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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 October 2019 (31.10.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/209681 A1

(51) International Patent Classification:
B25J 9/16 (2006.01) *B62D 57/02* (2006.01)

(21) International Application Number:
PCT/US2019/028454

(22) International Filing Date:
22 April 2019 (22.04.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/661,055 22 April 2018 (22.04.2018) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(72) Inventors: **TAN, Jie**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **ZHANG, Tingnan**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **ISCEN, Atil**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **COUMANS, Erwin**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **BAI, Yunfei**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(74) Agent: **HIGDON, Scott** et al.; 401 S. Fourth Street, Suite 2600, Louisville, KY 40202 (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, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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).

(54) Title: SYSTEMS AND METHODS FOR LEARNING AGILE LOCOMOTION FOR MULTIPED ROBOTS

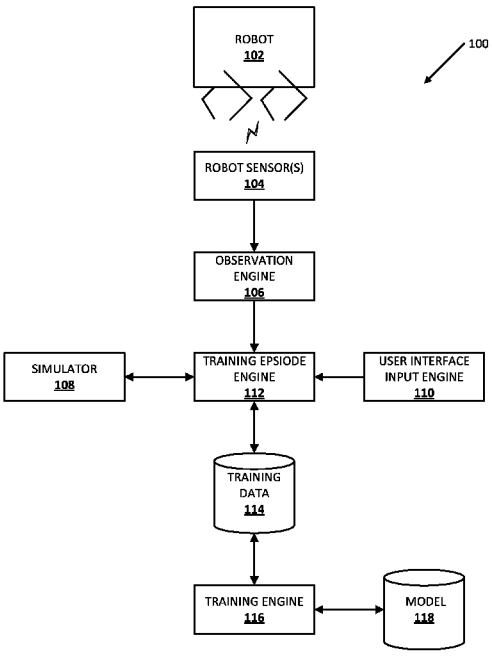

FIG. 1

(57) Abstract: Training and/or using a machine learning model for locomotion control of a robot, where the model is decoupled. In many implementations, the model is decoupled into an open loop component and a feedback component, where a user can provide a desired reference trajectory (*e.g.*, a symmetric sine curve) as input for the open loop component. In additional and/or alternative implementations, the model is decoupled into a pattern generator component and a feedback component, where a user can provide controlled parameter(s) as input for the pattern generator component to generate pattern generator phase data (*e.g.*, an asymmetric sine curve). The neural network model can be used to generate robot control parameters.

WO 2019/209681 A1

[Continued on next page]

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

SYSTEMS AND METHODS FOR LEARNING AGILE LOCOMOTION FOR MULTIPED ROBOTS

Background

[0001] Many robots are programmed to perform certain tasks. For example, a robot on an assembly line can be programmed to recognize certain objects, and perform particular manipulations to those certain objects.

[0002] Further, legged robots can be programmed to navigate complex terrains. Legged robots (*e.g.*, multiped robots with two or more legs) can adjust their gait, locomotion speed, foot placement, and/or ground clearance based on different walking surfaces. For example, a bipedal robot (*i.e.* a robot with two legs) can walk upright like a human to navigate a variety of terrains. Additionally or alternatively, a quadruped robot (*i.e.* a robot with four legs) can navigate surfaces on four limbs and in some cases can mimic the motions of a variety of animals such as horses, dogs, and/or primates. However, the task of training a legged robot to walk on different surfaces can be a very complex task. Physical robots themselves can be trained to walk, but this can be particularly time consuming and in some cases ineffective. As an alternative, simulations of a physical legged robot can be trained to navigate terrains. However, translating trained simulations of a legged robot to movement in the physical legged robots can presents its own set of challenges.

Summary

[0003] Implementations disclosed herein utilize deep reinforcement learning to train a model (*e.g.* a deep neural network model) that can be utilized to determine locomotion for a legged robot such as a quadruped robot, or other multiped robot. Implementations additionally or alternatively can relate to the utilization of such a model in controlling a multiped robot. The robotic locomotion in some of those implementations can be determined in part by a locomotion controller which can be decoupled into an open loop component and a feedback component. The open loop component can receive information that is based on input from a user and that is used in training robotic locomotion. The received information can include, for example, information that directly or indirectly defines robot gait, robot height, and/or other control parameter(s). In several implementations, the information that is provided to the open loop component can be in the form of a reference trajectory (*e.g.*, sine wave) that is generated based on user input. The feedback loop component can fill in missing portions of information, for

example the control of joint angles, base orientation, angular velocities, and/or other missing portions of information, such that the legged robot can still walk with respect to the user provided input to the open loop component. For example, the feedback loop component can be trained using reinforcement learning to determine the balance control (which can be tedious to design manually) for a user provided reference trajectory. Additionally or alternatively, the observation space can be reduced, which can make training simulations easier to translate to the real world. In some implementations, the observation space can be reduced by reducing the amount of sensor data used in training a legged robot for locomotion tasks.

[0004] In some additional or alternative implementations, robotic locomotion can be generated by a control policy controller (such as a locomotion policy controller), which can be decoupled into an open loop pattern generator and a neural network feedback component. A user can control the open loop pattern generator in training robotic locomotion by providing the open loop pattern generator control parameter(s) such as leg gait, leg height, and/or other control parameter(s). The neural network feedback component can fill in missing portions of information (*e.g.*, the control of joint angles, base orientation, angular velocities), such that the legged robot can still walk with respect to phase information provided by the open loop pattern generator and the user provided control parameters. In some implementations, the open loop pattern generator can create an asymmetric sine curve to provide to the neural network feedback component. The asymmetric sine curve can include a swing phase and a stance phase. The swing phase generally indicates one or more robotic legs are moving off the ground, and the stance phase generally indicates one or more robotic legs are positioned on the ground. In a variety of implementations, a user can provide robotic parameters to dynamically change robotic locomotion behavior even after the neural network feedback controller has been trained. For example, a user can dynamically change the speed, or gait of a robot after training by changing the user provided controlled parameters.

[0005] In some implementations, a method performed by one or more processors is provided that includes: receiving an instance of sensor data, the instance of sensor data generated based on output from one or more sensors of a robot, wherein the instance of sensor data is based on a state of the robot following control of the robot using a prior instance of robot control parameters generated using a neural network model, the neural network model representing a learned policy for a robotic locomotion task and being previously trained through reinforcement

learning. The method further includes receiving a reference trajectory for the robotic locomotion task, wherein the reference trajectory is decoupled from the sensor data and is influenced by user interaction via one or more user interface input devices. The method further includes generating an observation based on the instance of sensor data; and applying the observation and the reference trajectory to the neural network model to generate a current instance of robot control parameters. The method further includes controlling one or more actuators of a robot based on the current instance of robot control parameters.

[0006] These and other implementations can include one or more of the following features.

[0007] In some implementations, the robot is a legged robot includes a plurality of legs, and wherein the current instance of robot control parameters define, for each of the plurality of legs, a desired pose of the leg.

[0008] In some implementations, the generated observation indicates motor angles, roll of a base of the robot, pitch of a base of the robot, and angular velocity of the robot.

[0009] In some implementations, the generated observation indicates only the motor angles, the roll, the pitch, and the angular velocity of the robot.

[0010] In some implementations, the generated observations exclude one or more available observations that are indicated by the instance of sensor data.

[0011] In some implementations, the excluded one or more available observations include a yaw of the base of the robot.

[0012] In some implementations, the method further includes training the neural network model in a simulator using a simulated robot and using reinforcement learning.

[0013] In some implementations, training the neural network model in simulation includes modeling, for the simulated robot, a simulated latency between: the time when the output from one or more sensors of the robot is captured, and the time when one or more actuators of the robot are controlled based on the current instance of robot control parameters.

[0014] In some implementations, training the neural network model in simulation using reinforcement learning includes utilizing a reward function, during the reinforcement learning, wherein the utilized reward function penalizes a high robot energy consumption.

[0015] In some implementations, wherein the utilized reward function further encourages a faster forward robot speed.

[0016] In some implementations, the robot is a quadruped robot.

[0017] In some implementations, the reference trajectory indicates a robot gait and a robot height.

[0018] In some implementations, the reference trajectory include a symmetric sine function.

[0019] In some implementations, the locomotion task is trotting.

[0020] In some implementations, the locomotion task is galloping.

[0021] In some implementations, the one or more sensors are one or more motor encoders and one or more inertial measurement units.

[0022] In some implementations, a method implemented by one or more processors is provided and includes receiving an instance of sensor data, the instance of sensor data generated based on output from one or more sensors of a sensor component of a robot. The method further includes generating an observation based on the instance of sensor data for performing a robot action based on a neural network model representing a learned policy for a locomotion task for the robot, wherein the neural network model is decoupled into a pattern generator component and a neural network feedback component. The method further includes receiving controlled parameters based on user interaction with a user interface input device, wherein a user can change the controlled parameters at the user interface input device after the neural network model representing the reinforcement learning policy is trained. The method further includes applying the controlled parameters to the pattern generator component to generate pattern generator phase data. The method further includes applying the observation, the control parameters, and the pattern generator phase data to the neural network feedback component to generate robot control parameters. The method further includes controlling one or more actuators of a robot based on the robot control parameters.

[0023] In some implementations, the robot is a quadruped robot.

[0024] In some implementations, the controlled parameters are selected from the group consisting of gait, speed of locomotion, and height of locomotion.

[0025] In some implementations, generating pattern generator phase comprises generating parameterizing an asymmetric sine curve.

[0026] In some implementations, the asymmetric sine curve comprises a swing phase and a stance phase, wherein the swing phase indicates one or more legs of the quadruped robot are off the ground and the stance phase indicates one or more legs of the quadruped robot are on the ground.

[0027] In some implementations, the controlled parameters change the asymmetric sine curve.

[0028] In some implementations, the locomotion task is trotting.

[0029] In some implementations, the locomotion task is galloping.

[0030] In some implementations, the one or more sensors are one or more motor encoders and one or more inertial measurement units.

[0031] Other implementations may include a non-transitory computer readable storage medium storing instructions executable by one or more processors (*e.g.*, one or more central processing units to perform a method such as one or more of the methods described above and/or elsewhere herein. Yet another implementation may include a system of one or more computers and/or one or more robots that include one or more processors operable to execute stored instructions to perform a method such as one or more of the methods described above and/or elsewhere herein.

[0032] It should be appreciated that all combinations of the foregoing concepts and additional concepts described in greater detail herein are contemplated as being part of the subject matter disclosed herein. For example, all combinations of claimed subject matter appearing at the end of this disclosure are contemplated as being part of the subject matter disclosed herein.

Brief Description of the Drawings

[0033] FIG. 1 illustrates an example environment in which various implementations can be implemented.

[0034] FIG. 2 illustrates an example neural network model according to implementations disclosed herein.

[0035] FIG. 3 is a flowchart illustrating an example process of controlling actuator(s) of a robot based on robot control parameters generated using a neural network model according to implementations disclosed herein.

[0036] FIG. 4 is another flowchart illustrating an example process of training a neural network model according to implementations disclosed herein.

[0037] FIG. 5 illustrates another example of a neural network model according to implementations disclosed herein.

[0038] FIG. 6 is another flowchart illustrating another example process of controlling

actuator(s) of a robot based on robot control parameters generated using a neural network model according to implementations disclosed herein.

[0039] FIG. 7 is another flowchart illustrating another example process of training a neural network model according to implementations disclosed herein.

[0040] FIG. 8 schematically depicts an example architecture of a robot.

[0041] FIG. 9 schematically depicts an example architecture of a computer system.

Detailed Description

[0042] Various implementations are disclosed below that are related to training and/or utilizing a machine learning model (*e.g.*, a neural network model) in locomotion for a legged robot, such as a quadruped robot. In some implementations disclosed herein, the machine learning model is trained utilizing reinforcement learning and, when trained, represents a policy for use in generating control parameters that can be utilized in driving actuators of a legged robot to control locomotion of the legged robot. In some versions of those implementations, the control parameters generated at a given time utilizing such a model can be based on user input to an open loop component of a locomotion controller. Additionally or alternatively, the control parameters can be based on user provided controlled parameters to a pattern generator component of a control policy controller.

[0043] Robotic locomotion tasks can be learned in many implementations through reinforcement learning. The goal in reinforcement learning is to control an agent attempting to maximize a reward function which, in the context of a robotic skill (also referred to herein as a task), denotes a user-provided definition of what the robot should try to accomplish. At state x_t in time t, the agent chooses and executes action u_t according to its policy $\pi(x_t)$, transitions to a new state x_{t+1} according to the dynamics of the robot $p(x_t, u_t, x_{t+1})$, and receives a reward $r(x_t, u_t)$. The goal of reinforcement learning is to find the optimal policy $\pi *$ which maximizes the expected sum of rewards from an initial state distribution. The reward is determined based on the reward function which, as mentioned above, is dependent on the robotic task to be accomplished. Accordingly, reinforcement learning in the robotics context seeks to learn an optimal policy for performance of a given robotic task (*e.g.*, robotic locomotion).

[0044] Turning to the figures, FIG. 1 illustrates an example environment 100 in which implementations described herein may be implemented. FIG. 1 includes an example robot 102, an observation engine 106, a robotic simulator 108, a user interface input engine 120, a training

episode engine 112, and a training engine 116. Also included are robot senor(s) 104, training data 114, and one or more machine learning models 118.

[0045] Robot 102 is a legged robot having multiple degrees of freedom to enable robotic locomotion by controlling actuator(s) of the legs of the robot 102. For example, robot 102 can be a quadruped robot (*i.e.*, four legged robot), where each leg is controlled by two actuators that allow the leg to move in the sagittal plane. For instance, a first actuator of a corresponding leg can be at an attachment point between the leg and a body of the robot 102, and a second actuator of the corresponding leg can be between the attachment point and a distal end of the corresponding leg (*e.g.*, at a "knee" of the leg). The motors can be actuated through position control, with a Power Width Modulation (PWM) signal. Other quantities of motors and/or other motor control methodologies (besides PWM) can be utilized in some implementations.

[0046] In a variety of implementations, robot 102 is equipped with a variety of robot sensors 104, such as motor encoders that measure the motor angles, an inertial measurement unit (IMU) that measures the orientation and angular velocity of the robot base, and/or additional sensors to measure the position of the robot. Although a particular robot 102 is illustrated in FIG. 1, additional and/or alternative robots may be utilized including robots having more legs (*e.g.*, a five legged robot, a six legged robot, an eight legged robot, and/or a robot with additional legs), robots having fewer legs (*e.g.*, a three legged robot, a two legged robot), robots having robot arms, robots having a humanoid form, robots having an animal form, robots that include one or more wheels in addition to robot legs, and so forth.

[0047] Training machine learning models based on data from real-world physical robots can be time consuming (*e.g.*, actually navigating a large quantity of paths requires a large quantity of time), can consume a large amount of resources (*e.g.*, power required to operate the robots), and/or can cause wear and tear on the robots being utilized. In view of these and/or other considerations, robotic simulators (such as robotic simulator 108) can be utilized in generating simulated training data that can be utilized in training of machine learning models (such as model 118). However, there is often a meaningful "reality gap" that exists between real robots and real environments and the simulated robots and/or simulated environments simulated by a robot simulator.

[0048] In many implementations, the reality gap can be decreased by adapting the simulator (*e.g.*, simulator 108) and/or data generated using the simulator (*e.g.*, the simulated robot, the

simulated environment, and/or additional data generated using the simulator). For example, the actuator model that is utilized in simulator 108 can be designed to more accurately simulate a robotic actuator. In a variety of implementations, this increased accuracy of simulated actuator(s) can decrease the reality gap.

[0049] For example, a large reality gap can be found when simulating actuator(s) using traditional approaches. For example, one constraint $e_{n+1} = 0$ is formulated for each motor where e_{n+1} is an error at the end of the current time step. The error can be defined as

[0050] $$e_{n+1} = k_p\left(\underline{q} - q_{n+1}\right) + k_d(\underline{\dot{q}} - \dot{q}_{n+1}) \quad (1)$$

[0051] where $\underline{q}$ and $\underline{\dot{q}}$ are desired motor angle and velocity, q_{n+1} and $\dot{q}_{n+1}$ are the motor angle and velocity at the end of the current time step, k_p is the proportional gain and k_d is the derivative gain. Equation (1) ensures the motor angle and velocity in the future (*i.e.*, at the end of the time step) satisfy the error constraint e_{n+1}. This increases motor stability in simulation if large gains are used, but the motors could oscillate in reality.

[0052] To eliminate this discrepancy for actuators, many implementations utilize an actuator model according to the dynamics of an ideal DC motor. Given a PWM signal, the torque of the motor can be represented as

[0053] $$\tau = K_t I \quad (2)$$

[0054] $$I = \frac{V*PWM - V_{emf}}{R} \quad (3)$$

[0055] $$V_{emf} = K_t \dot{q} \quad (4)$$

[0056] where I is the armature current, K_t is the torque constant or back electromotive force (EMF) constant, V is the supplied voltage, V_{emf} is the back EMF voltage, and R is the armature resistance. The parameters K_t and R can be determined by the specific actuators. Utilizing the motor model represented in equations (2) – (4) in training a controller, a robot often sinks to its feet and cannot lift while the same controller works fine in simulation because the linear torque-current relation only holds for ideal motors. In reality, the torque saturates as the current increases. A piecewise linear function can be utilized to characterize this nonlinear torque-current relation. In simulation, once the current is computed from PWM (equations (3) and (4)), the piece-wise function can be utilized to look up the corresponding torque.

[0057] The PWM is controlled through a classic PD servo in the positional control mode.

[0058] $$PWM = k_p\left(\underline{q} - q_n\right) + k_d(\underline{\dot{q}} - \dot{q}_n) \quad (5)$$

[0059] Additionally or alternatively, the target velocity can be set to zero (*i.e.*, $\underline{\dot{q}} = 0$). Actuating a motor with a desired trajectory of sine curve using this actuator model agrees with the ground truth.

[0060] In many implementations, latency simulated using simulator 108 can provide for an additional and/or alternative reduction in the reality gap. Latency is a cause of instability for feedback control and can include: the time delay between when a motor command is sent that causes the state of the robot to change and the robot receives the motor command, the time delay between when the robot receives the motor command and the state of the robot changes, the time delay between when the sensor measurement of the change in state is captured at the robot and reported back to the controller, and/or additional delay(s). Robotic simulators where motor commands take effect immediately and the sensors report back the state instantaneously make the stability region of a feedback controller in simulation much larger than its implementation on hardware. This can cause a feedback policy learned in simulation start to oscillate, diverge, and ultimately fail in the real world. Accordingly, latency simulation techniques disclosed herein are utilized to mitigate these and/or other drawbacks, leading to mitigation of the reality gap and improved performance of a model, trained at least in part on simulated data, when utilized to control a real robot.

[0061] To model latency in accordance with simulator 108, a history can be kept of observations and their measurement time $\{(t_i, O_i)_{i=0,1,\ldots,n-1}\}$, where $t_i = i\Delta t$ and Δt is the time step. At the current step n, when the controller needs an observation, simulator 108 can search through the history to find two adjacent observations O_i and O_{i+1} where $t_i \leq n\Delta t - t_{latency} \leq t_{i+1}$ and linearly interpolate them. To measure latency on the physical system, a spike of PWM signal (*e.g.*, PWM = 1 for only one time step) can be sent that causes a sudden change of the motor angle. The time delay between when the spike is sent and when the resultant motor movement is reported can be measured.

[0062] Observation engine 106 can utilize data measured by robot sensor(s) 104 to determine a variety of observations. The observations can include one or multiple of the roll of the robot base, the pitch of the robot base, angular velocities of the robot base along one or more axes (such as an angular velocity of the robot base along the axis corresponding with roll and/or an

angular velocity of the robot base along the axis corresponding with pitch), motor angles corresponding to motor(s) of the robot legs, and/or other observations. In many implementations, the observation space can be limited to exclude unreliable measurements including measurements with a high level of noise such as motor velocities, measurements that can drift quickly such as the yaw of the robot base, and/or other unreliable measurements. Keeping the observation space compact helps transfer a policy trained in simulation to the real robot.

[0063] User interface input engine 110 can capture a variety of user inputs for use in training the machine learning model 118 as well as controlling the locomotion of a real robot. For example, a user can provide a reference trajectory (as illustrated in FIG. 2), controlled parameters (as illustrated in FIG. 5) and/or additional user interface input for use in training machine learning model 118.

[0064] Training episode engine 112, in accordance with a variety of implementations, can be utilized in generating reinforcement learning training episodes, such as training data 114. For example, training episode engine 112 can create a training episode using data generated by simulator 108 and user interface input engine 110. Additionally or alternatively, observations generated using observation engine 106 can be utilized by training episode engine 112 to generate training episodes. Training engine 116 can utilize training data 114 generated by training episode engine 112 to train machine learning model 118. In a variety of implementations, machine learning model 118 is a neural network model, which is a decoupled network and can include a convolutional neural network model, a recurrent network model, and/or an additional type of neural network model. Machine learning model 118 in a variety of implementations is trained by training engine 116 using reinforcement learning. An example machine learning model 118 is illustrated in FIG. 2. An additional or alternative machine learning model 118 is illustrated in FIG. 5.

[0065] Turning to FIG. 2, block diagram 200 illustrates a decoupled machine learning model (such as machine learning model 118 of FIG. 1) in accordance with implementations described herein. In many implementations, the machine learning model is decoupled into an open loop component 206 and a feedback component 208. The decoupled machine learning model allows a user more control over the training of the locomotion policy. Open loop component 206 allows a user to provide a reference trajectory 202 (*e.g.*, a user supplied symmetric sine curve) to

express, for example, the desired gait of the robot. The feedback component 208 of the machine learning model adjusts the leg poses on top of the reference trajectory 202 based on the observation 204 indicating the current state of the robot (such as simulated observation determined using simulator 108 and/or observations from the real robot determined using observation engine 106).

[0066] The policy of the network may be represented as:

[0067] $$a(t, o) = \underline{a}(t) + \pi(o) \quad (6)$$

[0068] where $a(t, o)$ is the machine learning model with respect to the reference trajectory 202 and observation 204, $\underline{a}(t)$ is the open loop component 206, $\pi(o)$ is the feedback component 208, t is the time and o is observation 204. This represents a hybrid policy that provides a full spectrum of robot controllability. It can be varied to determine robot locomotion from fully user-specified to entirely learned from scratch. For example, a user-specified policy can be used by setting both the lower and upper bounds of $\pi(o)$ (*i.e.*, the feedback component) to zero. Additionally or alternatively, a policy can be learned from scratch by setting $\underline{a}(t) = 0$ (*i.e.*, setting the open loop component to equal zero) and giving the feedback component $\pi(o)$ a wide output range. In many implementations, the amount of user control applied to the system can be determined by varying the open loop signal and/or the output bound of the feedback component.

[0069] FIG. 3 is a flowchart illustrating an example process 300 of generating robot control parameters to control robot locomotion using a decoupled neural machine learning model. For convenience, the operations of process 300 are described with reference to a system that performs the operations. This system may include various components of various computer systems, such as one or more components depicted in FIG. 8 and/or FIG. 9. Moreover, while operations of process 300 are shown in a particular order, this is not meant to be limiting. One or more operations by be reordered, omitted, and/or added.

[0070] At block 302, the system receives an instance of sensor data. Sensor data can be captured by a variety of sensors (*e.g.*, motor encoder(s), IMU, and/or additional sensor(s)) and is based on the state of the robot (*e.g.*, motor angle(s), orientation of the robot, velocity of the robot, etc.). In many implementations, the sensor data is based on the state of the robot following control of the robot using a prior instance of robot control parameters.

[0071] At block 304, the system generates an observation using the sensor data. For example, the system can limit the observation space to exclude measurements that drift quickly and/or typically contain large amounts of noise. For example, the observation space can be limited to the roll of the base, the pitch of the base, the angular velocities of the base along the roll and pitch axes, and the motor angles of the robot leg motors (*e.g.*, 8 motor angles where each leg of a quadruped robot includes two motors).

[0072] At block 306, the system receives a reference trajectory decoupled from the sensor data via user interface input device(s). The reference trajectory can define a user specified desired gait. In many implementations, the reference trajectory is a symmetric sine curve.

[0073] At block 308, the system generates robot control parameters by applying the observation and the reference trajectory to a trained machine learning model. Robot control parameters can indicate the desired pose of the robot at the next state. In a variety of implementations, robot control parameters can indicate the desired change from the current state to the next desired state.

[0074] At block 310, the system controls actuator(s) of a robot based on the robot control parameters.

[0075] FIG. 4 is a flowchart illustrating an example process 400 of training a decoupled machine learning model for robotic locomotion. For convenience, the operations of process 400 are described with reference to a system that performs the operations. This system may include various components of various computer systems, such as one or more components depicted in FIG. 8 and/or FIG. 9. Moreover, while operations of process 400 are shown in a particular order, this is not meant to be limiting. One or more operations by be reordered, omitted, and/or added.

[0076] At block 402, the system generates an instance of robot control parameters by applying an observation and a reference trajectory as input to a machine learning model. In some implementations, the instance of robot control parameters is generated using a robotic simulator such as simulator 108 of FIG. 1.

[0077] At block 404, the system controls actuator(s) of a robot based on the instance of robot control parameters. For example, the system can control the motor angle of one or more motors of the robot legs.

[0078] At block 406, the system determines an updated observation. In a variety of implementations, the updated observation is based on: the position of the robot, IMU readings,

motor angle(s), and/or additional sensor measurements of the robot after the system controls actuator(s) of the robot at block 404.

[0079] At block 408, the system determines a reward signal based on the observation, the updated observation, and the reference trajectory. In a variety of implementations, the reward signal can be determined using a reward function which encourages faster forward running speed and/or penalizes high energy consumption. For example, a reward function can include:

[0080] $$r = (p_n - p_{n-1}) \cdot d - \omega \Delta t |\tau_n \cdot \dot{q}_n| \quad (7)$$

[0081] where p_n is the position of the robot base at the current time step, p_{n-1} is the position of the robot base at the previous time step, d is the desired running direction, Δt is the time step, τ are the motor torques, and $\dot{q}$ are the motor velocities. The first term measures the running distance towards the desired direction and the second term measures the energy expenditure. ω is a weight that balances these two terms.

[0082] At block 410, the system updates one or more parameters of the machine learning model using the reward signal. In many implementations, the rewards are accumulated at each episode. In some implementations, a training episode terminates after a specific robot condition is met such as: the robot has taken a desired number of steps (*e.g.*, the training episode terminates after the robot has taken 1000 steps) and/or the robot loses balance (*e.g.*, the robot base tilts more than 0.5 radians with respect to the ground plane).

[0083] Turning to FIG. 5, block diagram 500 illustrates an additional and/or alternative machine learning model (such as machine learning model 118 of FIG. 1). In many implementations, the machine learning model is decoupled into a pattern generator component 504 and a feedback component 510. In many implementations, a user can supply controlled parameters 502 such as desired locomotion speed, walking height, and/or additional user supplied parameters to generate pattern generator phase data 506. In other words, changing one or more controlled parameters 502 will change the pattern generator phase data 506 generated using pattern generator component 504. In many implementations, pattern generator phase data 506 provides a reference of the overall behavior of the robot locomotion (such as the trajectory of the legs), and can be represented by asymmetric sine curves.

[0084] One or more observations 508 indicating the current state of the robot (such as simulated observations determined using simulator 108 and/or observations captured from a real robot using observation engine 106 of FIG. 1) can be provided as input to feedback

component 510 of the decoupled neural network. In many implementations, controlled parameters 502 and/or pattern generator phase data 506 may additionally or alternatively be processed as input using feedback component 510. Output generated by feedback component 510 may be combined with pattern generator phase data 506 to determine one or more robot control parameters 512.

[0085] FIG. 6 is a flowchart illustrating an example process 600 of generating robot control parameters using a decoupled machine learning model in accordance with a variety of implementations. For convenience, the operations of process 600 are described with reference to a system that performs the operations. This system may include various components of various computer systems, such as one or more components depicted in FIG. 8 and/or FIG. 9. Moreover, while operations of process 600 are shown in a particular order, this is not meant to be limiting. One or more operations by be reordered, omitted, and/or added.

[0086] At block 602, the system receives an instance of sensor data. As discussed above, sensor data can be captured by a variety of sensors (*e.g.*, motor encoder(s), IMU, and/or additional sensor(s)) and is based on the state of the robot. In many implementations, the sensor data is based on the state of the robot following control of the robot using a prior instance of robot control parameters.

[0087] At block 604, the system generates an observation based on the instance of sensor data. For example, the system can limit the observation space to exclude measurements that drift quickly and/or typically contain large amounts of noise. In many implementations, the observation space is limited to the roll of the base, the pitch of the base, the angular velocities along the roll and pitch axes, and the motor angles of robot leg motors.

[0088] At block 606, the system receives controlled parameters based on user interaction with a user interface input device. Controlled parameters can include one or more parameters defining a desired gait of the robot and can include locomotion speed, foot placement, ground placement, and/or additional parameter(s).

[0089] At block 608, the system generates pattern generator phase data by applying the controlled parameters as input to a pattern generator component of a trained machine learning model. In many implementations, the pattern generator phase data is an asymmetric sine curve representing the swing phase and the stance phase of a robotic leg.

[0090] At block 610, the system generates robot control parameters by applying: (1) the observation, (2) the controlled parameters, and (3) the pattern generator phase data as input to a feedback component of the machine learning model. In many implementations, the feedback component of the machine learning model is decoupled from the pattern generator component of the machine learning model.

[0091] At block 612, the system controls actuator(s) of a robot based on the robot control parameters.

[0092] FIG. 7 is a flowchart illustrating an example process 700 of training a machine learning model for robotic locomotion in accordance with a variety of implementations. For convenience, the operations of process 700 are described with reference to a system that performs the operations. This system may include various components of various computer systems, such as one or more components depicted in FIG. 8 and/or FIG. 9. Moreover, while operations of process 700 are shown in a particular order, this is not meant to be limiting. One or more operations by be reordered, omitted, and/or added.

[0093] At block 702, the system generates pattern generator phase data by applying controlled parameters as input to a pattern generator component of a machine learning model. In a variety of implementations, the controlled parameters are provided to the system by a user.

[0094] At block 704, the system generates an instance of robot control parameters by applying: (1) an observation, (2) the controlled parameters, and (3) the pattern generator phase data as input to a feedback component of the machine learning model. In many implementations, the feedback component of the machine learning model is decoupled from the pattern generator component.

[0095] At block 706, the system controls actuator(s) of a robot based on the instance of robot control parameters. For example, the system can move one or more legs of the robot by controlling actuator(s) of the robot.

[0096] At block 708, the system determines an updated observation. In many implementations, the updated observation is determined using feedback data captured by one or more sensors of the robot.

[0097] At block 710, the system determines a reward signal based on the observation, the updated observation, and the controlled parameters. In some implementations, the reward

signal optimizes energy efficient locomotion. In some implementations, the reward signal is similar to the reward signal determined at block 408 of FIG. 4.

[0098] At block 712, the system updates one or more parameters of the machine learning model using the reward signal. For example, one or more weights of the feedback component of the machine learning model can be updated. In many implementations, the rewards are accumulated at each episode. In some implementations, a training episode terminates after a specific robot condition is met such as: the robot has taken a desired number of steps (*e.g.*, the training episode terminates after the robot has taken 1000 steps) and/or the robot loses balance (*e.g.*, the robot base tilts more than 0.5 radians with respect to the ground plane).

[0099] FIG. 8 schematically depicts an example architecture of a robot 825. The robot 825 includes a robot control system 860, one or more operational components 825a-825n, and one or more sensors 842a-842m. The sensors 842a-842m may include, for example, vision sensors, light sensors, pressure sensors, pressure wave sensors (e.g., microphones), proximity sensors, accelerometers, gyroscopes, thermometers, barometers, and so forth. While sensors 842a-m are depicted as being integral with robot 825, this is not meant to be limiting. In some implementations, sensors 842a-m may be located external to robot 825, e.g., as standalone units.

[0100] Operational components 840a-840n may include, for example, one or more end effectors and/or one or more servo motors or other actuators to effectuate movement of one or more components of the robot. For example, the robot 825 may have multiple degrees of freedom and each of the actuators may control actuation of the robot 825 within one or more of the degrees of freedom responsive to the control commands. As used herein, the term actuator encompasses a mechanical or electrical device that creates motion (*e.g.*, a motor), in addition to any driver(s) that may be associated with the actuator and that translate received control commands into one or more signals for driving the actuator. Accordingly, providing a control command to an actuator may comprise providing the control command to a driver that translates the control command into appropriate signals for driving an electrical or mechanical device to create desired motion.

[0101] The robot control system 860 may be implemented in one or more processors, such as a CPU, GPU, and/or other controller(s) of the robot 825. In some implementations, the robot 825 may comprise a "brain box" that may include all or aspects of the control system 860. For

example, the brain box may provide real time bursts of data to the operational components 840a-n, with each of the real time bursts comprising a set of one or more control commands that dictate, *inter alia*, the parameters of motion (if any) for each of one or more of the operational components 840a-n. In some implementations, the robot control system 860 may perform one or more aspects of processes 300, 400, 500, and/or 700 described herein. As described herein, in some implementations all or aspects of the control commands generated by control system 860 can position limb(s) of robot 825 for robotic locomotion tasks. Although control system 860 is illustrated in FIG. 8 as an integral part of robot 825, in some implementations, all or aspects of the control system 860 may be implemented in a component that is separate from, but in communication with robot 825. For example, all or aspects of control system 860 may be implemented on one or more computing devices that are in wired and/or wireless communication with the robot 825, such as computing device 910.

[0102] FIG. 9 is a block diagram of an example computing device 910 that may optionally be utilized to perform one or more aspects of techniques described herein. For example, in some implementations computing device 910 may be utilized to provide desired locomotion by robot 825 and/or other robots. Computing device 910 typically includes at least one processor 914 which communicates with a number of peripheral devices via bus subsystem 912. These peripheral devices may include a storage subsystem 924, including, for example, a memory subsystem 925 and a file storage subsystem 926, user interface output devices 920, user interface input devices 922, and a network interface subsystem 916. The input and output devices allow user interaction with computing device 910. Network interface subsystem 916 provides an interface to outside networks and is coupled to corresponding interface devices in other computing devices.

[0103] User interface input devices 922 may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a touchscreen incorporated into the display, audio input devices such as voice recognition systems, microphones, and/or other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and ways to input information into computing device 910 or onto a communication network.

[0104] User interface output devices 920 may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem may include

a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), a projection device, or some other mechanism for creating a visible image. The display subsystem may also provide non-visual display such as via audio output devices. In general, use of the term "output device" is intended to include all possible types of devices and ways to output information from computing device 910 to the user or to another machine or computing device.

[0105] Storage subsystem 924 stores programming and data constructs that provide the functionality of some or all of the modules described herein. For example, the storage subsystem 924 may include the logic to perform selected aspects of the process of FIGS. 3, 4, 5, and/or 7.

[0106] These software modules are generally executed by processor 914 alone or in combination with other processors. Memory 925 used in the storage subsystem 924 can include a number of memories including a main random access memory (RAM) 930 for storage of instructions and data during program execution and a read only memory (ROM) 932 in which fixed instructions are stored. A file storage subsystem 926 can provide persistent storage for program and data files, and may include a hard disk drive, a floppy disk drive along with associated removable media, a CD-ROM drive, an optical drive, or removable media cartridges. The modules implementing the functionality of certain implementations may be stored by file storage subsystem 926 in the storage subsystem 924, or in other machines accessible by the processor(s) 914.

[0107] Bus subsystem 912 provides a mechanism for letting the various components and subsystems of computing device 910 communicate with each other as intended. Although bus subsystem 912 is shown schematically as a single bus, alternative implementations of the bus subsystem may use multiple busses.

[0108] Computing device 910 can be of varying types including a workstation, server, computing cluster, blade server, server farm, or any other data processing system or computing device. Due to the ever-changing nature of computers and networks, the description of computing device 910 depicted in Fig. 9 is intended only as a specific example for purposes of illustrating some implementations. Many other configurations of computing device 910 are possible having more or fewer components than the computing device depicted in Fig. 9.

[0109] While several implementations have been described and illustrated herein, a variety of other means and/or structures for performing the function and/or obtaining the results

and/or one or more of the advantages described herein may be utilized, and each of such variations and/or modifications is deemed to be within the scope of the implementations described herein. More generally, all parameters, dimensions, materials, and configurations described herein are meant to be exemplary and that the actual parameters, dimensions, materials, and/or configurations will depend upon the specific application or applications for which the teachings is/are used. Those skilled in the art will recognize, or be able to ascertain using no more than routine experimentation, many equivalents to the specific implementations described herein. It is, therefore, to be understood that the foregoing implementations are presented by way of example only and that, within the scope of the appended claims and equivalents thereto, implementations may be practiced otherwise than as specifically described and claimed. Implementations of the present disclosure are directed to each individual feature, system, article, material, kit, and/or method described herein. In addition, any combination of two or more such features, systems, articles, materials, kits, and/or methods, if such features, systems, articles, materials, kits, and/or methods are not mutually inconsistent, is included within the scope of the present disclosure.

CLAIMS

We claim:

1. A method implemented by one or more processors, comprising:

receiving an instance of sensor data, the instance of sensor data generated based on output from one or more sensors of a robot,

wherein the instance of sensor data is based on a state of the robot following control of the robot using a prior instance of robot control parameters generated using a neural network model, the neural network model representing a learned policy for a robotic locomotion task and being previously trained through reinforcement learning;

receiving a reference trajectory for the robotic locomotion task, wherein the reference trajectory is decoupled from the sensor data and is influenced by user interaction via one or more user interface input devices;

generating an observation based on the instance of sensor data;

applying the observation and the reference trajectory to the neural network model to generate a current instance of robot control parameters; and

controlling one or more actuators of a robot based on the current instance of robot control parameters.

2. The method of claim 1, wherein the robot is a legged robot comprising a plurality of legs, and wherein the current instance of robot control parameters define, for each of the plurality of legs, a desired pose of the leg.
3. The method of claim 1, wherein the generated observation indicates motor angles, roll of a base of the robot, pitch of a base of the robot, and angular velocity of the robot.
4. The method of claim 3, wherein the generated observation indicates only the motor angles, the roll, the pitch, and the angular velocity of the robot.
5. The method of claim 3, wherein the generated observations exclude one or more available observations that are indicated by the instance of sensor data.
6. The method of claim 5, wherein the excluded one or more available observations include a yaw of the base of the robot.
7. The method of claim 1, further comprising:

training the neural network model in a simulator using a simulated robot and using reinforcement learning.

8. The method of claim 7, wherein training the neural network model in simulation comprises modeling, for the simulated robot, a simulated latency between: the time when the output from one or more sensors of the robot is captured, and the time when one or more actuators of the robot are controlled based on the current instance of robot control parameters.
9. The method of claim 7, wherein training the neural network model in simulation using reinforcement learning comprises:

utilizing a reward function, during the reinforcement learning, wherein the utilized reward function penalizes a high robot energy consumption.

10. The method of claim 9, wherein the utilized reward function further encourages a faster forward robot speed.
11. The method of claim 1, wherein the reference trajectory indicates a robot gait and a robot height.
12. The method of claim 1, wherein the reference trajectory comprises a symmetric sine function.
13. The method of claim 1, wherein the one or more sensors are one or more motor encoders and one or more inertial measurement units.
14. A method implemented by one or more processors, comprising:

receiving an instance of sensor data, the instance of sensor data generated based on output from one or more sensors of a sensor component of a robot;

generating an observation based on the instance of sensor data for performing a robot action based on a neural network model representing a learned policy for a locomotion task for the robot, wherein the neural network model is decoupled into a pattern generator component and a neural network feedback component;

receiving controlled parameters based on user interaction with a user interface input device, wherein a user can change the controlled parameters at the user interface input device after the neural network model representing the reinforcement learning policy is trained;

applying the controlled parameters to the pattern generator component to generate pattern generator phase data;

applying the observation, the controlled parameters, and the pattern generator phase data to the neural network feedback component to generate robot control parameters; and

controlling one or more actuators of a robot based on the robot control parameters.

15. The method of claim 14, wherein the robot is a quadruped robot.
16. The method of claim 15, wherein the controlled parameters are selected from the group consisting of gait, speed of locomotion, and height of locomotion.
17. The method of claim 15, wherein generating pattern generator phase comprises generating an asymmetric sine curve.
18. The method of claim 15, wherein the asymmetric sine curve comprises a swing phase and a stance phase, wherein the swing phase indicates one or more legs of the quadruped robot are off the ground and the stance phase indicates one or more legs of the quadruped robot are on the ground.
19. The method of claim 18, wherein the controlled parameters change the asymmetric sine curve.
20. The method of claim 14, wherein the one or more sensors are one or more motor encoders and one or more inertial measurement units.

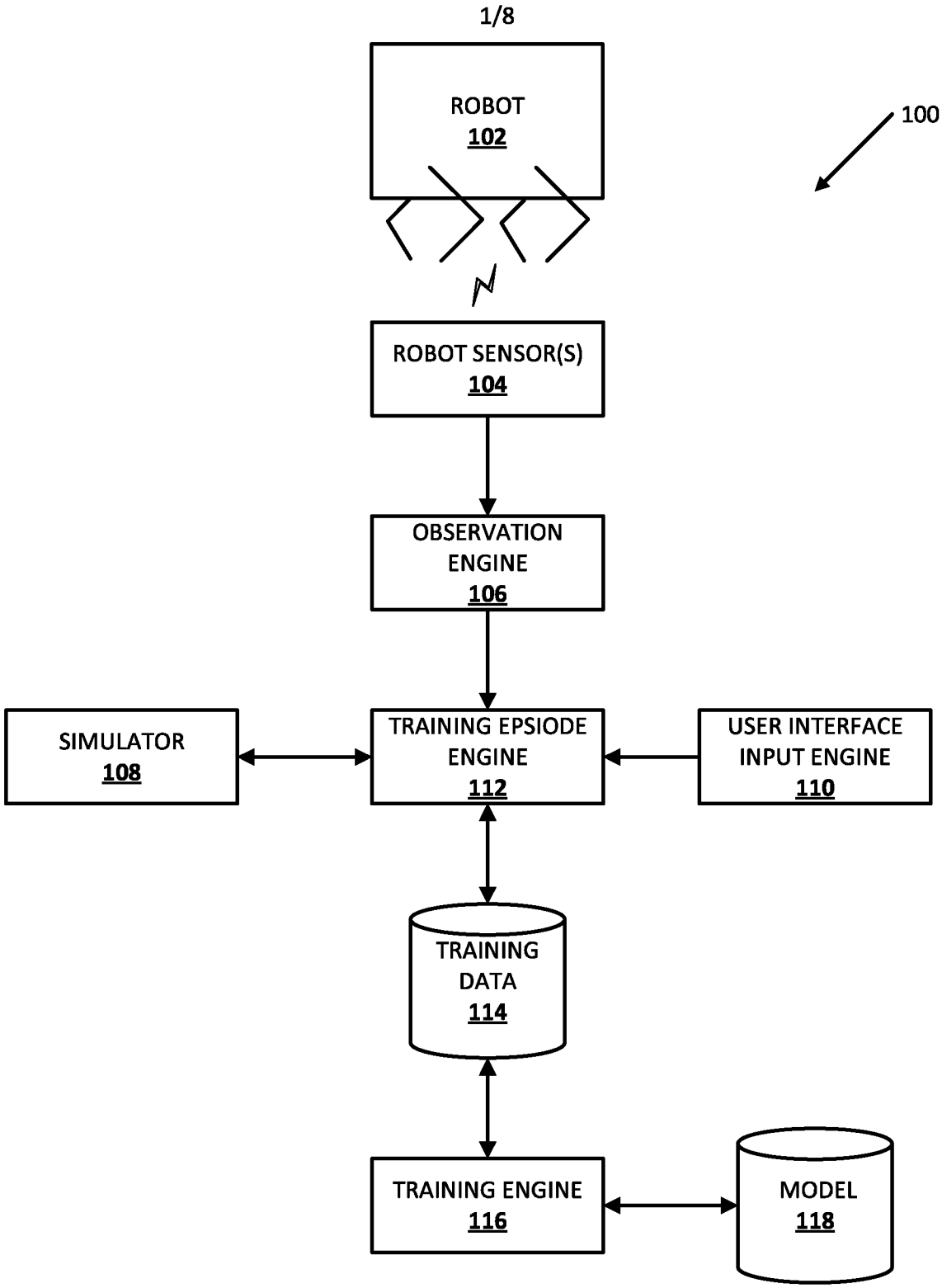
ROBOT
102
100
ROBOT SENSOR(S)
104
OBSERVATION ENGINE
106
SIMULATOR
108
TRAINING EPSIODE ENGINE
112
USER INTERFACE INPUT ENGINE
110
TRAINING DATA
114
TRAINING ENGINE
116
MODEL
118

FIG. 1

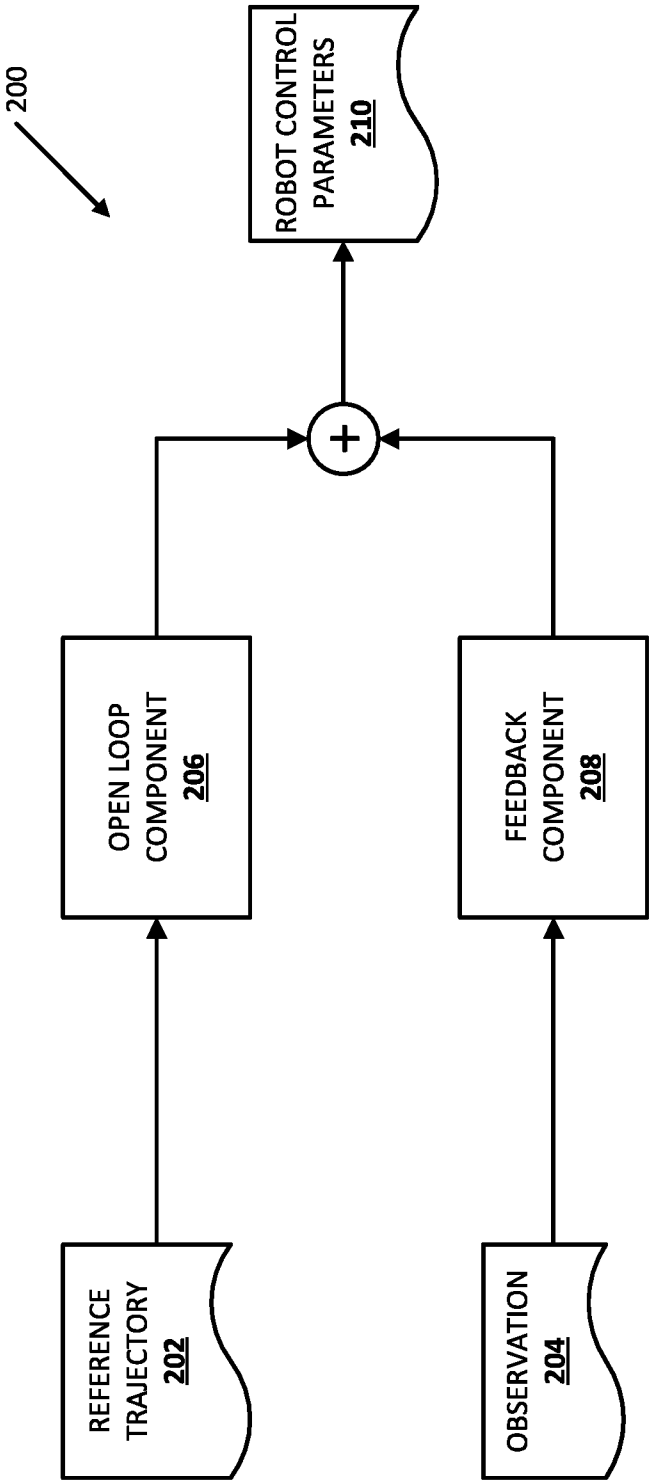
200
ROBOT CONTROL PARAMETERS 210
OPEN LOOP COMPONENT 206
FEEDBACK COMPONENT 208
REFERENCE TRAJECTORY 202
OBSERVATION 204

FIG. 2

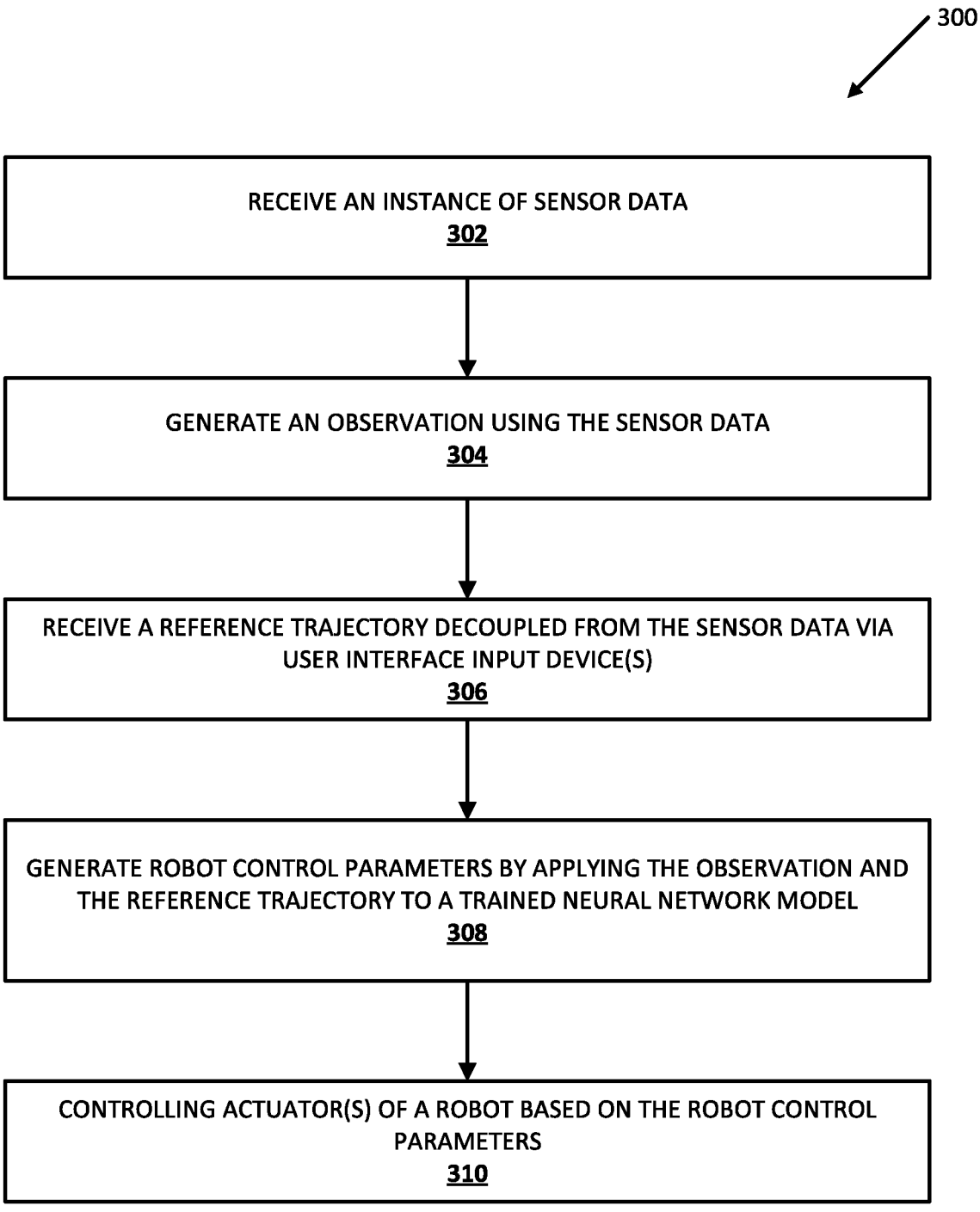
300
RECEIVE AN INSTANCE OF SENSOR DATA
302
GENERATE AN OBSERVATION USING THE SENSOR DATA
304
RECEIVE A REFERENCE TRAJECTORY DECOUPLED FROM THE SENSOR DATA VIA USER INTERFACE INPUT DEVICE(S)
306
GENERATE ROBOT CONTROL PARAMETERS BY APPLYING THE OBSERVATION AND THE REFERENCE TRAJECTORY TO A TRAINED NEURAL NETWORK MODEL
308
CONTROLLING ACTUATOR(S) OF A ROBOT BASED ON THE ROBOT CONTROL PARAMETERS
310

FIG. 3

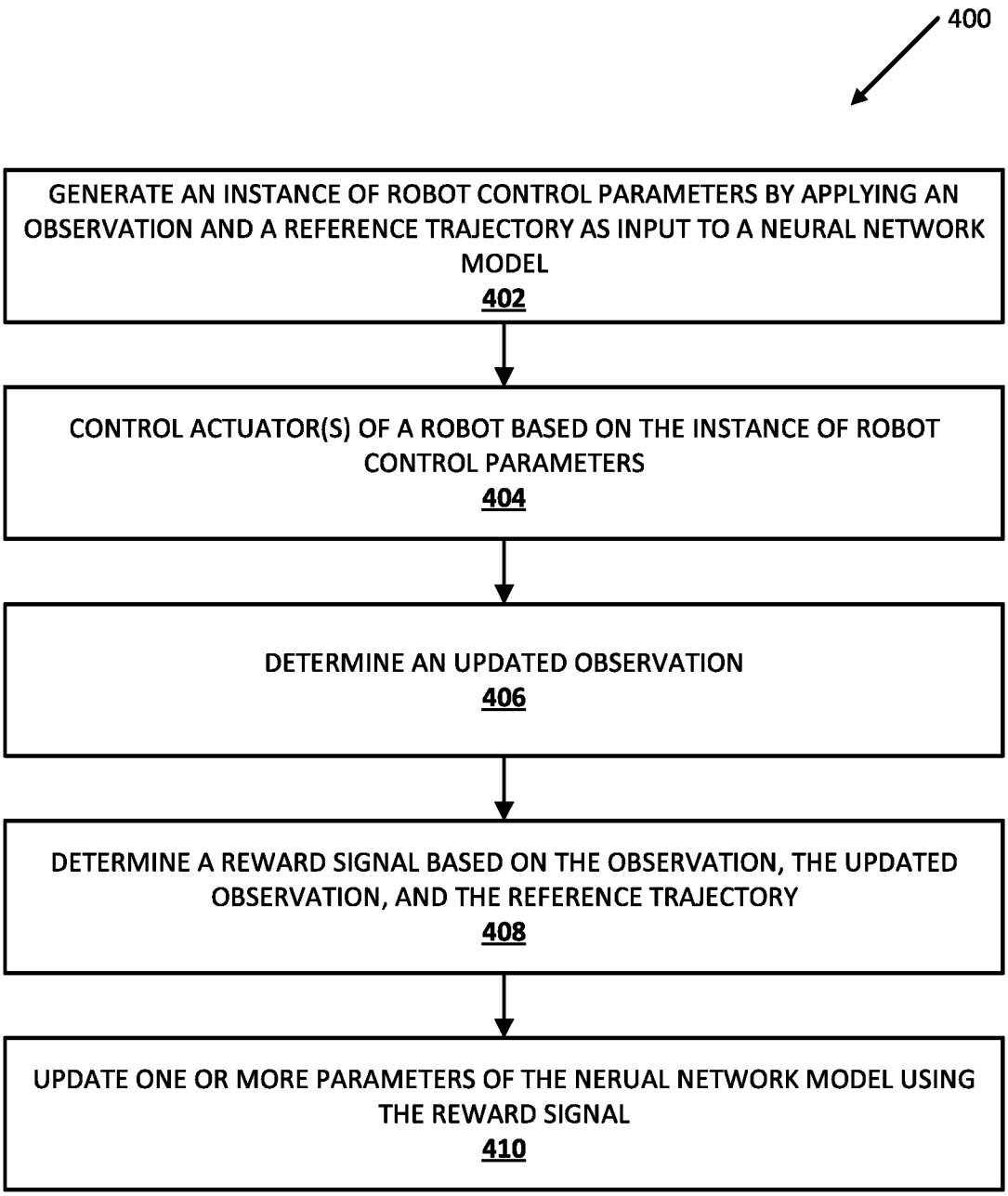
400
GENERATE AN INSTANCE OF ROBOT CONTROL PARAMETERS BY APPLYING AN OBSERVATION AND A REFERENCE TRAJECTORY AS INPUT TO A NEURAL NETWORK MODEL
402
CONTROL ACTUATOR(S) OF A ROBOT BASED ON THE INSTANCE OF ROBOT CONTROL PARAMETERS
404
DETERMINE AN UPDATED OBSERVATION
406
DETERMINE A REWARD SIGNAL BASED ON THE OBSERVATION, THE UPDATED OBSERVATION, AND THE REFERENCE TRAJECTORY
408
UPDATE ONE OR MORE PARAMETERS OF THE NERUAL NETWORK MODEL USING THE REWARD SIGNAL
410

FIG. 4

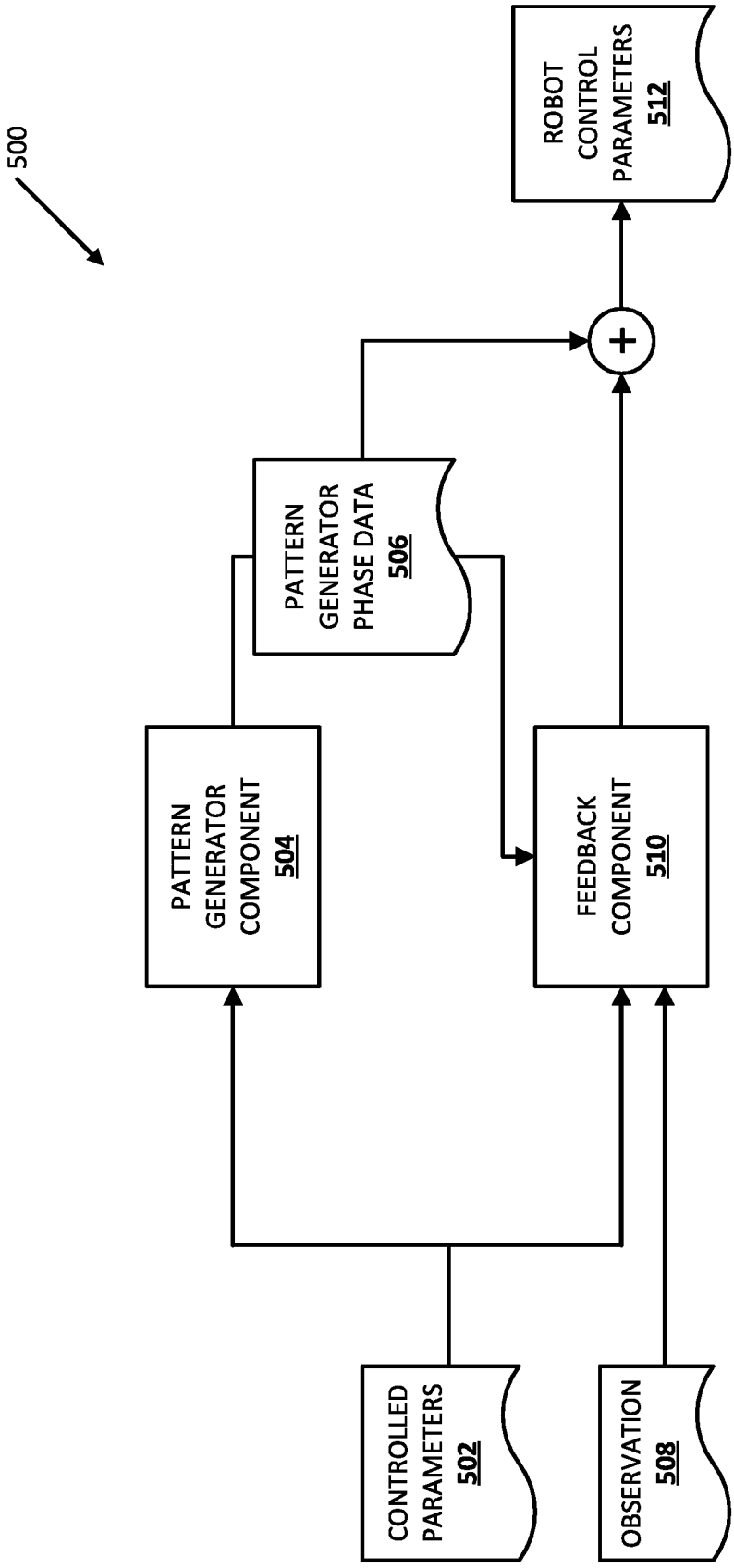
500
ROBOT CONTROL PARAMETERS 512
+
PATTERN GENERATOR PHASE DATA 506
PATTERN GENERATOR COMPONENT 504
FEEDBACK COMPONENT 510
CONTROLLED PARAMETERS 502
OBSERVATION 508

FIG. 5

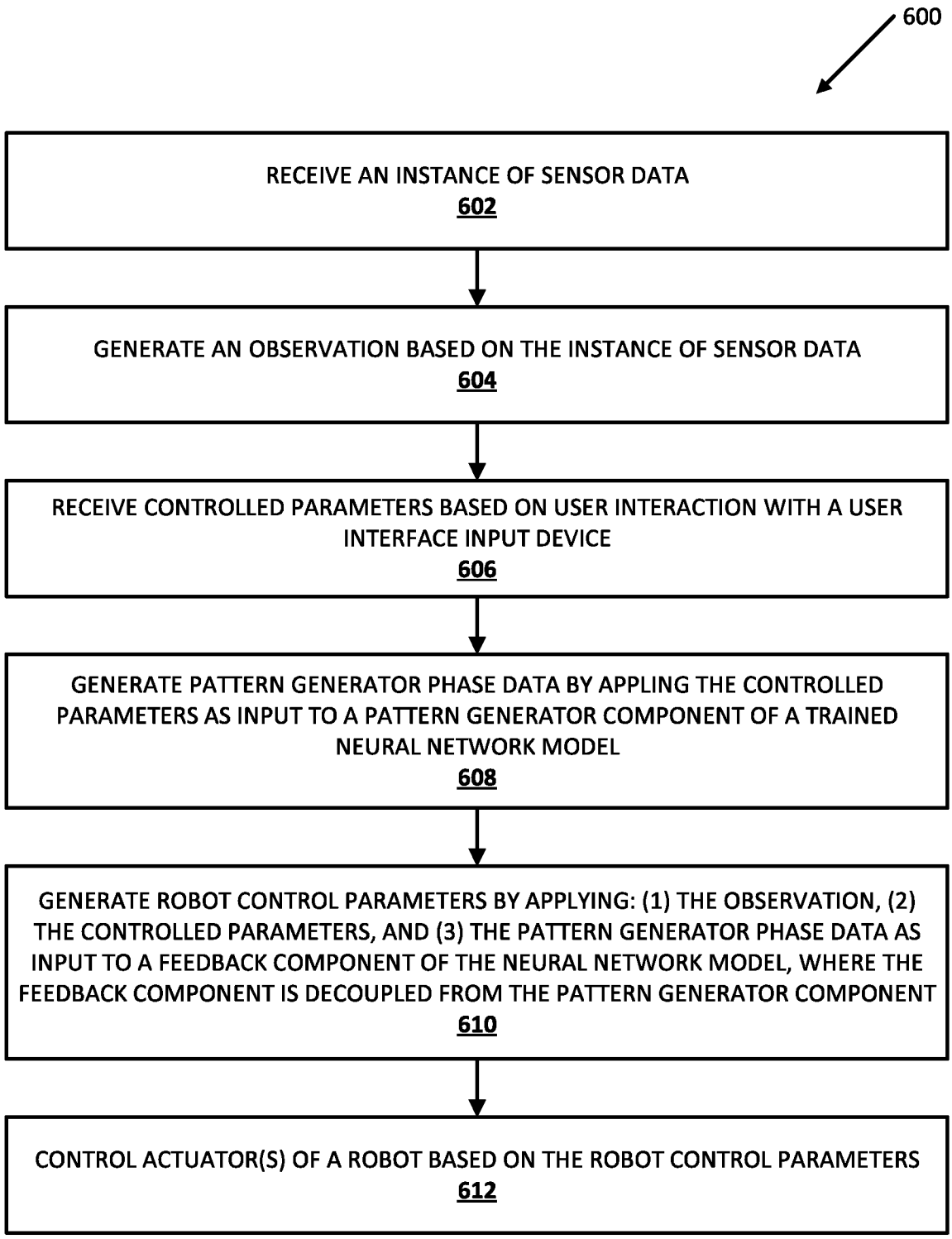
600
RECEIVE AN INSTANCE OF SENSOR DATA
602
GENERATE AN OBSERVATION BASED ON THE INSTANCE OF SENSOR DATA
604
RECEIVE CONTROLLED PARAMETERS BASED ON USER INTERACTION WITH A USER INTERFACE INPUT DEVICE
606
GENERATE PATTERN GENERATOR PHASE DATA BY APPLING THE CONTROLLED PARAMETERS AS INPUT TO A PATTERN GENERATOR COMPONENT OF A TRAINED NEURAL NETWORK MODEL
608
GENERATE ROBOT CONTROL PARAMETERS BY APPLYING: (1) THE OBSERVATION, (2) THE CONTROLLED PARAMETERS, AND (3) THE PATTERN GENERATOR PHASE DATA AS INPUT TO A FEEDBACK COMPONENT OF THE NEURAL NETWORK MODEL, WHERE THE FEEDBACK COMPONENT IS DECOUPLED FROM THE PATTERN GENERATOR COMPONENT
610
CONTROL ACTUATOR(S) OF A ROBOT BASED ON THE ROBOT CONTROL PARAMETERS
612

FIG. 6

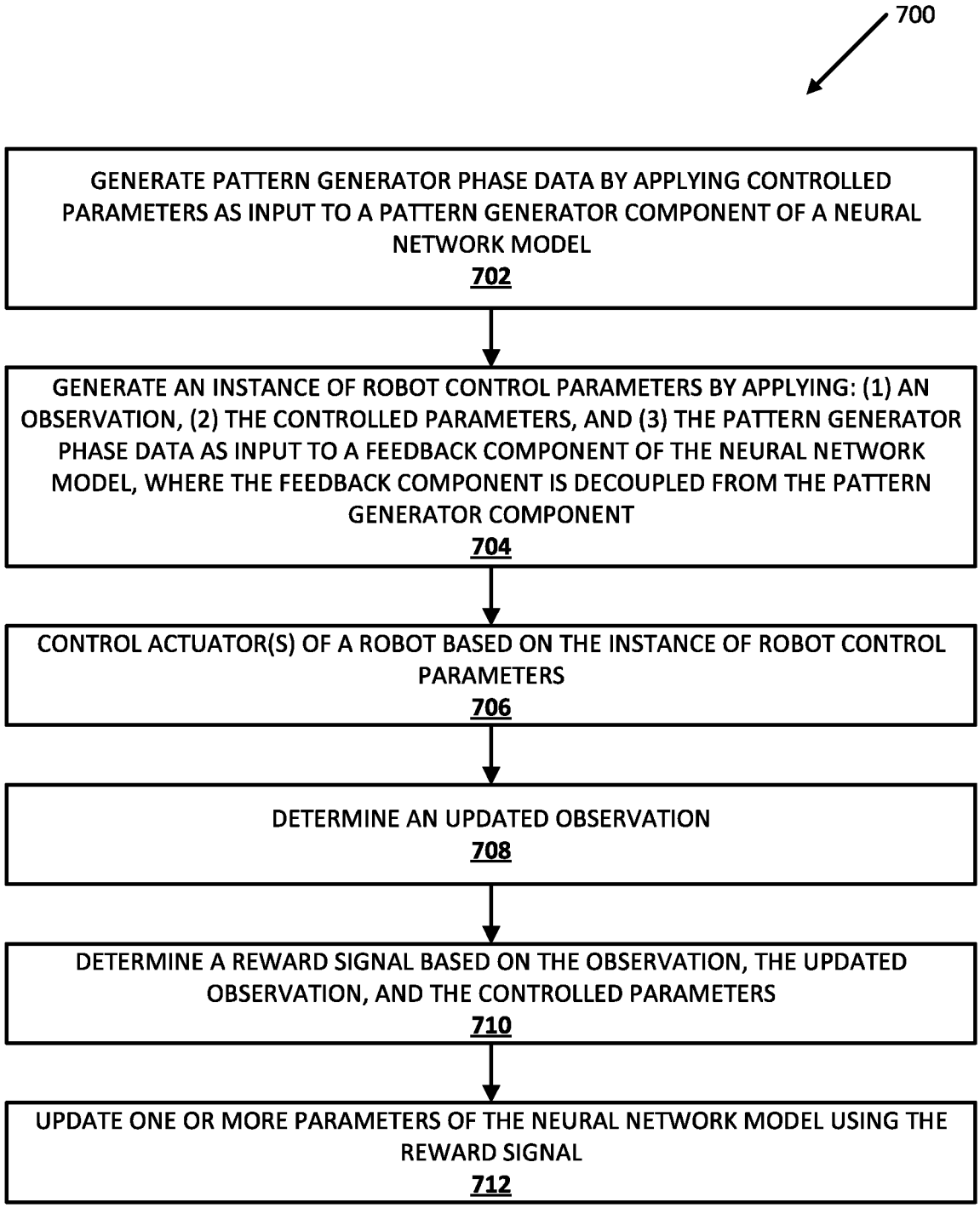
700
GENERATE PATTERN GENERATOR PHASE DATA BY APPLYING CONTROLLED PARAMETERS AS INPUT TO A PATTERN GENERATOR COMPONENT OF A NEURAL NETWORK MODEL
702
GENERATE AN INSTANCE OF ROBOT CONTROL PARAMETERS BY APPLYING: (1) AN OBSERVATION, (2) THE CONTROLLED PARAMETERS, AND (3) THE PATTERN GENERATOR PHASE DATA AS INPUT TO A FEEDBACK COMPONENT OF THE NEURAL NETWORK MODEL, WHERE THE FEEDBACK COMPONENT IS DECOUPLED FROM THE PATTERN GENERATOR COMPONENT
704
CONTROL ACTUATOR(S) OF A ROBOT BASED ON THE INSTANCE OF ROBOT CONTROL PARAMETERS
706
DETERMINE AN UPDATED OBSERVATION
708
DETERMINE A REWARD SIGNAL BASED ON THE OBSERVATION, THE UPDATED OBSERVATION, AND THE CONTROLLED PARAMETERS
710
UPDATE ONE OR MORE PARAMETERS OF THE NEURAL NETWORK MODEL USING THE REWARD SIGNAL
712

FIG. 7

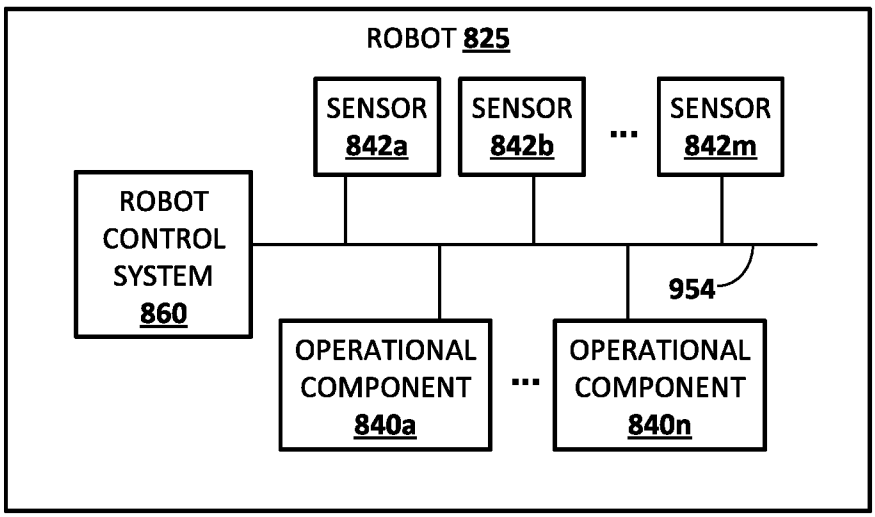
ROBOT 825
SENSOR 842a
SENSOR 842b
SENSOR 842m
ROBOT CONTROL SYSTEM 860
954
OPERATIONAL COMPONENT 840a
OPERATIONAL COMPONENT 840n

FIG. 8

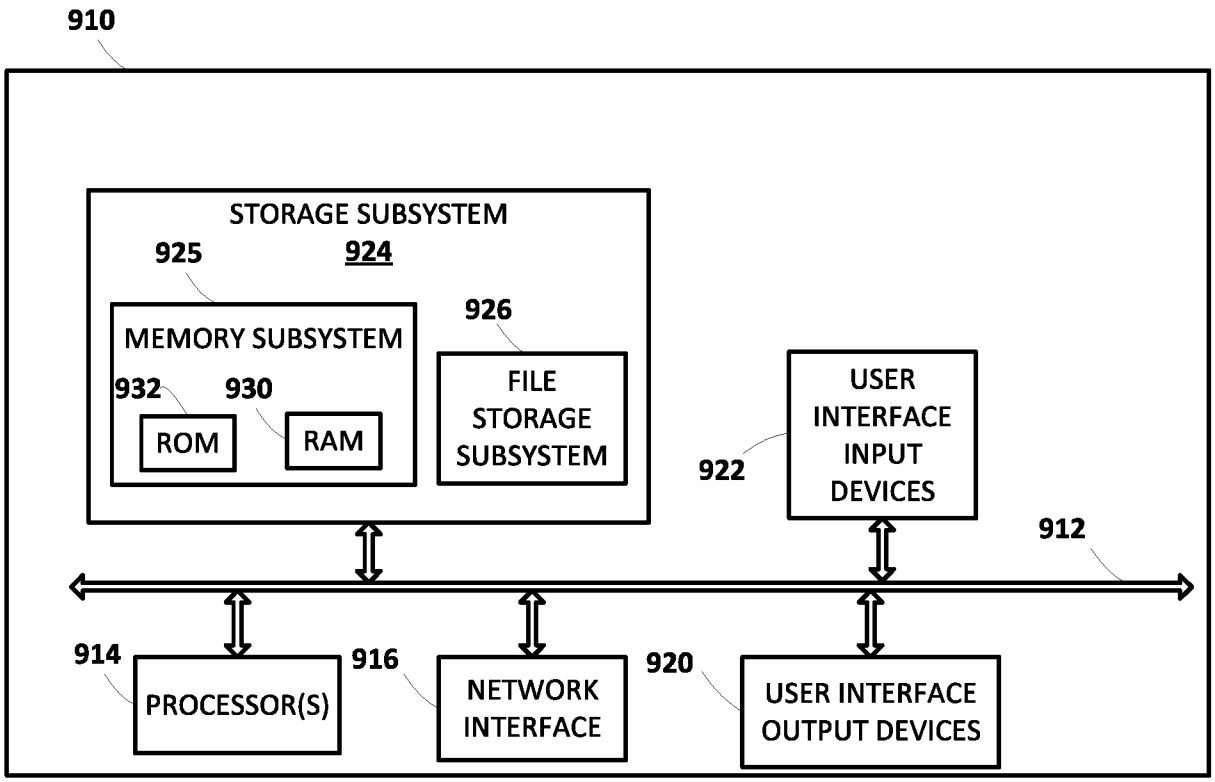
910
STORAGE SUBSYSTEM 924
925
MEMORY SUBSYSTEM
932
ROM
930
RAM
926
FILE STORAGE SUBSYSTEM
922
USER INTERFACE INPUT DEVICES
912
914
PROCESSOR(S)
916
NETWORK INTERFACE
920
USER INTERFACE OUTPUT DEVICES

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/028454

A. CLASSIFICATION OF SUBJECT MATTER
INV. B25J9/16
ADD. B62D57/02

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)
B25J B62D

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DUSKO M KATIC ET AL: "Hybrid Dynamic Control Algorithm for Humanoid Robots Based on Reinforcement Learning", JOURNAL OF INTELLIGENT AND ROBOTIC SYSTEMS ; THEORY AND APPLICATIONS - (INCORPORATING MECHATRONIC SYSTEMS ENGINEERING), KLUWER ACADEMIC PUBLISHERS, DO, vol. 51, no. 1, 25 September 2007 (2007-09-25), pages 3-30, XP019552842, ISSN: 1573-0409 figures 2,3,4,5 page 12 - page 17 ----- -/--	1-3, 14-20

[X] 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

"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
28 August 2019	13/09/2019

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Antonopoulos, A

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/028454

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ARAO M ET AL: "Flexible intelligent system based on fuzzy neural networks and reinforcement learning", FUZZY SYSTEMS, 1995. INTERNATIONAL JOINT CONFERENCE OF THE FOURTH IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS AND THE SECOND INTERNATIONA L FUZZY ENGINEERING SYMPOSIUM., PROCEEDINGS OF 1995 IEEE INTERNATIONAL CONFERENCE ON YOKOHAMA, JAPAN 20-24 MA, vol. 5, 20 March 1995 (1995-03-20), pages 69-70, XP010143970, DOI: 10.1109/FUZZY.1995.410045 ISBN: 978-0-7803-2461-9 the whole document -----	1-20