [JP/US]; 1-7-1 Konan, Minato-Ku, Tokyo 108-0075 (JP).

(72) Inventors: **BASHKIROV, Sergey**; C/O Sony Interactive Entertainment LLC, 2207 Bridgepointe Parkway, San Mateo, California 94404 (US). **TAYLOR, Michael**; C/O Sony Interactive Entertainment LLC, 2207 Bridgepointe Parkway, San Mateo, California 94404 (US).

(74) Agent: **ISENBERG, Joshua** et al.; C/O JDI PATENT, 809 Coporate Way, Fremont, California 94539 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) **Title:** METHOD FOR ROBOTIC TRAINING BASED ON RANDOMIZATION OF SURFACE STIFFNESS

(57) **Abstract:** A method, system and computer product for training a control input system involve taking an integral of an output value from a Motion Decision Neural Network for one or more movable joints to generate an integrated output value and generating a subsequent output value using a machine learning algorithm that includes a sensor value and a previous joint position if the integrated output value does not at least meet the threshold. Surface stiffness interactions with at least a simulated environment, a rigid body position and a position of the one or more movable joints based on an integral of the subsequent output value are simulated. The Motion Decision Neural Network is trained with the machine learning algorithm based upon at least a result of the simulation of the simulated environment and position of the one or more movable joints.

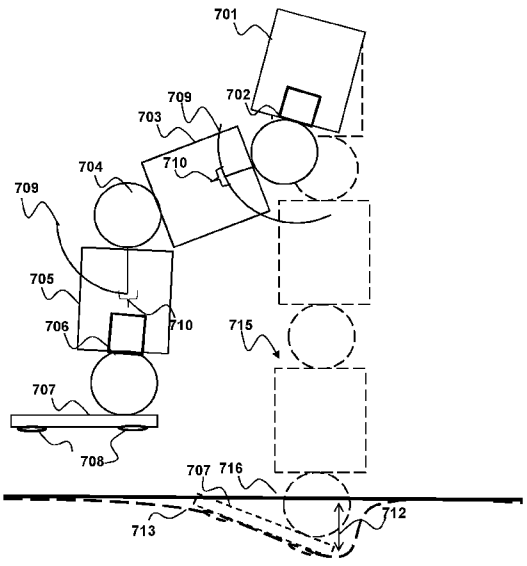


FIG. 7

WO 2022/103676 A1

METHOD FOR ROBOTIC TRAINING BASED ON RANDOMIZATION OF SURFACE STIFFNESS

FIELD OF THE DISCLOSURE

Aspects of the present disclosure relate to motion control using machine learning specifically aspects of the present disclosure relate to the training of Neural Networks in physics based animation and motion control systems.

BACKGROUND OF THE DISCLOSURE

A common technique for models is to create a virtual skeleton for the model with flexible or movable joints and rigid bones. A virtual skin is overlaid on top the virtual skeleton similar to how human muscle, fat, organs, and skin is integrated over bones. Human artists then painstakingly hand animate movement sets for the object using the virtual skeleton as a guide for the range of motion. This is a time consuming process and also requires an artistic touch as there is a narrow window between life like movements and movements that fall into the uncanny valley. Some production studios avoid the difficult and time-consuming process of life-like animation by employing motion capture of human models. This technique is expensive and can be time consuming if a large number of motions are required or if there are many different characters that need to be modeled.

Robots may be modeled virtually with bones for rigid sections and joints for movable sections. This type of model control makes it easier for robot animators to create life-like movements for the robot. Movement of the joints in the virtual skeleton may be translated to movement of the motors controlling the joints in the robot. The virtual model applies constraints to the joints to simulate the real world limitations of the joints of the robot. Thus, a virtual model of the robot may be used to control the physical robot. This sort of control is useful for animatronics.

A major problem with animation is the need for human controlled movement creation. Hand animation of characters is time consuming and infeasible for situations where many characters are needed with different movement characteristics, such as a scene of a mall in space where there are many different alien characters that have vastly different anatomies. One technique that has been used to lighten the load of animators in these situations is to generate one or two different movement models and then apply those movement models randomly to moving characters in the scene. This technique works well with many different

models and a few different characters but on large scale, it creates a noticeable unnatural effect where many characters obviously are identical.

Machine learning represents an area that could be employed in character animation to reduce the need for human animators. Currently movement produced by neural networks trained using machine learning techniques results in unnatural jittery movements and special efforts have to be taken and or constraints on the solution put in place to avoid the problem of jitter. Additionally, current machine learning animation techniques fail to account for several real-world constraints. The lack of real-world constraints in animation models created through machine learning means that they are unsuitable for use as virtual models for controlling physical robots, especially in condition sensitive areas such as walking and balancing mechanics.

It is within this context that aspects of the present disclosure arise.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present disclosure can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1A is a simplified node diagram of a neural network for use in motion control according to aspects of the present disclosure.

FIG. 1B is a simplified node diagram of an unfolded neural network for use in motion control according to aspects of the present disclosure.

FIG. 1C is a simplified diagram of a convolutional neural network for use in motion control according to aspects of the present disclosure.

FIG. 1D is a block diagram of a method for training a neural network in development of motion control to aspects of the present disclosure.

FIG. 2A is a block diagram showing Q reinforcement learning implemented with neural networks and machine learning algorithms according to aspects of the present disclosure.

FIG. 2B is a block diagram showing Proximal Policy reinforcement learning implemented with neural networks and machine learning algorithms according to aspects of the present disclosure.

FIG. 3 is a diagram depicting motion control using sensor data and an integrated output that includes an integral and backlash threshold according to aspects of the present disclosure.

FIG. 4 is a diagram depicting motion control using sensor data and an integrated output that includes a second integral and a backlash threshold according to aspects of the present disclosure.

FIG. 5 is a diagram depicting motion control using sensor data, other data and an integrated output that includes a second integral and a backlash threshold for motion control according to aspects of the present disclosure.

FIG. 6 is a diagram showing model character rig in a simulation for training according to aspects of the present disclosure.

FIG. 7 is a diagram depicting the interactions and range of motions of leg portions of an example model according to aspects of the present disclosure.

FIG. 8 depicts an example of a surface in a simulated environment according to aspects of the present disclosure.

FIG. 9 is a system-level block diagram depicting a system implementing the training of neural networks and use of the motion control according to aspects of the present disclosure.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Although the following detailed description contains many specific details for the purposes of illustration, anyone of ordinary skill in the art will appreciate that many variations and alterations to the following details are within the scope of the invention. Accordingly, the exemplary embodiments of the invention described below are set forth without any loss of generality to, and without imposing limitations upon, the claimed invention.

Physics based animation requires a control scheme to generate joint actuator commands in such a way that it fulfills 4 goals at the same time: 1) approximately follow target animation; 2) preserve balance (don't fall down in case of walk, for example); 3) recover from external disturbances such as stumbling, external force, real/virtual model mismatch 4) account for real world constraints of physical robotic systems. According to aspects of the present disclosure, smooth life-like motions of a character may be obtained through training a NN to accept controlled mechanism/object sensor readings/observations as inputs and outputs either

first or second derivative of mechanism servo control commands. The commands in the case of first derivative are passed through external time integration. Output of time integration are compared to a threshold to account for motor backlash, the time integrated commands that meets or exceeds the threshold are fed a) back to NN and b) to controlled mechanism the controlled mechanism values are simulated using a model that account for surface stiffness and surface stiffness. In the case of second derivative described above pattern is repeated twice. First and second integrations of the NN output are performed. The result of the second integration may be compared to a threshold to account for motor backlash, results that meet or exceed the threshold go a) back to the NN and b) to the controlled mechanism, the controlled mechanism values may be simulated using a model that account for surface stiffness and surface damping.

In accordance with the foregoing, a generalized method for training a control input system may proceed as follows by taking an integral of an output value from a Motion Decision Neural Network for one or more movable joints to generate an integrated output value. A subsequent output value is then generated using a machine learning algorithm that includes a sensor value and a previous joint position. In some implementations, the integrated output value may be compared to a backlash threshold and the subsequent output is generated if the integrated output value does not at least meet the threshold. Joint positions, rigid body positions, surface stiffness or surface damping interaction with a simulated environment may be simulated based on an integral of the subsequent output value. The Motion Decision Neural Network may then be trained with the machine learning algorithm based upon at least a result of the simulation.

General Neural Network Training

According to aspects of the present disclosure, the control input scheme may use machine learning with neural networks (NN). The NNs may include one or more of several different types of neural networks and may have many different layers. By way of example and not by way of limitation the neural network may consist of one or multiple convolutional neural networks (CNN), recurrent neural networks (RNN) and/or dynamic neural networks (DNN). The Motion Decision Neural Network may be trained using the general training method disclosed herein.

FIG. 1A depicts the basic form of an RNN having a layer of nodes **120**, each of which is characterized by an activation function **S**, one input weight **U**, a recurrent hidden node transition weight **W**, and an output transition weight **V.** The activation function **S** may be any non-linear function known in the art and is not limited to the (hyperbolic tangent (tanh) function. For example, the activation function **S** may be a Sigmoid or ReLu function. Unlike other types of neural networks, RNNs have one set of activation functions and weights for the entire layer. As shown in FIG. 1B, the RNN may be considered as a series of nodes **120** having the same activation function moving through time **T** and **T+1**. Thus, the RNN maintains historical information by feeding the result from a previous time **T** to a current time **T+1**.

In some embodiments, a convolutional RNN may be used. Another type of RNN that may be used is a Long Short-Term Memory (LSTM) Neural Network which adds a memory block in a RNN node with input gate activation function, output gate activation function and forget gate activation function resulting in a gating memory that allows the network to retain some information for a longer period of time as described by Hochreiter & Schmidhuber "Long Short-term memory" Neural Computation 9(8):1735-1780 (1997), which is incorporated herein by reference.

FIG. 1C depicts an example layout of a convolution neural network such as a CRNN according to aspects of the present disclosure. In this depiction, the convolution neural network is generated for an input **132** with a size of 4 units in height and 4 units in width giving a total area of 16 units. The depicted convolutional neural network has a filter **133** size of 2 units in height and 2 units in width with a skip value of 1 and a channel **136** of size 9. For clarity in FIG. 1C only the connections **134** between the first column of channels and their filter windows is depicted. Aspects of the present disclosure, however, are not limited to such implementations. According to aspects of the present disclosure, the convolutional neural network may have any number of additional neural network node layers **131** and may include such layer types as additional convolutional layers, fully connected layers, pooling layers, max pooling layers, local contrast normalization layers, etc. of any size.

As seen in FIG. 1D Training a neural network (NN) begins with initialization of the weights of the NN at **141**. In general, the initial weights should be distributed randomly. For example,

an NN with a tanh activation function should have random values distributed between $-\frac{1}{\sqrt{n}}$ and $\frac{1}{\sqrt{n}}$ where n is the number of inputs to the node.

After initialization the activation function and optimizer is defined. The NN is then provided with a feature vector or input dataset at **142**. Each of the different feature vectors may be generated by the NN from inputs that have known labels. Similarly, the NN may be provided with feature vectors that correspond to inputs having known labeling or classification. The NN then predicts a label or classification for the feature or input at **143**. The predicted label or class is compared to the known label or class (also known as ground truth) and a loss function measures the total error between the predictions and ground truth over all the training samples at **144**. By way of example and not by way of limitation the loss function may be a cross entropy loss function, quadratic cost, triplet contrastive function, exponential cost, etc. Multiple different loss functions may be used depending on the purpose. By way of example and not by way of limitation, for training classifiers a cross entropy loss function may be used whereas for learning pre-trained embedding a triplet contrastive function may be employed. The NN is then optimized and trained, using the result of the loss function and using known methods of training for neural networks such as backpropagation with adaptive gradient descent etc., as indicated at **145**. In each training epoch, the optimizer tries to choose the model parameters (i.e., weights) that minimize the training loss function (i.e. total error). Data is partitioned into training, validation, and test samples.

During training, the Optimizer minimizes the loss function on the training samples. After each training epoch, the model is evaluated on the validation sample by computing the validation loss and accuracy. If there is no significant change, training can be stopped and the resulting trained model may be used to predict the labels of the test data.

Thus, the neural network may be trained from inputs having known labels or classifications to identify and classify those inputs. Similarly, a NN may be trained using the described method to generate a feature vector from inputs having a known label or classification. While the above discussion is relation to RNNs and CRNNS the discussions may be applied to NNs that do not include Recurrent or hidden layers.

Reinforcement Learning

According to aspects of the present disclosure, the NN training may include reinforcement learning. Reinforcement Learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. It may be used without a neural network but in situations where there are many possible actions a NN layout may be employed to capture the elements in reinforcement learning.

The goal of reinforcement learning is to choose the optimal action based on a current state. A reward mechanic is used to train the reinforcement model to make the correct decision based on the state. It should be noted that, the reinforcement model is not limited to Neural Network and may include for example and without limitation values in a table or spreadsheet.

FIG. 2A shows Q learning or discrete state reinforcement learning implemented with neural networks and machine learning algorithms **200**. The reinforcement learning algorithm as discussed above seeks to determine an action **203** from a current state **201**. Once an action is chosen, the effect of the action is determined **204** and a reward function **205** is applied based on how closely the effect achieved an optimal action or chosen goal. A motion decision NN **202** may be employed to determine the action **203** from the current state **201**. Additionally, the current state information **201** is updated at **206** with the effect **204** information and the NN can predict information based on the updated current state for a next action. In some embodiments, the NN **203** may be trained with a machine learning algorithm that uses Q-learning feedback as a value in the loss function for training the NN. For example and without limitation the loss function for the NN used reinforcement learning may be a sum of squares function. The sum of squares loss function with feedback is given by the equation here Q is the output of the NN:

$$Loss = \sum(feedback - Q)^2 \qquad \text{EQ.1}$$

In reinforcement learning one example of feedback may be given by the Q-learning equation:

$$feedback = r(i, a, j) + \lambda \max_b Q(j, b) \qquad \text{EQ.2}$$

Where the immediate reward is denoted by r(i, a, j,) where i is the current state, a is the action chosen at current state and j is the next state. The value of any state is given by the maximum value of Q value of actions in that state. Thus max Q (j,b) represents the expected reward from the best possible action taken at the following state. The quantity λ represents a future state discounting factor which serves to bias learning towards choosing immediate

rewards. In some embodiments, $\lambda = 1/(1+R)$ where R is a discounting rate chosen to suit the particular task being learned. In the case of applications using Q-learning the controls must be made discrete for applications involving physical simulations or robots.

Thus, in reinforcement learning after an action is taken a feedback is calculated and a loss function is calculated using the feedback. The model is then updated using the loss function and backpropagation with adaptive gradient descent. This is best for a system that has discrete positions for actions. Many robotic and animation systems do not include discrete controls thus a Proximal Policy Optimization training algorithm may be used to implement continuous stochastic controls for the system.

In other embodiments, a Proximal Policy Optimization training algorithm may be used. As shown in FIG. 2B the proximal Policy Optimization has an action **213** output space that is a continuous probability distribution as depicted by the bell curve in the action. Such an algorithm uses two networks: a Policy network (also called an Actor) to determine an action to take and an Advantage network (also called a Critic) to determine how good each action is, given the current state. Some implementations of the motion decision NNs **212** may include a policy subnetwork configured to provide a probability distributions for the action **213** that is optimal for achieving the desired effect **214** given the current state **211** and an advantage subnetwork for determining how good each action is given the current state **211.** In other words, the policy $\pi(s,a) = p(a|s)$ represents the conditional probability density function of selection action $a \in A$ in state $s \in S$ at each control step t; the network receives a state s_t and samples an action a_t from π. The simulated environment provides a new state $s_t` = s_{t+1}$ **216** and generating a reward $r_t$ **215** sampled from its dynamics $p(s`|s,a)$. The reward function is defined by the result of the transition between s_t to s_{t+1} by taking a corresponding action a_t: $r_t=R(s_t, a_t,s_{t+1})$. For a parameterized policy, $\pi_\theta(s,a)$ the goal of the agent is to learn the parameters θ, which maximize cumulative reward given by the equation:

$$J(\pi_\theta) = E[\sum_{t=0}^{T} \gamma^t r_t |\pi_\theta] \qquad \text{EQ. 3}$$

Where $\gamma \in [0,1]$a discounting factor and T is the training horizon. The gradient of the expected reward $\nabla_\theta J(\pi_\theta)$ can be determined using a policy gradient theory, which adjusts policy parameter θ to provide a direction of improvement according to the equation:

$$\nabla_\theta J(\pi_\theta) = \int_S d_\theta(s) \int_A \nabla_\theta \log\big(\pi_\theta(s,a)\big)\mathbb{A}(s,a)da\, ds \qquad \text{EQ. 4}$$

$d_\theta(s) = \int_S \sum_{t=0}^{T} \gamma^t p_0(s_0)(p(s_0 \rightarrow s|t, \pi_\theta) ds_0$ is a discounted state distribution, p_0 is an initial state distribution and $p(s_0 \rightarrow s|t, \pi_\theta)$models the likelihood of reaching state s by starting at s_0 and following the policy $\pi_\theta(s, a)$ for T steps. $\mathbb{A}(s, a)$ represents a general advantage function. There are many advantage functions for policy gradient based reinforcement learning and any suitable advantage function may be used with this function according to aspects of the present disclosure. One advantage function that may be used is a one-step temporal advantage function given by the equation:

$$\mathbb{A}(s_t, a_t) = r_t + \gamma V(s`_t) - V(s_t) \qquad \text{EQ. 5}$$

Where $V(s) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, \pi_\theta]$is a state-value function defined recursively through EQ. 6

$$V(s_t) = \mathbb{E}_{r_t, s`_t}[r_t + \gamma V(s`_t) | s_t, \pi_\theta] \qquad \text{EQ. 6}$$

Parameterized value function $V\phi(s)$, with parameters ϕ are learned iteratively similar to Q-learning as described above. The bellman loss function is minimized in this case according to the form:

$$L(\phi) = \mathbb{E}_{s_t, r_t, s`_t}\left[\frac{1}{2}\left(y_t - V_\phi(s_t)\right)^2\right], \; y_t = r_t + \gamma V_\phi(s`_t) \qquad \text{EQ.7}$$

π_θ and V_ϕ are trained in tandem using an actor critic framework. The action network may be biased toward exploration using a Gaussian distribution with a parameterized mean μ_θ and a fixed covariance matrix $\sum = \text{diag}\{\sigma_i^2\}$where σ_i is specified for each action parameter. Actions are sampled from the distribution by applying Gaussian noise to the mean action choice EQ. 8

$$a_t = \mu_\theta(s_t) + \mathcal{N}(0, \Sigma) \qquad \text{EQ. 8}$$

The Gradient for maximizing the action choice in EQ. 8 takes the form:

$$\nabla_\theta J(\mu_\theta) = \int_S d_\theta(s) \int_A \nabla_\theta \mu_\theta(s) \Sigma^{-1}\big(a - \mu_\theta(s)\big) \mathbb{A}(s, a) da \, ds \qquad \text{EQ.9}$$

The result of optimization of the gradient EQ. 9 is to shift the mean of the action distribution towards actions that lead to higher expected rewards and away from lower expected rewards. For additional information see Peng et al. “Learning Locomotion Skills Using Deep RL: Does Choice of Action Space Matter?” SCA`17 July 28, 2017.

Application to Movement

According to aspects of the present disclosure, the NN may be trained with a machine-learning algorithm to mimic realistic movement. The training set may be for example and without limitation, a time sequence of positions in space, directions, and/or orientations of a preselected subset of controlled object body parts. It is up to the machine learning algorithm to prepare a NN which is capable of changing joint angles in such a way that the controlled object exhibits a desired behavior and preserves balance at the same time. By way of example and not by way of limitation, time sequence of positions in space, directions, and/or orientations may be generated by motion capture of real movement, hand animation using motion capture dolls, hand animation using a virtual models, or any other method of capturing a set of realistic movements. In some embodiments, the training may use a reward function that uses misalignment errors of various raw and/or integral parameters which evolve the reward function towards a desired movement.

The State **201 or 211** may be a feature transformation $\Phi(q, v, \varepsilon)$ where ε is an integral input taken from the integral of velocity with respect to time$\varepsilon = (\int v\, dt)$ generated as an output of the NN. According to some alternative aspects of the present disclosure, the feature transformation $\Phi(q, v, \varepsilon, i)$ may include the second integral of acceleration with respect to time $i = (\iint A\, dt)$. The transformation extracts a set of features from inputs to place them in a form compatible with the variable of the model being trained. In training it useful to include target reference motions $\Phi(\hat{q}, \hat{v}, \hat{\varepsilon})$ thus giving a combined state of $s_t = \Phi(q, v, \varepsilon), \Phi(\hat{q}, \hat{v}, \hat{\varepsilon})$. Alternatively, the quaternion link locations for the state and reference motion may be used as discussed below.

The reward function may consist of a weighted sum of terms that encourage the policy to follow the reference motion:

$$r_{total} = r_{link} + (-r_{collision}) + r_{ground} + (-r_{limit}) \qquad \text{EQ. 10}$$

Where w is a weight for the given term and r reference term.

As shown in FIG. 6 each joint **601** with rigid body **606** may be considered a link, a series of links, or a chain that may form an agent or character rig model **602**. The L_2 quaternion distance between a reference link location and an agent link location generated by the NN at time step t is subtracted from the L_2 quaternion distance between a reference link location and

an agent link location generated by the NN at timestep t-1. This provides a differential error between the target link locations and the agent link locations that rewards the link for moving in the correct direction while penalizing it for moving in the wrong direction. The error between the agent pose and target pose is a weighted sum of the individual link orientations errors. The weights w_{link} are chosen such that that the first link (including joint **601** and rigid body **606**) in the chain is weighted higher than the last link **603** in the chain. As shown the first link in the chain includes both a joint and a rigid body while the last link only includes a rigid body. In many cases the last link in the chain may be as specialized tool such as, without limitation, a hand, grabber, foot, or other interaction device. This pushes the system to focus on aligning the root links before the end links during training. The same differential approach may be used for link velocities but with only a small velocity coefficient v_{coeff} of the velocity error mixed with the distance error. The total differential reward is calculated as the sum of all individual link rewards.

$$r_{link} = \sum_{l \in links} w_{linkl} * \left(dist_{link}(t-1) - dist_{link}(t)\right) + v_{coeff} * \left(vel_{link}(t-1) - vel_{link}(t)\right) \quad EQ.\ 11$$

where w_{link} is the individual link weight and v_{coeff} is a small non-negative constant. The quantity $dist_{link}$ is the quaternion distance between link orientations which will now be described. In the case of application of real movement models taken from, for example and without limitation, motion capture or video may have a mismatch between the degrees of freedom in joints of the real movement model compared to the degrees of freedom of the joints of the agent **602**. For example and without limitation a real human's joints may have three degrees of freedom whereas the agent's joints **601** may only have two degrees of freedom. Each link unit's axis is defined as a unit vector in the links local reference frame. For the quaternion distance metric, q_a and q_t represent the agent and link orientation quaternions respectively. The difference between orientations is thus provided by:

$$\Delta q = q_a^* * q_t \quad EQ.\ 12$$

Where the quaternion distance between links d_{link} is provided by the equation:

$$d_{link} = 2 * \sin^{-1}\left(\Delta q_x^2 + \Delta q_y^2 + \Delta q_z^2\right) \quad EQ.\ 13$$

The angle between link axes is computed by the following. Let $\overrightarrow{e_a}$ and $\overrightarrow{e_t}$ be the agent and target link axes converted to the world reference frame. The Axis distance is then computed by:

$$d_{axis} = \cos^{-1}\left(e_{a,x} * e_{t,x} + e_{a,y} * e_{t,y} + e_{a,z} * e_{t,z}\right) \quad \text{EQ. 14}$$

This introduced unit axis distance allows links to be mapped where there is an insufficient number of degrees of freedom.

Returning to the reward function of equation 10. The ($-c_{ollision}$) term is a penalty for self-collisions. As seen in Fig. 6 each link **604** may take up a volumetric space. The penalty may be applied whenever the volumetric space of link **604** comes into contact with the volumetric space of another link **605.**

The term r_{ground} may be applied based on foot ground interactions between the agent and the world. When processing training set data, an additional field is added to each link at each time step indicating if the link is on or off the ground. This information is used in the reward function to give a positive reward if the foot pressure sensor **610** is over a threshold and the foot is record as on the ground or alternatively if the pressure sensor is below a threshold and the foot is recorded to be off the ground.

$$r_P = \begin{cases} r_P \; if \; gnd_{on} \; and \; f_p > P_{th} \\ 0 \; if \; gnd_{on} \; and \; f_p < P_{th} \\ r_P \; if \; gnd_{off} \; and \; f_p < P_{th} \\ 0 \; if \; gnd_{off} \; and \; f_p > P_{th} \end{cases} \quad \text{EQ. 15}$$

Where gnd_{on} and gnd_{off} indicate the foot ground state, f_p represents a foot pressure sensor **710** reading and P_{th} is a foot pressure threshold.

An additional positive may be given when the foot is on the ground. The reward is proportional to the angle between the foot local vertical axis and the world up vector.

$$\overrightarrow{e_z} = (0,0,1)^T$$

$$\overrightarrow{e_{z,world}} = Q_{foot} * \overrightarrow{e_z} * Q^*_{foot}$$

$$\alpha = \cos^{-1}\left(\overrightarrow{e_{z,world}}, [z]\right) \quad \text{EQ. 16}$$

$$r_{flat} = K_1 * (K_2 - \alpha)$$

Where $\overrightarrow{e_z}$ indicates the vertical axis Q_{foot} is the foot orientation quaternion, $\overrightarrow{e_{z,world}}$ is the foot up vector in the world reference frame and K_1 and K_2 are constants. The complete ground reward is calculated as:

$$r_{ground} = r_P + r_{flat} \qquad \text{EQ. 17}$$

The ($-r_{limit}$) term provides a penalty on a per joint basis if a target joint position is outside the physical limits of the joint this penalty pushes the training process to avoid entering areas where the control policy is unable to affect the agent state.

$$r_{limit} = \begin{cases} k_{joint} * \left(C_{joint} + \left(\alpha_i - \alpha_{i,lim}^{up}\right)\right) & \alpha_i > \alpha_{i,lim}^{up} \\ k_{joint} * \left(C_{joint} + \left(\alpha_{i,lim}^{low} - \alpha_i\right)\right) & \alpha_i < \alpha_{i,lim}^{low} \end{cases} \qquad \text{EQ.18}$$

Where C_{joint} and k_{joint} are constants that define how sharply the penalty increases, $\alpha_i$is the i-th joint position, $\alpha_{i,lim}^{up}$is the i-th joint upper limit and $\alpha_{i,lim}^{low}$is the i-th joint lower limit.

Iterations of the network may be trained updating algorithm to apply updates to the state as soon as possible based on sample rates of the inputs. For this purpose, input includes as many observations as possible. All available sensor readings are fed into NN. Some sensor readings are also preprocessed. For example, accelerometer and gyroscope readings are fed both as-is and fused into attitude and gravity direction in robot's ref. frame. Preprocessed readings are also fed into NN.

Improved Motion control with NNs

One major problem with NN control is choosing which information to provide to the NN as input, which is enough to restore dynamical state of the system at each moment in time. As depicted in FIG. 3 and FIG. 4, as a matter of feedback control output integral and output second integral in case of FIG. 4 are fed back into NN only after being compared and at least meeting a backlash threshold. As discussed above, prior trained NN for movement produced jittery, unrealistic movements. Thus according to aspects of the present disclosure smooth realistic movement is achieved by generating an integral of an output of the NN and using input information from a movable joint state input parameters in a NN.

FIG. 3 depicts training motion control according to aspects of the present disclosure. A character rig **301** within a simulation, which by way of example and not by way of limitation may be a character model, skeleton, robot or robotic apparatus that has at least once movable joint **302.** As shown the character rig **301** has multiple movable joints as indicated by the black circles. The movable joint **302** may be a motor actuated hinge, ball joint, or any other connection between two rigid bodies that allows a defined range of controlled relative movement between the two rigid bodies. As discussed above in relation to FIG. 7 the rigid bodies of the link may have volume of space **704** that they occupy and that may collide with the space occupied by any other rigid body or movable joint. The movable joint may be connected to a sensor **303**, which is configured to generate information about the state of the movable joint, referred to herein as sensor values. This sensor **303** within the simulation may be configured to deliver information similar to information generated by sensors for physical robots which may include for example and without limitation, encoders, potentiometers, linear variable differential transformers, pressure sensors, gyroscopes, gravimeters, accelerometers, resolvers, velocity, or speed sensor. The sensor values for such sensors would correspond to the outputs of such sensors or information derived therefrom. Examples of sensor values from sensors on a robot include, but are not limited to a joint position, a joint velocity, a joint torque, a robot orientation, a robot linear velocity, a robot angular velocity, a foot contact point, a foot pressure, or two or more of these. For virtual characters, the sensor **303** may be virtual sensors and the sensor values may simply include data, e.g., position, velocity, acceleration data, related to the state of the movable joint. Examples of sensor values from a robot simulation include, but are not limited to a joint position, a joint velocity, a joint torque, a model orientation, a model linear velocity, a model angular velocity, a foot contact point, a foot pressure, or two or more of these. Position Data from the controller or virtual monitor may be passed **306** to the motion decision neural network **307** and used as state data during reinforcement learning.

As shown in FIG. 3, input information such as from the sensor **303** is passed directly **306** to the NN **307**. The integrator **305** receives an output value **308** from the NN **307**. During training the integrator **305** provides the integrated output to the motor backlash threshold comparator **310.** When not training, the NN **307,** the integrator **305** provides the integrated output to the movable joint **302** and as integral output feedback to the NN **307.**

The motor backlash comparator **310** used during training, compares the integrated output to a motor backlash threshold. When the integrated output meets or exceeds the motor backlash threshold the identical integrated output value is passed through to the movable joint **302** and as feedback **304** to the NN **307**. If the integrated output **304** does not at least meet the motor backlash threshold the integrated output value is not passed back to NN **307** or the movable joint **302**.

The motor backlash comparator **310** simulates real world constraints of physical motors. Physical motors require varying levels of force to move the limb position depending upon various factors for example and without limitation wear, temperature, motor design, gearing etc. The motor backlash comparator compensates for the backlash problem by training the NN **307** to over shoot the desired joint position and thus moving the joint in a way that accounts for motor backlash. The motor backlash threshold may be randomized during training. The motor backlash threshold may be randomized at the start of training or after each round of NN training.

Alternatively, the motor backlash threshold may be based on other factors in the simulation. For example and without limitation, these factors may include: time dependence of wear on the joint and motor may be modeled by having the motor backlash threshold increase with time. More granular wear models may also be applied to the backlash threshold that replicate the non-linearity of component wear. In a simple example the backlash threshold may be changed depending on the number of times the joint passes through or remains on a position. Alternatively, the threshold may be changed based on the amount of time the joint spends at a position. Training the NN may require randomization of the backlash threshold to reduce NN overfitting. The motor backlash threshold may be randomized in non-linear ways to simulate non-linear wear on the joints and motor. For example a non-uniform growth equation such as;

$$B_{th} = Ae^{(-(x-\mu)^2/\sigma^2)} \qquad \text{EQ.19}$$

Where A, μ and σ are randomized to simulate non-uniform joint and motor backlash. Alternatively A, μ and σmay be dependent upon a joint angle use or joint position use. The dependency of A, μ and σmay be probabilistic so that angles or positions that are used frequently have a higher chance of getting an increased motor backlash threshold. While EQ 19 describes one example on an equation for non-uniform in yet another example, a heat map may be generated to describe wear on different areas of the joint and on different surfaces. The heat map may describe areas with more use as hotter and areas with less use as cooler,

random noise may then be applied to reduce over-fitting. The heat map may be correlated with the backlash threshold so that areas of high use receive a higher threshold value than areas of low use and the threshold includes some random noise values to reduce over-fitting.

The motion decision NN **307** as discussed above, may be trained iteratively using machine learning algorithms that include reinforcement learning techniques such as policy learning. Q-leaning may be applied with discretized controls additionally any other machine learning technique suitable for the task may be used with control scheme provided according to aspects of the present disclosure. The motion decision NNs **307** may include additional subnetworks to generate embeddings or otherwise process state data. The motion decision NNs **307** may be configured to output one or more types of information to the movable joint or a motor/actuator controlling the movable joint.

The movable joint **302** may move based on the information output **308** by the motion decision NN **307** and this change may be detected by the sensor **303.** During training, a simulation virtually replicates the movement of the movable joint **302** based on the information output **308** by the motion decision network with simulated physical constraints as discussed in the next section. From the simulation, the movement change in the movable joint is reported as a sensor output **303.** Subsequently the new position and acceleration information may be used by the NN in a repetition of the process described above. This cycle may continue until a goal is achieved.

Here, an improvement to smooth movement imparted with the movable joint is achieved with the addition of integrated output **304** feedback calculated at the integrator **305** from the output **308** of the NN **307.** One explanation for the smoothing effect created according to aspects of the present disclosure may be that the integral of the step function is a continuous function and the discontinuous controls output by the NN are converted to continuous actuator controls after going through the integrator.

As shown in FIG. 4, other integrated output feedback may be used as an input to the NN to create smoother movement. According to alternative aspects of the present disclosure, smooth movement may be created with a motion decision NN **404** using state information that includes a second integrated output **403** value generated from the second integrator **405**. The second integrated output is provided to a motor backlash comparator **409** which, as discussed above, may pass the second integrated output value to the NN or ignore the second

integrated output depending on whether the value meets or exceeds the motor backlash threshold (B_{th}). The second integrator **405** is configured to take the output of a first integrator **402**. The first integrator **402** receives an output from the motion decision NN **404** and provides the first integral of that output.

FIG. 5 shows another alternative embodiment of the smooth motion, motion control using NNs according to aspects of the present disclosure. In this embodiment, the other information **505** is generated by other sensors and passed **503** to the motion decision NNs **502.** The other information may be without limitation, visual information, motion information or sound information that indicates the presence of a person or other important object. The addition of other information aids allows the motion decision NNs **502** to produce more situationally appropriate movements by providing more information to use in motion decision**.** The second integrator integrates the first integrated value and passes it to the motor backlash comparator **510**. The second integrated output is compared to the motor backlash threshold and if it does not at least meet the threshold the value is discarded and is not passed to joint or the motion decision NN **502**. If the first integrated output meets or exceeds the motor backlash threshold it is passed to the motion decision NN **502** and movable joint.

It should be noted that the controller or variable monitor according to aspects of the present disclosure may also detect torque from the movable joint and the torque information may also be provided to the NNs. The NNs may also be configured to produce torque information.

Control inputs obtained as discussed herein may be used for control of physical robots as well as for control of robot simulations, e.g., in video games, cloud video games, or game development engines, such as Unity3D from Unity Technologies of San Francisco, California, Lumberyard from Amazon Game Studios of Seattle, Washington, and Unreal Engine by Epic Games of Cary, North Carolina.

SIMULATION

FIG. 6 as discussed above shows a model character rig in a simulator. The model may be thought of as links in a chain with rigid bodies **606** connected by movable joints **601**. Here, the links in the chains are arranged to model a humanoid style robot **602**. The model **602** also include joints that simulate multiple revolute joints of a robot, the joint here (for example joint **601**) include rotation joint portion (represented by the smaller bold rectangle) and hinge portion (represented by the circle). The arrangement of the links allows the model to replicate

human-like movement by having different types of joints in different areas of the model. As shown in the pelvis of the model there are rotation joints without hinge joints. A hip joint **613** of the model includes a hinge with a rotation joint. Similarly, a knee joint **614** of the model includes a hinge without a rotation joint. The end links of the chain, such as hand link **603** may include interaction devices; feet links may include sensor devices for balance such as pressure sensors **610.** Within the simulation, each link is associated with dynamic properties. These dynamic properties include mass and inertia tensors. Links are considered to be interacting when the associated collision volumes **604, 605** intersect in space. Normal reaction and dry friction forces are applied to rigid bodies and a simulated environment.

FIG. 7 depicts the interactions and range of motions of the leg portions of one example of the model according to aspects of the present the present disclosure. The links of the simulator may be modeled with rigid bodies connected via one or more revolute joints. The joints have a finite range of motion limited by collision with other rigid bodies or the joint design. As shown each link may include one or more joints. For example and without limitation a pelvis link may include a pelvis rigid body **701** and a pelvis joint **702**. A thigh link may include a thigh rigid body **703** and knee joint **704.** A shin link may include a shin rigid body **705** and ankle joint **706** and a foot link may include a foot rigid body **707** and one or more foot pressure sensors **708.** The one or more pressure sensors **708** may be used to describe the model's interaction with the simulated environment **816**. Each joint may have a dead zone **710** which simulates backlash and which must be overcome before the rigid body may change position. Additionally the joints have a range of motion **709** limited by the design of the join itself and collisions with other rigid bodies in the chain. The rigid bodies also interact with the simulated environment **716**. The dotted lines represent a position of the leg **715** in which it is extended and interacting with the simulated environment **716.** Here the original state of the simulated environment **716** is shown as a solid flat line and the deformation of the environment by interaction with the collision volume of the foot rigid body **707** is show by the dashed line **713.** The mutual penetration depth of the collision volumes **712** and the dry friction forces defines the reaction force with rigid bodies and simulated environment. Note here that the simulated environment **711** is much more plastic than the foot rigid body **707** as such its depth of penetration **712** is much greater. Further, note that as result of the plastic deformation of the simulated environment **713** the angle of the **711** is different than if it were on a non-deformed flat environment **716**. The dry friction force is given by:

$$F = -\frac{v}{|v|} k F_{react} \quad \text{EQ. 20}$$

Where F is the force of friction, v is the point relative velocity projection onto the surface and F_{react} is the absolute value of force applied to the surface through the point.

The mutual penetration depth of the collision includes complex real-world constraints such as surface stiffness and surface damping. The stiffness of a surface may be measured by any number of different measurements including, Shore Durometer, Young's modulus, the Brinell scale, Rockwell hardness, Vickers hardness, or any other measurement which describes the elasticity of a material or force required to deform a material. The surface stiffness of the environment may be modeled in the simulation to replicate different surfaces a robot or other device may encounter during operation. An additional factor depth accounts for how deep into a material a limb or rigid body may penetrate.

A related, but different, constraint than surface stiffness is surface damping. Surface damping is related to the time dependence of surface deformation, whereas surface stiffness describes the force required to deform the material. Damping affects the time derivative of depth in other words damping affects how fast a limb or rigid body deforms a material when the two are in contact. Additionally, damping may be non-constant with time, meaning that sometimes a surface may deform slowly initially but then deformation quickly accelerates as more force is applied. For example, a surface such as clay may have a dry hardened outer crust that slowly deforms but once broken the surface deforms quickly as the underlying mud and dirt is easily displaced.

The mutual penetration depth of the collision for example and without limitation the collision of a robot foot on a clay surface may be partially modeled by:

$$F_{pen} = E * D + d_k * Ddt$$

Where F_{pen} is the force of the mutual penetration depth of the collision, E is the surface stiffness, which may be provided by the stiffness of the material, D is the penetration depth, d_k is the surface damping and Ddt is the time derivative of depth.

During training of the NN the variables of dry friction force, surface stiffness and/or surface damping may be randomized or otherwise modified. The use of randomized constraint values may train the NN to act on surfaces differently depending on the type of material for

example, the NN may output different control values for soft surfaces where foot pose may change over time due to surface deformation under load compared to a hard non-pliable surface.

FIG. 8 depicts an example of a surface **801** in a simulated environment according to aspects of the present disclosure. As shown, the surface **801** may include areas **802, 803, 804,** with different, penetration depth, stiffness, dry friction force, and/or damping coefficients (hereinafter referred to as constraints) than other areas **805**. As shown areas with different shading patterns represent areas having a different penetration depth, stiffness, dry friction and/or damping. There may be defined borders **806** between different areas having different constraints. The constraints may be constant within the borders **806** of each area but vary randomly between areas. The values of the constraints may be randomized using structured noise or simple Gaussian noise. For example and without limitation constraint values of each of the areas may be constant and the constraint values may be generated using Gaussian noise or uniformly distributed noise. In another example, the constraints value of each of the areas is not constant and is generated using coherent noise. In yet another example, the constraints value of a first subset of the areas is not constant and is generated using coherent noise and the constraint values of a second subset of the areas is generated using Gaussian noise or uniformly distributed noise.

The shapes of the different areas defined by their borders **806** may be randomized or, alternatively, the shapes of the areas may be generated to resemble real world objects such as rugs, carpet to hardwood transitions, tiles on tile floors, etc. The different areas having different constraints may be generated using Gaussian noise. Alternatively, the areas having different constraints may be generated using structured noise or coherent noise. Coherent noise may be for example and without limitation outlines having a recognizable outlines with noise added to randomize the borders of the recognizable outlines without losing the overall recognizable shape. Simplex or Perlin coherent noise may be used to pattern floor properties distribution while training the robot controller. The boundary shape may be governed by the initial coherent noise before applying a transformation. The overall shape of an area may be defined by the number of octaves, lacunarity and time persistence of the coherent noise frequency distribution of the coherent noise. Lacunarity refers to a measure of gaps in the coherent noise distribution, where a distribution having more or larger gaps generally has higher lacunarity. Beyond being an intuitive measure of gappiness, lacunarity can quantify

additional features of patterns such as heterogeneity (i.e., the degree to which certain statistical properties of any part of the coherent noise distribution are the same as for any other part). Time Persistence, refers to a degree to which a prediction of a future value can be determined from extrapolation of a trend observed in past values.

In the simulation, the joints and rigid bodies may be associated with one or more sensor values, which replicate the position and type of sensors in a real robot. During training and simulation, the sensors provide information about the simulation to the NN. The virtual sensors may be for example and without limitation inertial measurement units (IMU), joint angle sensors, foot pressure sensors, clocks, etc. These different sensors may provide reading to the NN that may be used with movement training sets combined with simulated bias and noise during training.

System

FIG. 9 depicts a system for physics based character animation using NNs with reinforcement learning like that shown in Figures throughout the application for example Figs. 2, 3, 4 and 5. The system may include a computing device **900** coupled to a user input device **902**. The user input device **902** may be a controller, touch screen, microphone, keyboard, mouse, joystick or other device that allows the user to input information including sound data in to the system. The user input device may be coupled to a haptic feedback device **921.** The haptic feedback device may be for example a vibration motor, force feedback system, ultrasonic feedback system, or air pressure feedback system. Additionally the system may include a controller **901** for a movable joint for example and without limitation, the controller may control a motor or actuator for a joint.

The computing device **900** may include one or more processor units **903**, which may be configured according to well-known architectures, such as, e.g., single-core, dual-core, quad-core, multi-core, processor-coprocessor, cell processor, and the like. The computing device may also include one or more memory units **904** (e.g., random access memory (RAM), dynamic random access memory (DRAM), read-only memory (ROM), and the like).

The processor unit **903** may execute one or more programs, portions of which may be stored in the memory **904** and the processor **903** may be operatively coupled to the memory, e.g., by accessing the memory via a data bus **905**. The programs may include machine learning algorithms **921** configured to adjust the weights and transition values of NNs **910** as

discussed above. Additionally, the Memory **904** may store integrated outputs **908** that may be used, as input to the NNs **910** as state data additionally the integrated outputs may be stored database **922** for later training iterations. Sensor data **909** generated from the sensor may be stored in the Memory **904** and used as state data with the NNs **910** where the sensor data is either from a real sensor or a virtual model in a simulation. The memory **904** may also store a database **922**. The database may contain other information such as information associated with creation and movement of the virtual character rig in a simulation. Such information may include, but is not limited to: motor backlash thresholds, friction coefficients, stiffness values, penetration depths, damping coefficients, reference movement information and movement simulations. Additionally, the database **922** may be used during generation of the error **908** to store integral values of Control data **909** according to FIGs 3, 4 or 5. Simulation data **923** including physical properties of materials of the virtual character rigs, simulated environments and instructions for simulating interactions between virtual characters and environments may also be stored in memory **904.** The database **922,** sensor data **909** integrated outputs **908** and machine-learning algorithms **921** may be stored as data **918** or programs **917** in the Mass Store **915** or at a server coupled to the Network **920** accessed through the network interface **914.**

Control data and the error, may be stored as data **918** in the Mass Store **915**. The processor unit **903** is further configured to execute one or more programs **917** stored in the mass store **915** or in memory **904** which cause processor to carry out the one or more of the methods described above.

The computing device **900** may also include well-known support circuits **906**, such as input/output (I/O) circuits **907**, power supplies (P/S) **911**, a clock (CLK) **912**, and cache **913**, which may communicate with other components of the system, e.g., via the bus **905**. The computing device may include a network interface **914**. The processor unit **903** and network interface **914** may be configured to implement a local area network (LAN) or personal area network (PAN), via a suitable network protocol, e.g., Bluetooth, for a PAN. The computing device may optionally include a mass storage device **915** such as a disk drive, CD-ROM drive, tape drive, flash memory, or the like, and the mass storage device may store programs and/or data. The computing device may also include a user interface **916** to facilitate interaction between the system and a user. The user interface may include a monitor,

Television screen, speakers, headphones or other devices that communicate information to the user.

The computing device **900** may include a network interface **914** to facilitate communication via an electronic communications network **920**. The network interface **914** may be configured to implement wired or wireless communication over local area networks and wide area networks such as the Internet. The device **900** may send and receive data and/or requests for files via one or more message packets over the network **620**. Message packets sent over the network **920** may temporarily be stored in a buffer in memory **904**. The control data **909** and NNs **910** may be available through the network **920** and stored partially in memory **904** for use.

While the above is a complete description of the preferred embodiment of the present invention, it is possible to use various alternatives, modifications and equivalents. Therefore, the scope of the present invention should be determined not with reference to the above description but should, instead, be determined with reference to the appended claims, along with their full scope of equivalents. Any feature described herein, whether preferred or not, may be combined with any other feature described herein, whether preferred or not. In the claims that follow, the indefinite article "A", or "An" refers to a quantity of one or more of the item following the article, except where expressly stated otherwise. The appended claims are not to be interpreted as including means-plus-function limitations, unless such a limitation is explicitly recited in a given claim using the phrase "means for."

WHAT IS CLAIMED IS:

1. A method for training a control input system comprising:
a) taking an integral of an output value from a Motion Decision Neural Network for one or more movable joints to generate an integrated output value;
b) generating a subsequent output value using a machine learning algorithm that includes a sensor value and a previous joint position if the integrated output value does not at least meet the threshold;
c) simulating surface stiffness interactions with at least a simulated environment, a rigid body position and a position of the one or more movable joints based on an integral of the subsequent output value; and
d) training the Motion Decision Neural Network with the machine learning algorithm based upon at least a result of the simulation of the simulated environment and position of the one or more movable joints.

2. The method of claim 1 wherein simulating surface stiffness interactions includes a simulating penetration depth of a rigid body.

3. The method of claim 2 wherein the penetration depth is randomized.

4. The method of claim 1 wherein a surface stiffness value of the simulated environment is randomized.

5. The method of claim 1 further comprising repeating steps a) through d).

6. The method of claim 5 wherein the surface stiffness is randomized for each repetition.

7. The method of claim 1 wherein simulating surface stiffness interactions includes simulating dry friction forces.

8. The method of claim 1 wherein simulating surface stiffness interactions includes surface stiffness values of at least the simulated environment modeled as areas on a surface where each area has an associated surface stiffness value.

9. The method of claim 8 wherein the surface stiffness value of each area is randomly varied.

10. The method of claim 8 wherein the surface stiffness value of each of areas is not constant and is generated using coherent noise.

11. The method of claim 8 wherein the surface stiffness value of each of the areas is constant and is generated using Gaussian noise or uniformly distributed noise.

12. The method of claim 8 wherein the surface stiffness value of a first subset of the areas is not constant and is generated using coherent noise constant and wherein the surface

stiffness of a second subset of the areas is generated using Gaussian noise or uniformly distributed noise.

13. The method of claim 8 wherein a shape of the areas is randomized.

14. The method of claim 8 wherein shapes of the areas are generated using coherent noise having at least one area shape based on a real object with noise added to the shape of the area.

15. The method of claim 8 wherein a shape of the areas is defined using coherent noise including the number of octaves, lacunarity, time persistence of the coherent noise or frequency distribution of the coherent noise.

16. The method of claim 1 wherein values of the surface stiffness of at least the simulated environment are modeled as a simplex or Perlin distribution of values on a surface.

17. A input control system comprising:

a processor;

a memory coupled to the processor;

non-transitory instruction embedded in the memory that when executed by the processor cause the processor to carry out the method for training control input comprising:

a) taking an integral of an output value from a Motion Decision Neural Network for one or more simulated movable joints to generate an integrated output value;

b) generating a subsequent output value using a machine learning algorithm that includes a simulated sensor value and a previous joint position if the integrated output value does not at least meet the threshold;

c) simulating surface stiffness interactions with at least a simulated environment, a rigid body position and a position of the one or more simulated movable joints based on an integral of the subsequent output value; and

d) training the Motion Decision Neural Network with the machine learning algorithm based upon at least a result of the simulation of the simulated environment and position of the one or more movable joints.

18. The system of claim 17 wherein simulating surface stiffness interactions includes a simulating penetration depth of a rigid body.

19. The system of claim 18 wherein the penetration depth is randomized.

20. The system of claim 17 wherein a surface stiffness value of the simulated environment is randomized.

21. The system of claim 17 further comprising repeating steps a) through d) wherein the surface stiffness is randomized for each repetition.

22. A computer readable medium having non-transitory instruction embedded thereon that when executed cause a computer to carry out the method for training a control input system comprising:

a) taking an integral of an output value from a Motion Decision Neural Network for one or more movable joints to generate an integrated output value;

b) generating a subsequent output value using a machine learning algorithm that includes a sensor value and a previous joint position if the integrated output value does not at least meet the threshold;

c) simulating surface stiffness interactions with at least a simulated environment, a rigid body position and a position of the one or more movable joints based on an integral of the subsequent output value; and

d) training the Motion Decision Neural Network with the machine learning algorithm based upon at least a result of the simulation of the simulated environment and position of the one or more movable joints.

FIG. 1A

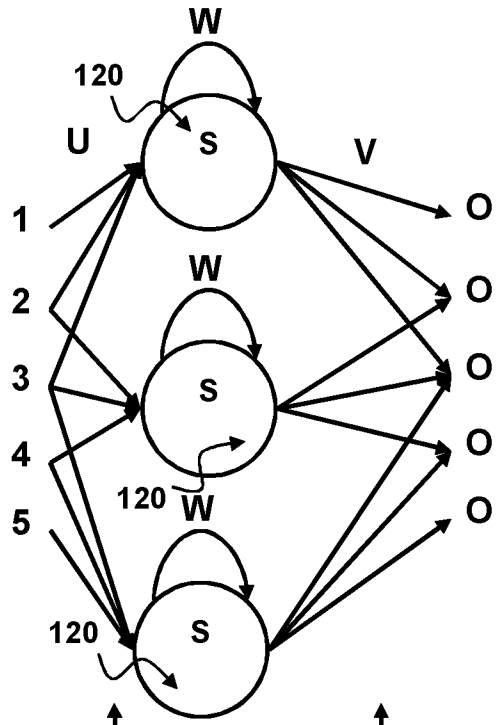


FIG. 1B

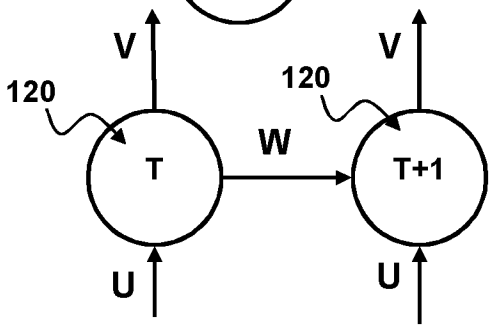
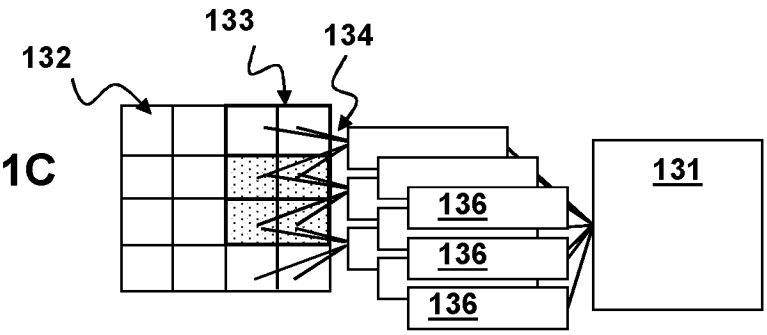


FIG. 1C



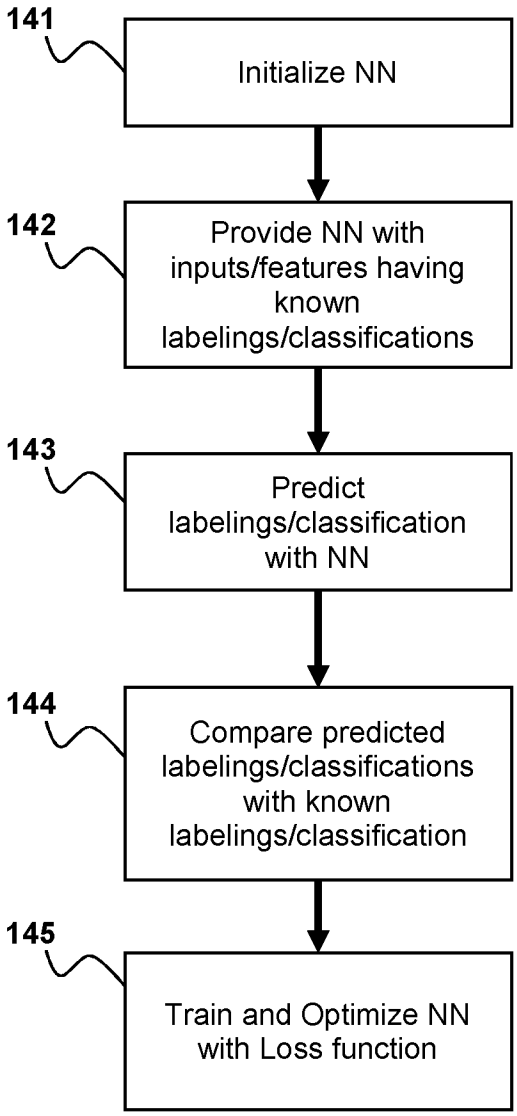

141
Initialize NN
142
Provide NN with inputs/features having known labelings/classifications
143
Predict labelings/classification with NN
144
Compare predicted labelings/classifications with known labelings/classification
145
Train and Optimize NN with Loss function

FIG. 1D

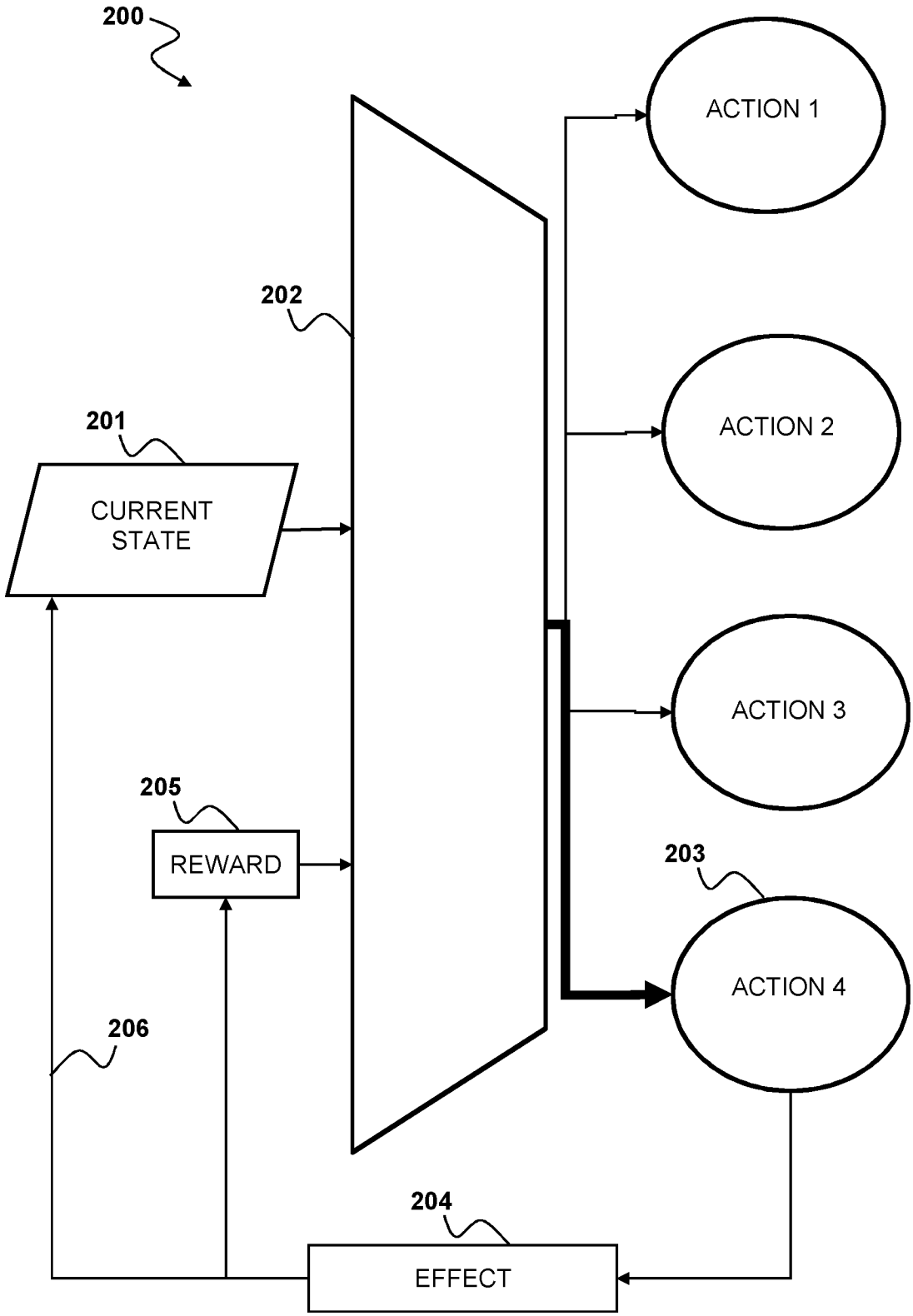

200
ACTION 1
202
ACTION 2
201
CURRENT STATE
ACTION 3
205
REWARD
203
ACTION 4
206
204
EFFECT

FIG. 2A

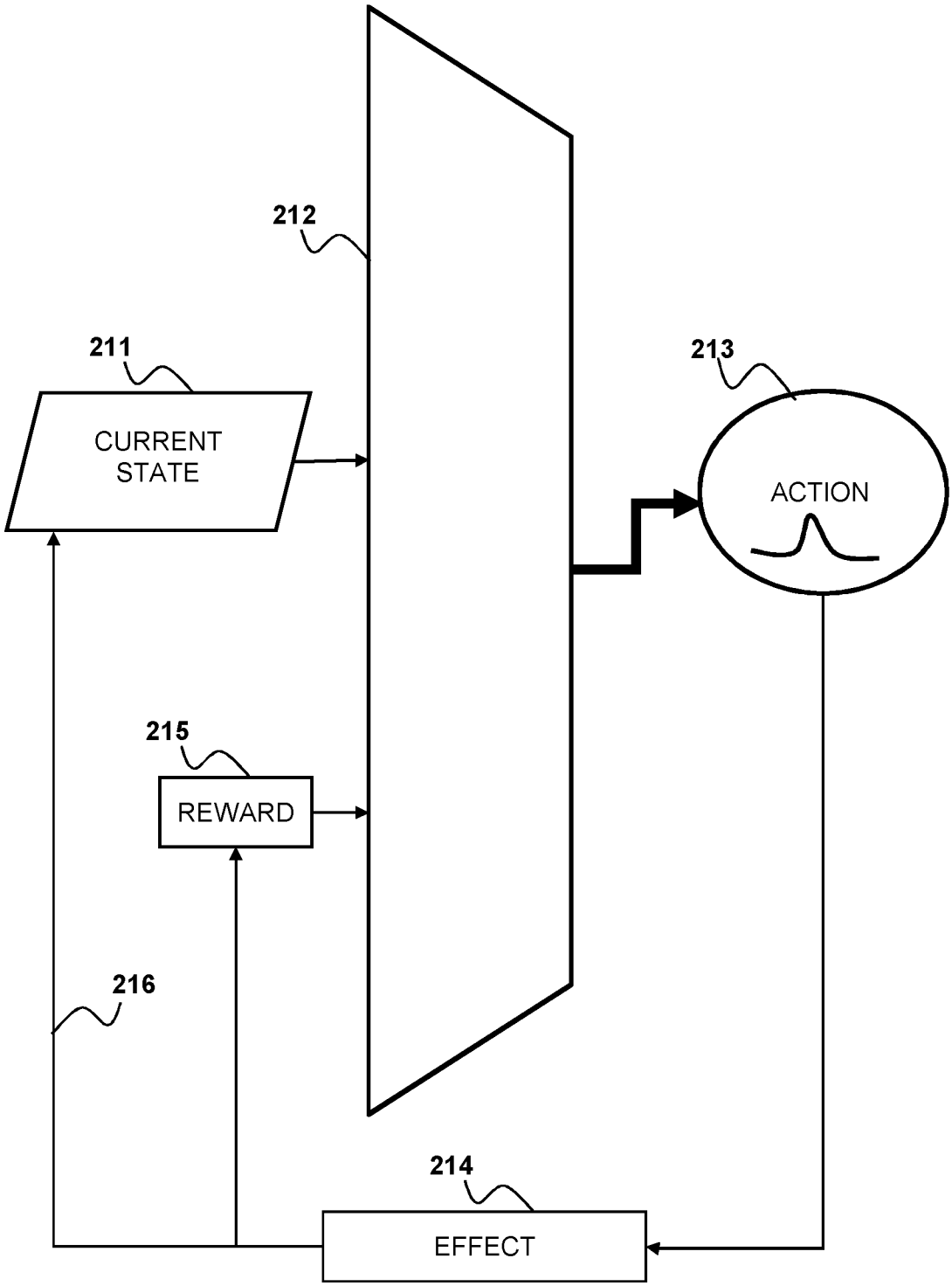


FIG. 2B

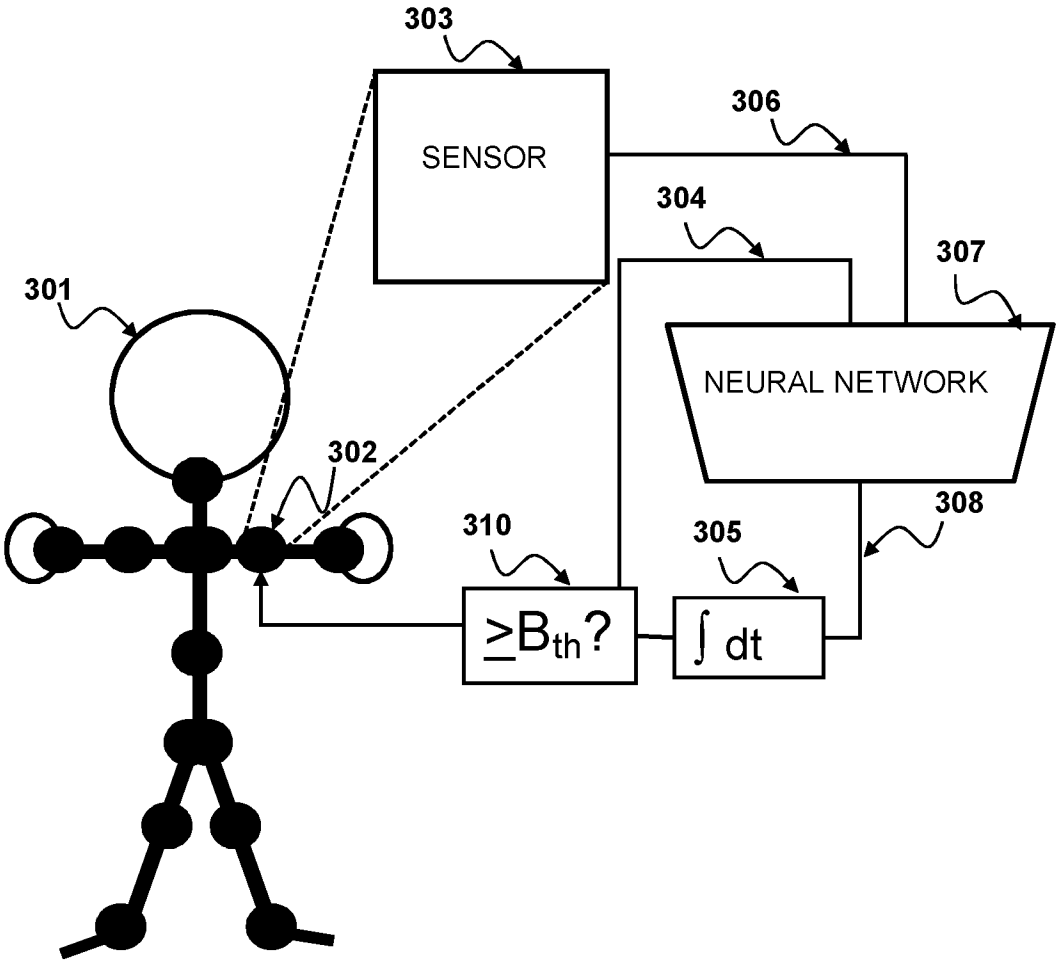

303
SENSOR
306
304
307
301
NEURAL NETWORK
302
308
310
305
≥Bth?
∫ dt

FIG. 3

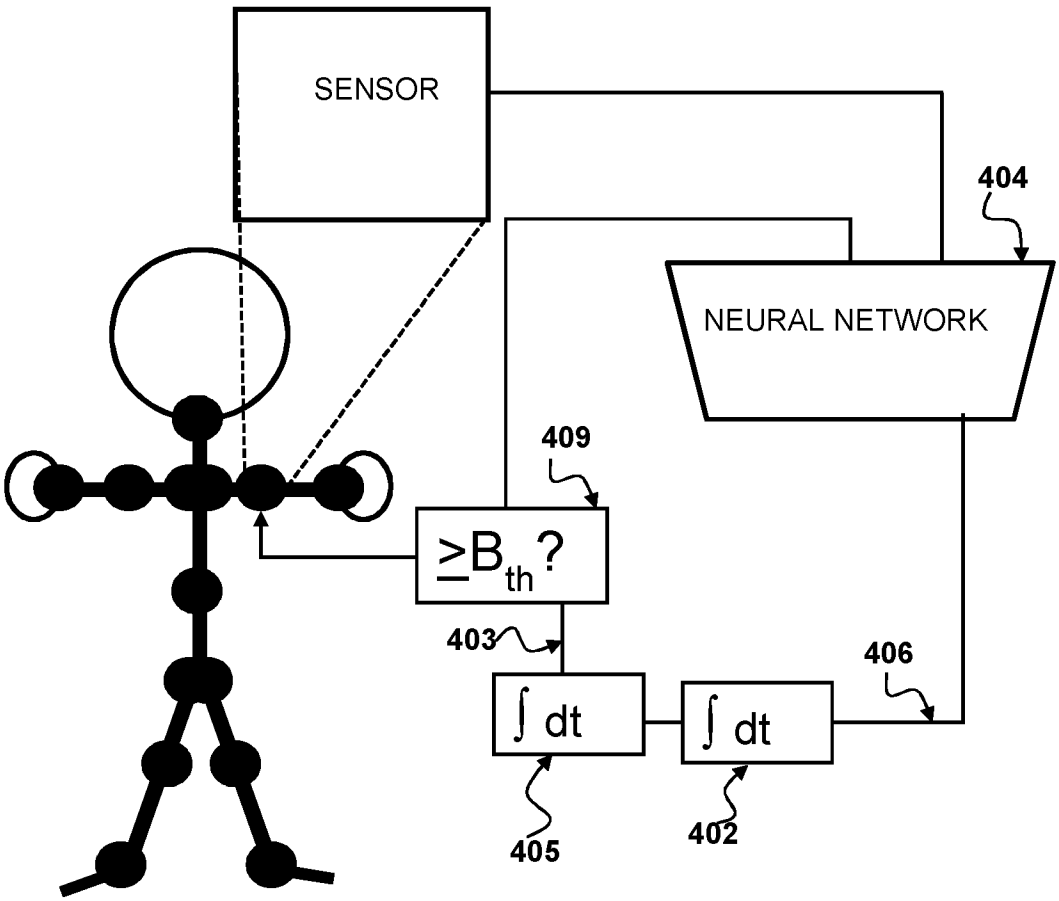


FIG. 4

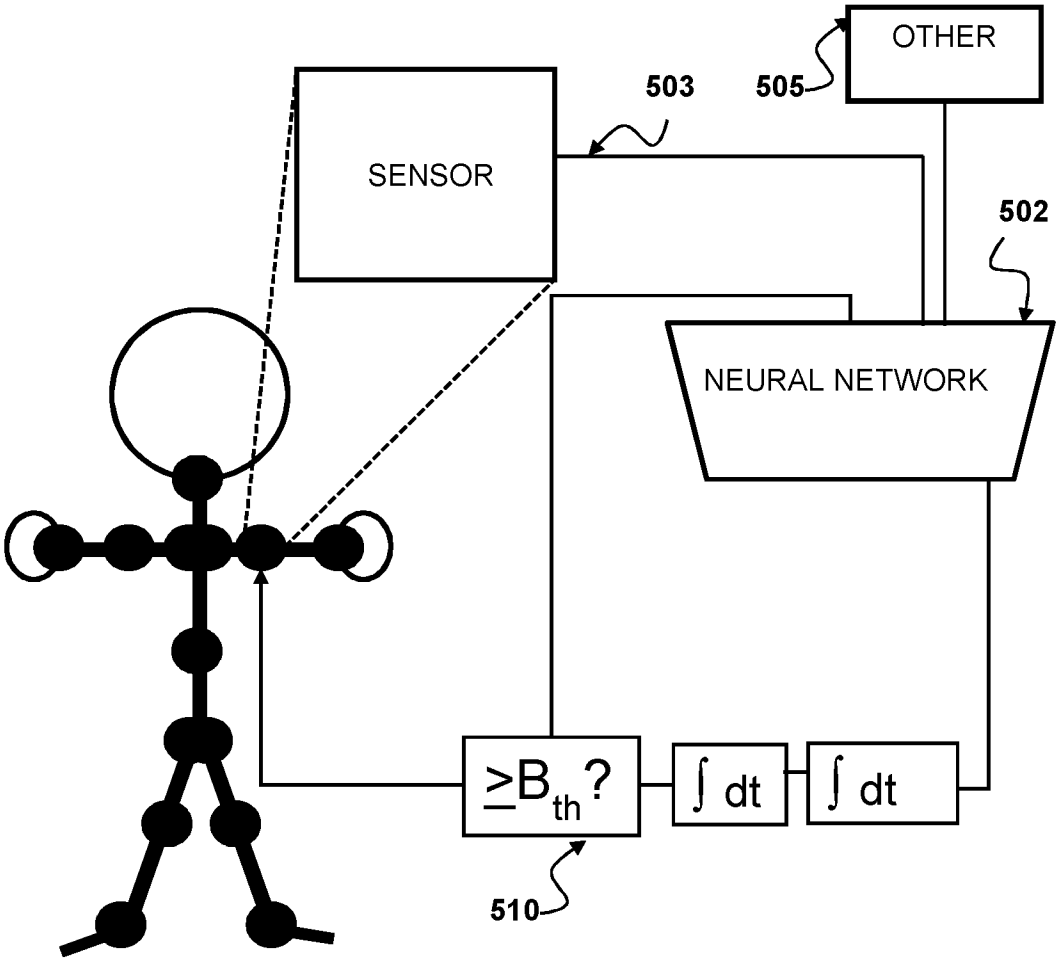


FIG. 5

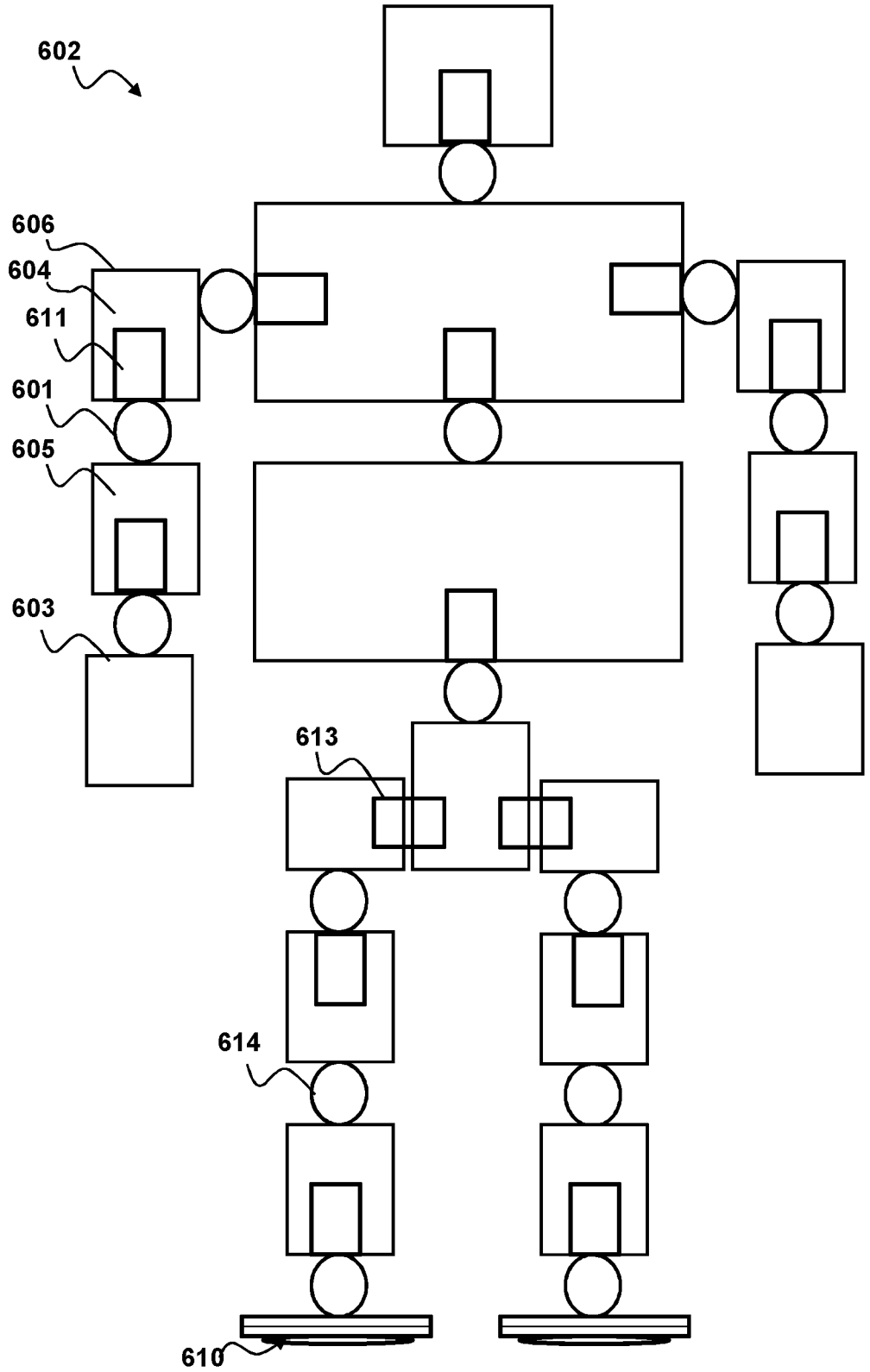


FIG. 6

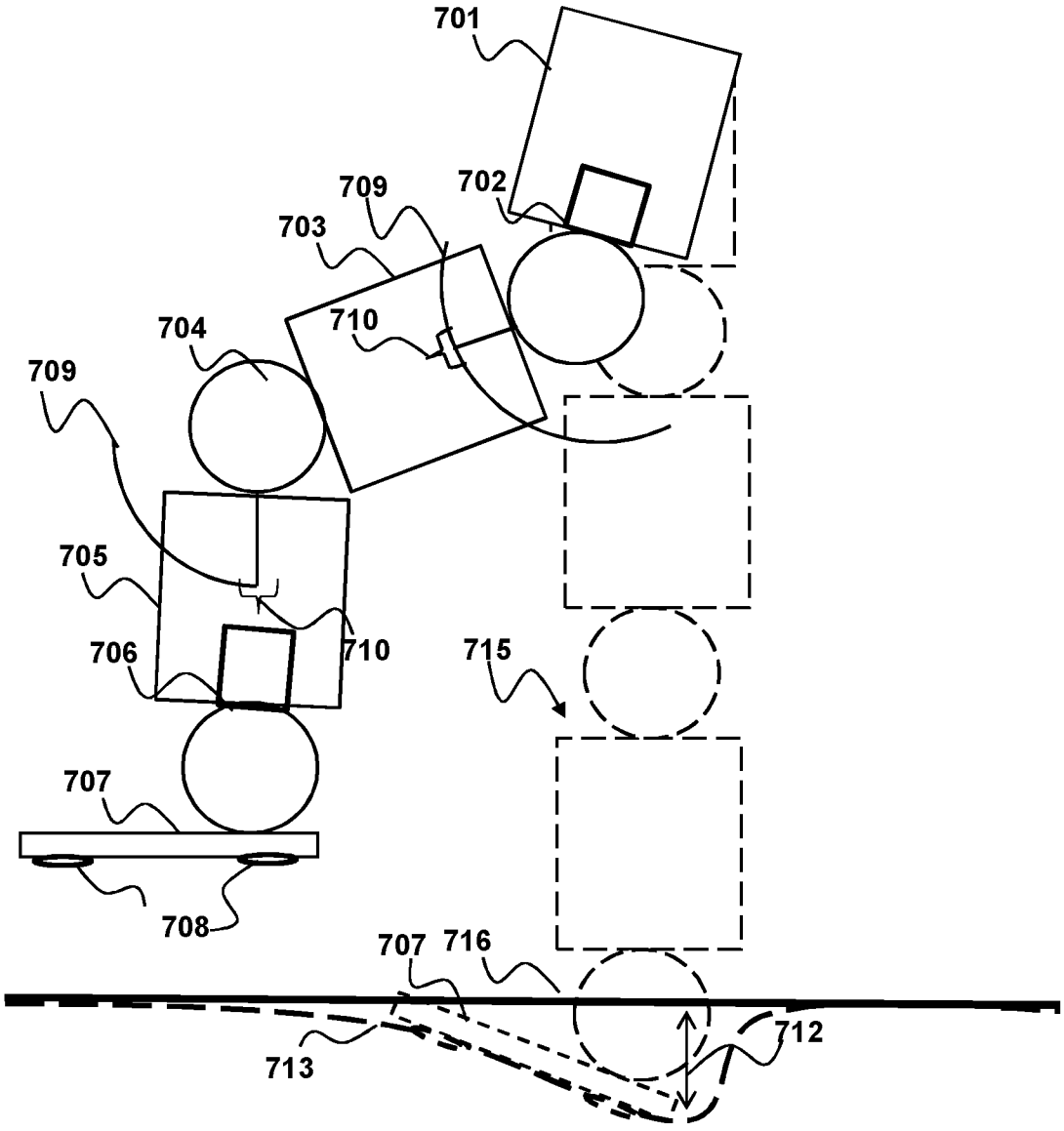

701
709
702
703
704
710
709
705
706
710
715
707
708
707
716
712
713

FIG. 7

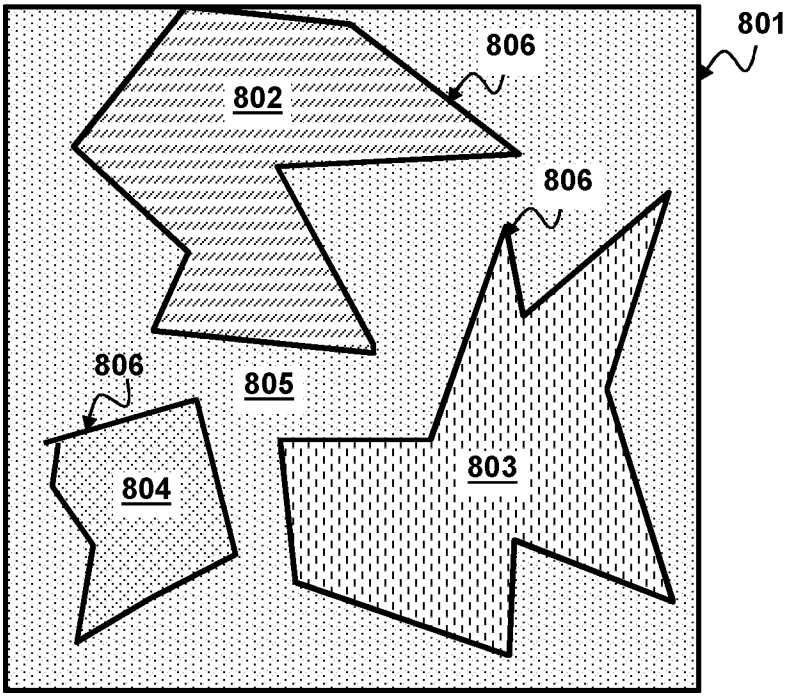


FIG. 8

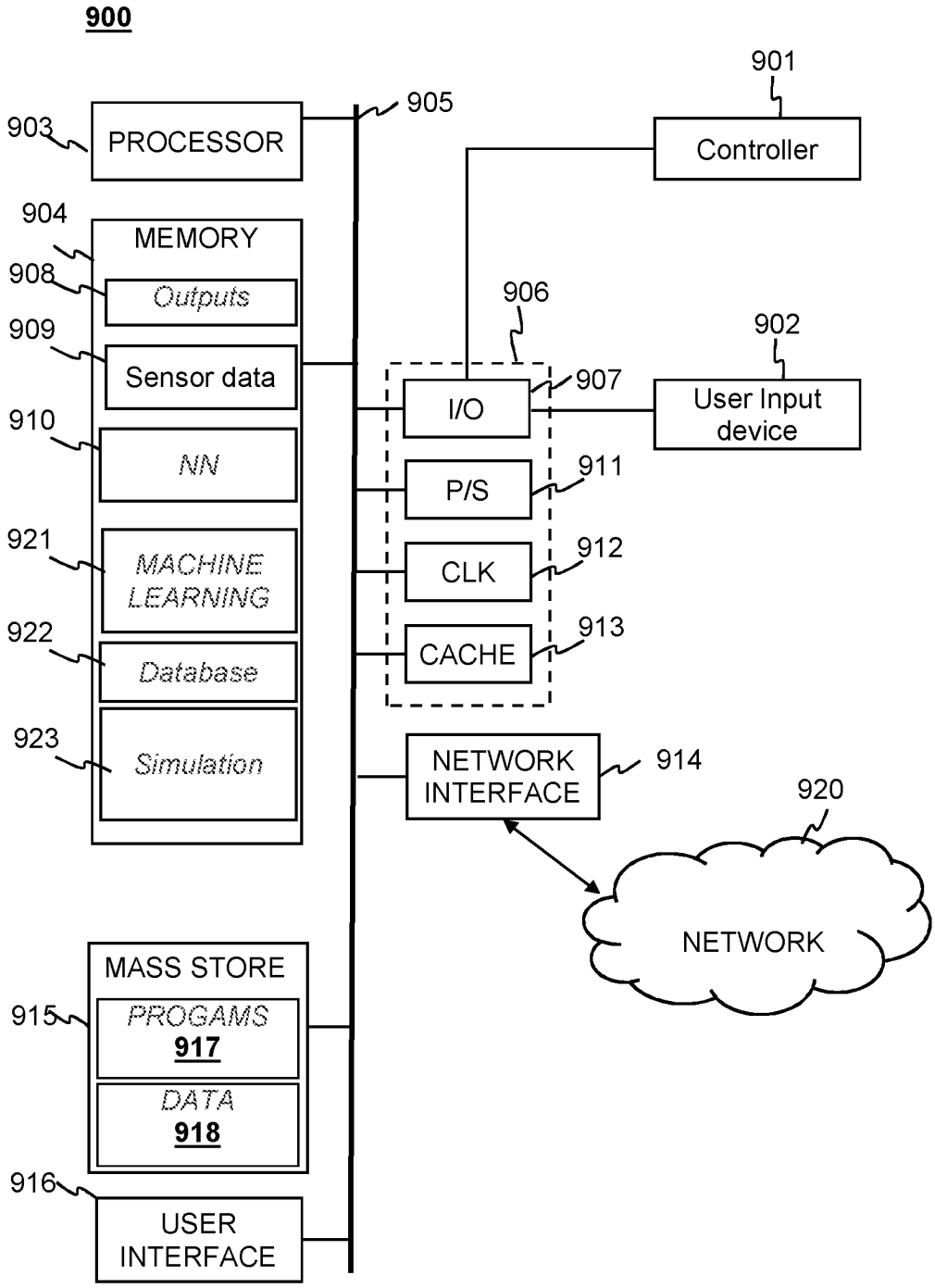

900
901
Controller
905
903
PROCESSOR
904
MEMORY
908
Outputs
906
902
909
Sensor data
907
I/O
User Input device
910
NN
911
P/S
921
MACHINE LEARNING
912
CLK
922
Database
913
CACHE
923
Simulation
NETWORK INTERFACE
914
920
NETWORK
MASS STORE
915
PROGAMS
917
DATA
918
916
USER INTERFACE

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 21/58390

A. CLASSIFICATION OF SUBJECT MATTER

IPC - A61N 1/18 (2021.01)

CPC - G06F 3/015, G09B 19/003, A61F 2/76, A61F 2/72, G16H 50/50, A61N 1/36003, G06F 3/011

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2020/0290203 A1 (SONY INTERACTIVE ENTERTAINMENT INC), 17 September 2020 (17.09.2020), entire document	1-22
A	Erlhagen et al. "The dynamic neural field approach to cognitive robotics." In: J. Neural Eng. 3 (2006) R36-R54, [online] [retrieved on 05 January 2022 (05.01.2022)] Retrieved from the Internet < URL: https://pubmed.ncbi.nlm.nih.gov/16921201/ >, entire document	1-22
A	Hu et al. "Impedance with Finite-Time Control Scheme for Robot-Environment Interaction." In: Mathematical Problems in Engineering, vol. 2020, Article ID 2796590, 18 pages, 25 May 2020, [online] [retrieved on 05 January 2022 (05.01.2022)] Retrieved from the Internet < URL: https://doi.org/10.1155/2020/2796590>, entire document	1-22
A	US 2020/0293881 A1 (Sony Interactive Entertainment Inc.), 17 September 2020 (17.09.2020), entire document	1-22
A	WO 2017/129200 A1 (MAX-PLANCK-GESELLSCHAFT ZUR FURDERUNG DER WISSENSCHAFTEN E.V.), 03 August 2017 (03.08.2017), entire document	1-22
A	US 2014/0107841 A1 (Board of Regents of the Nevada System of Higher Education, on behalf of the University of Nevada), 17 April 2014 (17.04.2014), entire document	1-22

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
06 January 2022	FEB 07 2022
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Kari Rodriquez Telephone No. PCT Helpdesk: 571-272-4300

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 21/58390

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization." In: Cornell University Library/ Computer Science/Robotics, 3 Mar 2018, [online] [retrieved on 05 January 2022 (05.01.2022)] Retrieved from the Internet < URL: arXiv:1710.06537v3 >, entire document	1-22
P,X	US 2021/0158141 A1 (Sony Interactive Entertainment Inc.) 27 May 2021 (27.05.2017), entire document	1-22