US 20240265659A1

(19) **United States**

(12) **Patent Application Publication**

**ALI AKBARIAN et al.**

(10) **Pub. No.: US 2024/0265659 A1**

(43) **Pub. Date: Aug. 8, 2024**

(54) # UPDATING POSE OF AN ARTICULATED OBJECT

(71) Applicant: **Microsoft Technology Licensing, LLC**, Redmond, WA (US)

(72) Inventors: **Mohammand Sadegh ALI AKBARIAN**, Cambridge (GB); **Tadas BALTRUSAITIS**, Cambridge (GB)

(21) Appl. No.: **18/403,709**

(22) Filed: **Jan. 3, 2024**

**Related U.S. Application Data**

(63) Continuation-in-part of application No. 18/164,391, filed on Feb. 3, 2023.

**Publication Classification**

(51) **Int. Cl.**

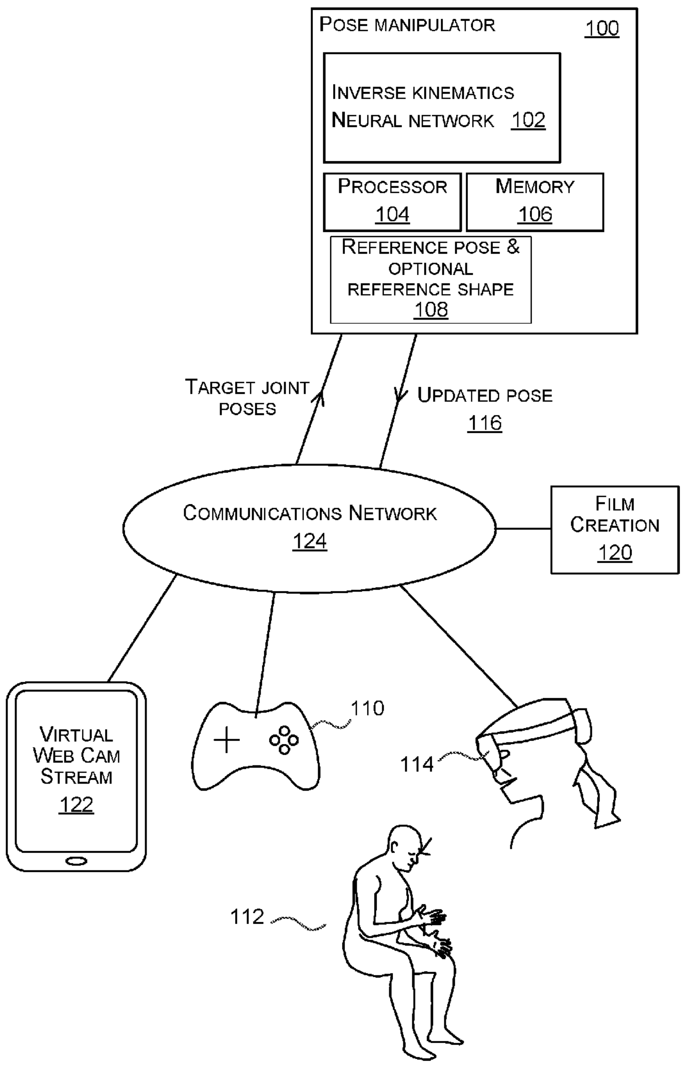
| | |
|---|---|
| ***G06T 19/20*** | (2006.01) |
| ***G06T 7/73*** | (2006.01) |

(52) **U.S. Cl.**

CPC ................ ***G06T 19/20*** (2013.01); ***G06T 7/73*** (2017.01); *G06T 2200/24* (2013.01); *G06T 2207/20081* (2013.01); *G06T 2207/20084* (2013.01); *G06T 2219/2004* (2013.01); *G06T 2219/2021* (2013.01)

(57) **ABSTRACT**

A method of updating a pose of a plurality of joints of a kinematic tree of an articulated object is described. The method comprises receiving, for each of the joints in the kinematic tree, an initial pose. A single first embedding vector is computed by encoding the initial poses in an embedding space. For each of some but not all of the joints in the kinematic tree, a target pose is received. A single second embedding vector representing the target poses is computed in the embedding space. The first embedding vector is modified using the second embedding vector to form a third embedding vector. Decoding the third embedding vector produces the updated pose of the articulated object.


POSE MANIPULATOR 100
INVERSE KINEMATICS NEURAL NETWORK 102
PROCESSOR 104
MEMORY 106
REFERENCE POSE & OPTIONAL REFERENCE SHAPE 108
TARGET JOINT POSES
UPDATED POSE 116
COMMUNICATIONS NETWORK 124
FILM CREATION 120
VIRTUAL WEB CAM STREAM 122
110
114
112

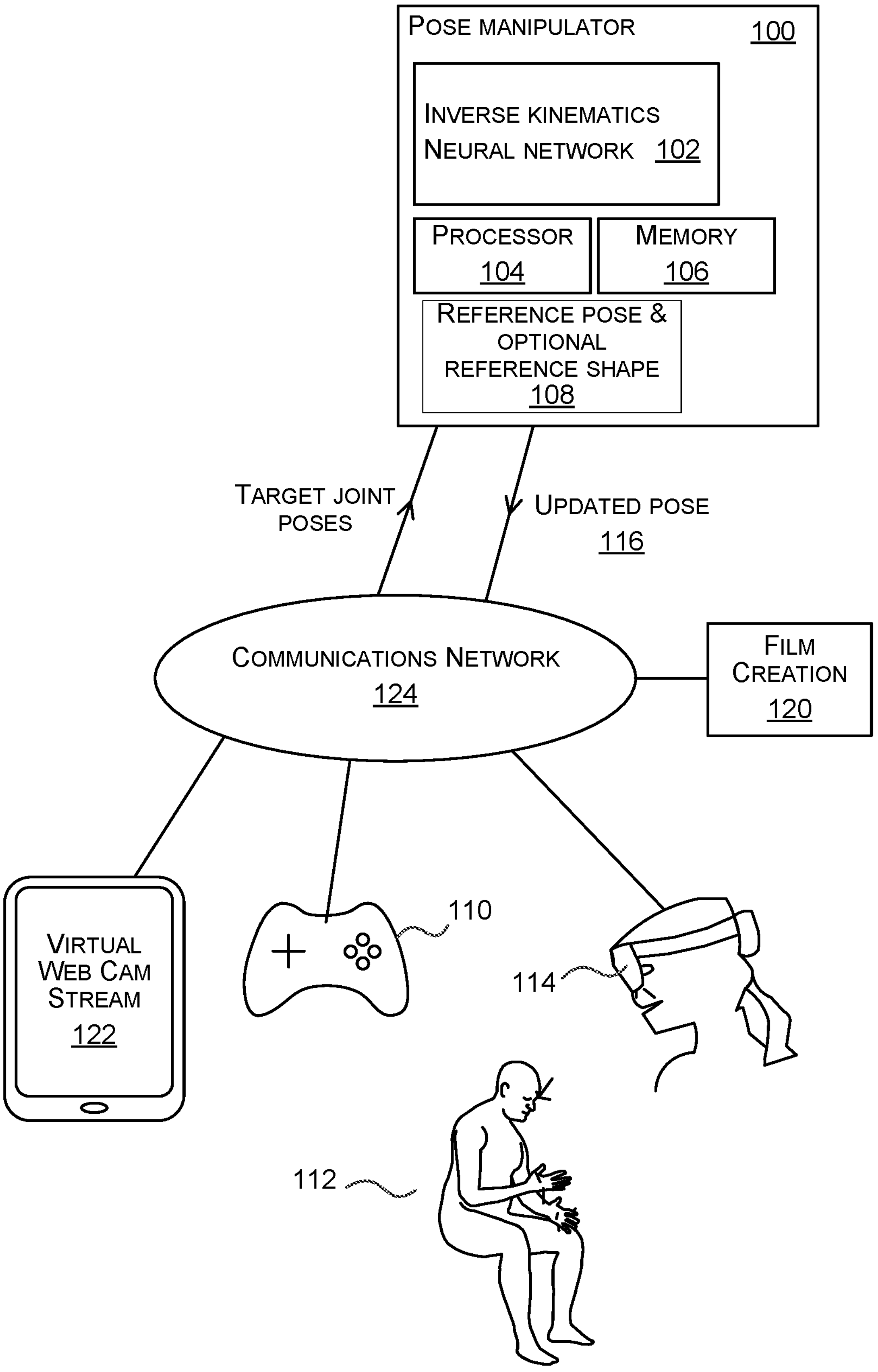

Pose manipulator 100
Inverse kinematics neural network 102
Processor 104
Memory 106
Reference pose & optional reference shape 108
Target joint poses
Updated pose 116
Communications Network 124
Film Creation 120
Virtual Web Cam Stream 122
110
114
112

FIG. 1

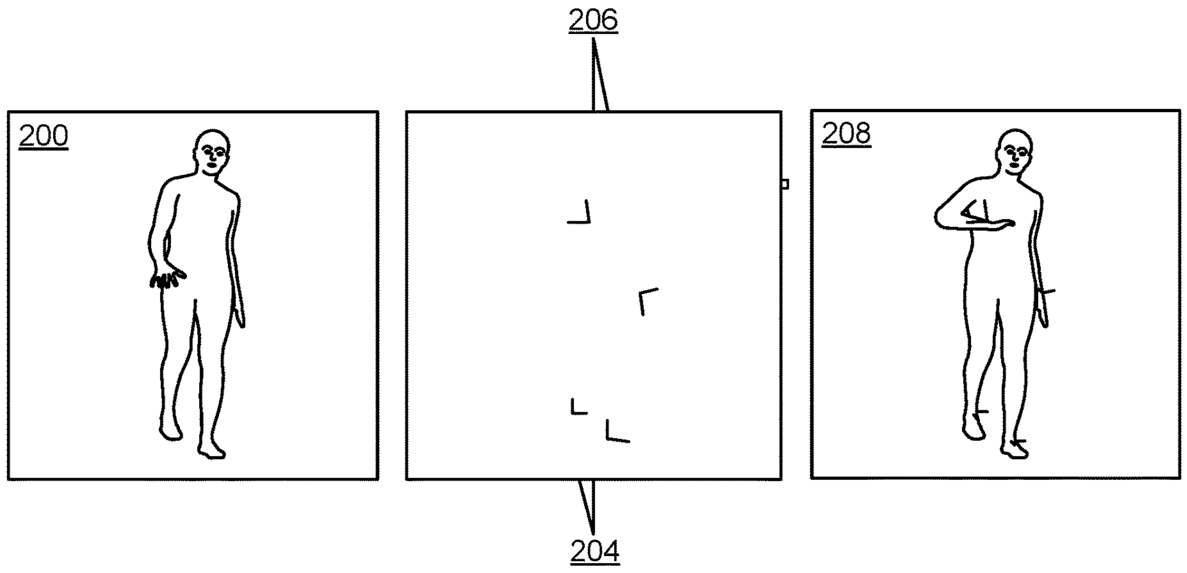


FIG. 2

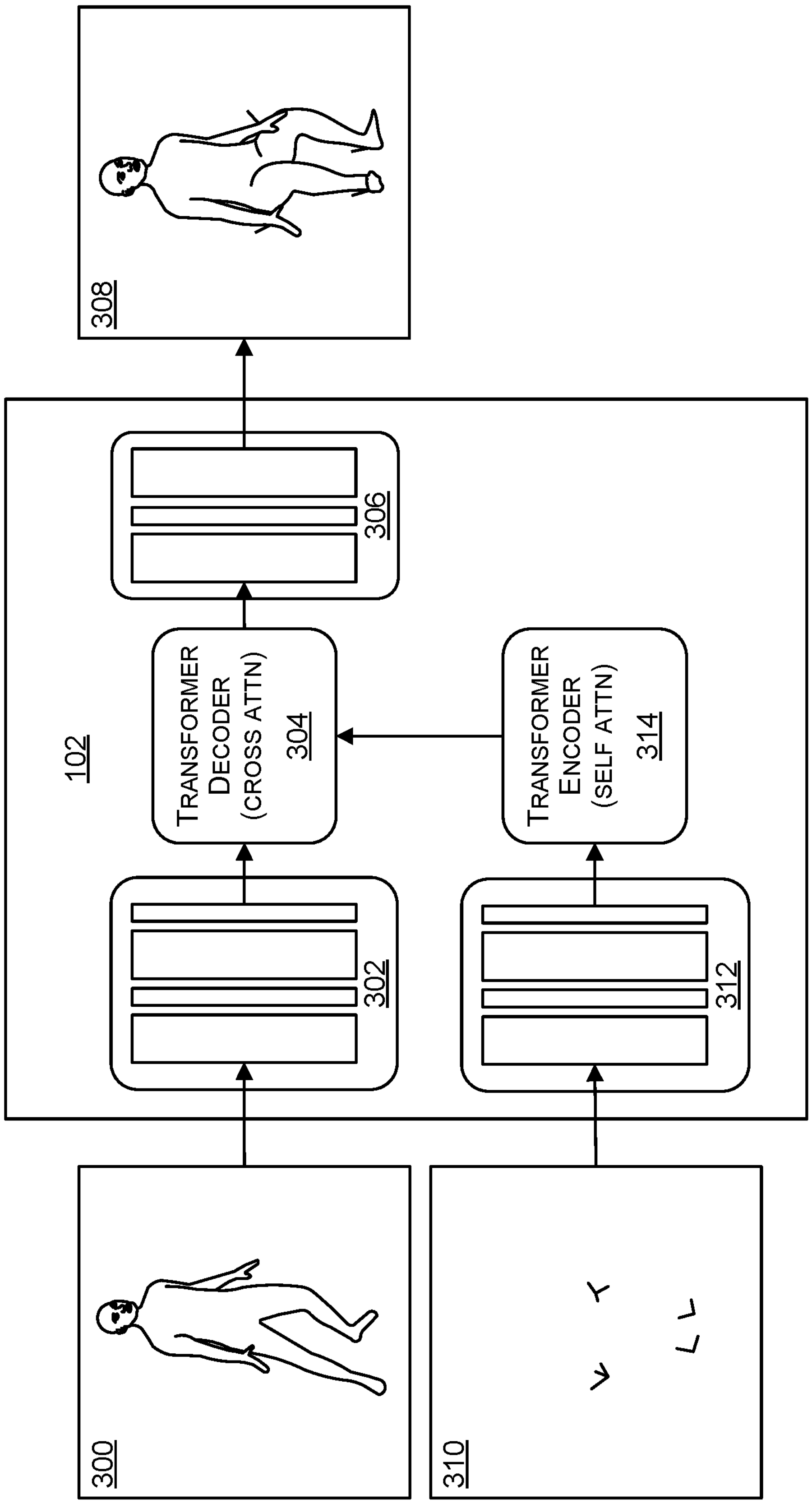

300
310
302
312
102
TRANSFORMER DECODER (CROSS ATTN)
304
TRANSFORMER ENCODER (SELF ATTN)
314
306
308

FIG. 3

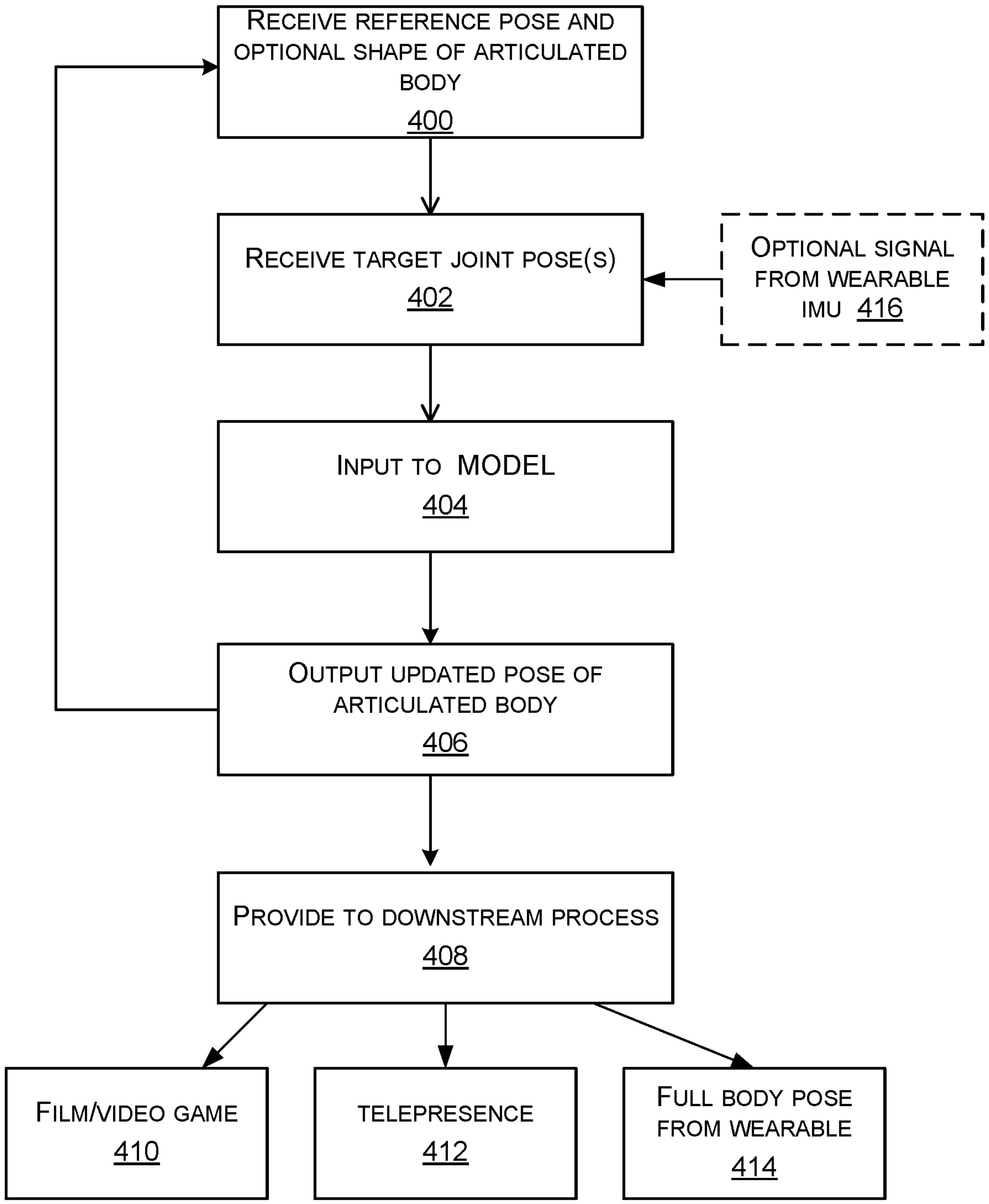
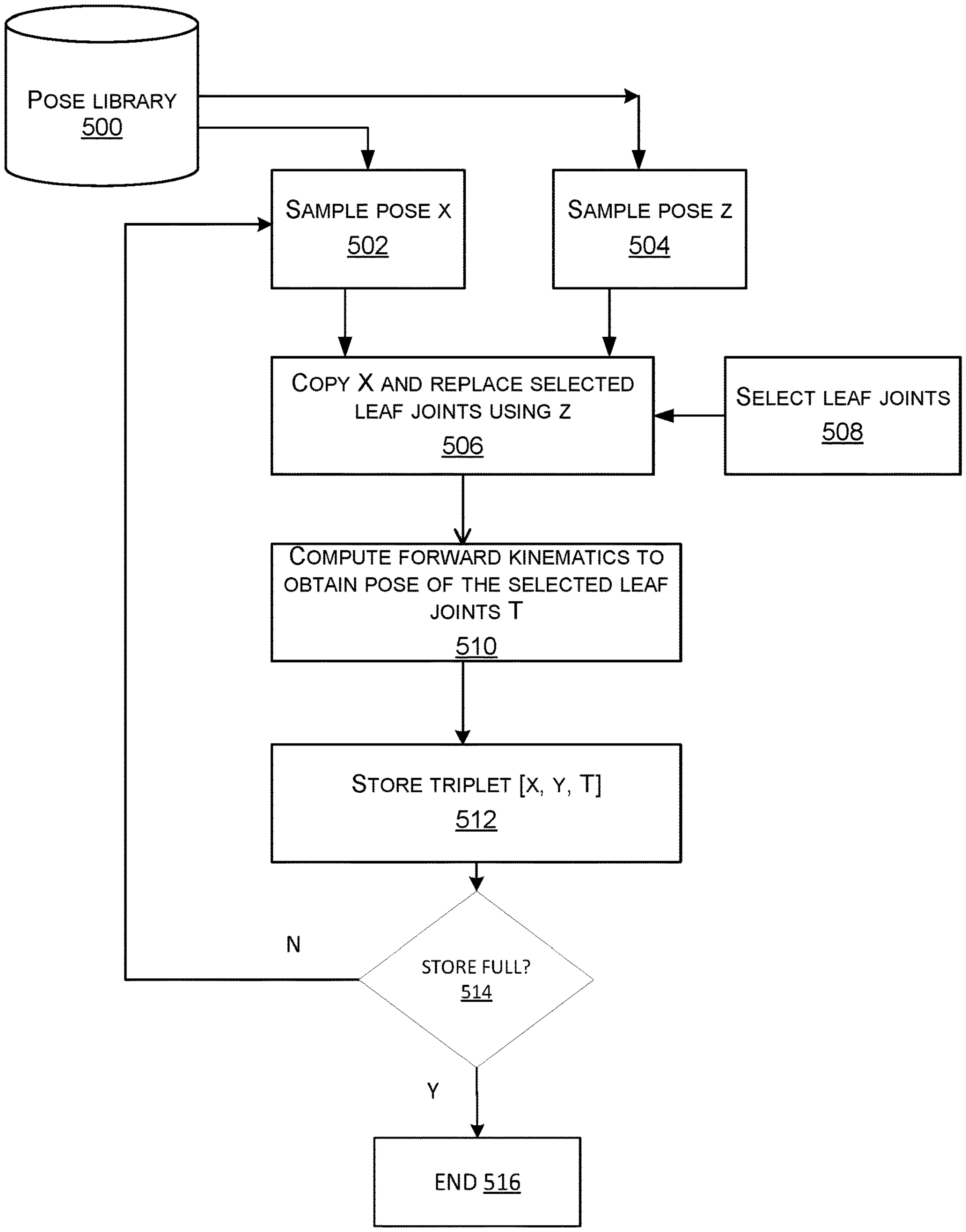

RECEIVE REFERENCE POSE AND OPTIONAL SHAPE OF ARTICULATED BODY 400
RECEIVE TARGET JOINT POSE(S) 402
OPTIONAL SIGNAL FROM WEARABLE IMU 416
INPUT TO MODEL 404
OUTPUT UPDATED POSE OF ARTICULATED BODY 406
PROVIDE TO DOWNSTREAM PROCESS 408
FILM/VIDEO GAME 410
TELEPRESENCE 412
FULL BODY POSE FROM WEARABLE 414

FIG. 4



POSE LIBRARY 500
SAMPLE POSE X 502
SAMPLE POSE Z 504
COPY X AND REPLACE SELECTED LEAF JOINTS USING Z 506
SELECT LEAF JOINTS 508
COMPUTE FORWARD KINEMATICS TO OBTAIN POSE OF THE SELECTED LEAF JOINTS T 510
STORE TRIPLET [X, Y, T] 512
STORE FULL? 514
N
Y
END 516

FIG. 5

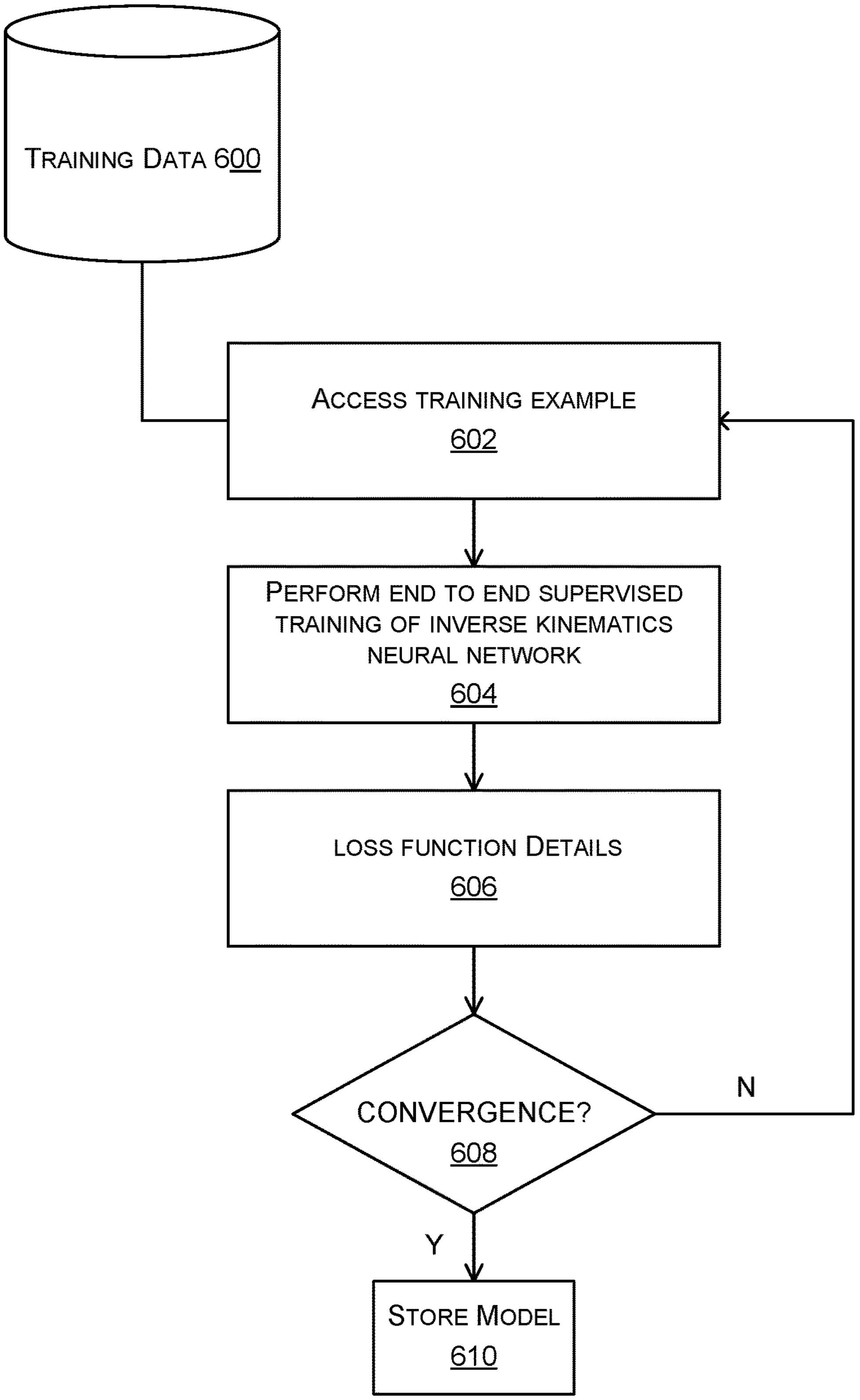

TRAINING DATA 600
ACCESS TRAINING EXAMPLE
602
PERFORM END TO END SUPERVISED TRAINING OF INVERSE KINEMATICS NEURAL NETWORK
604
LOSS FUNCTION DETAILS
606
CONVERGENCE?
608
N
Y
STORE MODEL
610

FIG. 6

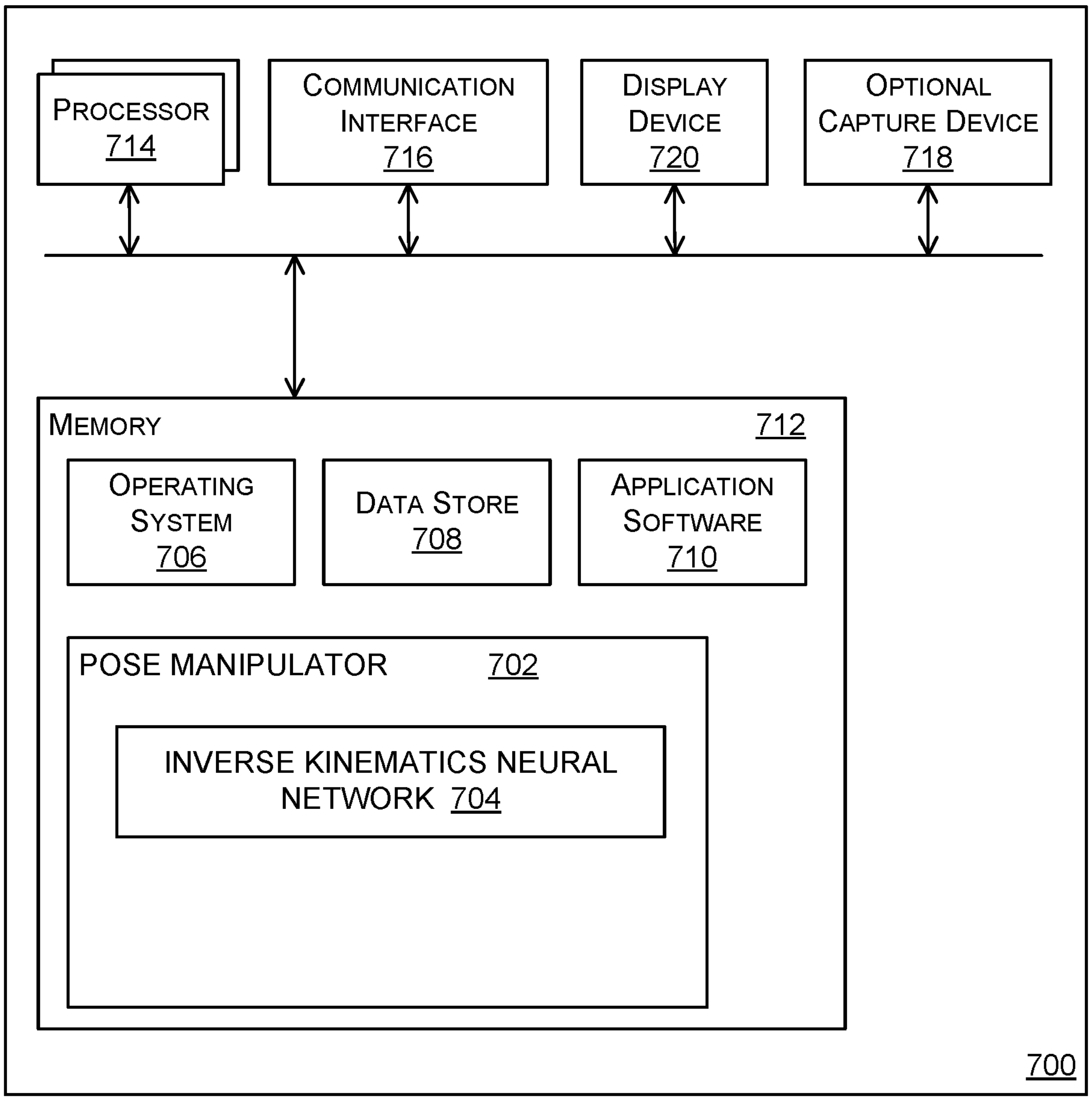

PROCESSOR 714
COMMUNICATION INTERFACE 716
DISPLAY DEVICE 720
OPTIONAL CAPTURE DEVICE 718
MEMORY 712
OPERATING SYSTEM 706
DATA STORE 708
APPLICATION SOFTWARE 710
POSE MANIPULATOR 702
INVERSE KINEMATICS NEURAL NETWORK 704
700

FIG. 7

# UPDATING POSE OF AN ARTICULATED OBJECT

## CROSS-REFERENCE TO RELATED APPLICATION

**[0001]** This application is a continuation-in-part application of and claims priority to U.S. patent application Ser. No. 18/164,391, entitled "PREDICTING BODY MOTION," filed on Feb. 3, 2023, the disclosure of which is incorporated herein by reference in its entirety.

## BACKGROUND

**[0002]** Articulated objects such as the human body, motor vehicles, laptop computers, animals and other articulated objects are often represented in computer processes by storing poses of joints of a kinematic tree. A kinematic tree is a plurality of joints connected together by rigid bodies such as bones of a skeleton or parts of a motor vehicle. A pose of the articulated entity may be recorded by recoding a pose (position and orientation) of each of the joints of the kinematic tree. Updating the pose in the light of sensor data about the articulated object is not straightforward.

**[0003]** The embodiments described below are not limited to implementations which solve any or all of the disadvantages of known ways of updating pose of an articulated object.

## SUMMARY

**[0004]** The following presents a simplified summary of the disclosure in order to provide a basic understanding to the reader. This summary is not intended to identify key features or essential features of the claimed subject matter nor is it intended to be used to limit the scope of the claimed subject matter. Its sole purpose is to present a selection of concepts disclosed herein in a simplified form as a prelude to the more detailed description that is presented later.

**[0005]** Updating a pose of an articulated object using a kinematic tree is useful for a variety of purposes including but not limited to: animating avatars, completing the pose of a full human body when only part of the pose is visible, full body pose estimation using signals from inertial measurement units worn only on wrists and ankles.

**[0006]** A method of updating a pose of a plurality of joints of a kinematic tree of an articulated object is described. The method comprises receiving, for each of the joints in the kinematic tree, an initial pose. A single first embedding vector is computed by encoding the initial poses in an embedding space. For each of some but not all of the joints in the kinematic tree, a target pose is received. A single second embedding vector representing the target poses is computed in the embedding space. The first embedding vector is modified using the second embedding vector to form a third embedding vector. Decoding the third embedding vector produces the updated pose of the articulated object.

**[0007]** Many of the attendant features will be more readily appreciated as the same becomes better understood by reference to the following detailed description considered in connection with the accompanying drawings.

## DESCRIPTION OF THE DRAWINGS

**[0008]** The present description will be better understood from the following detailed description read in light of the accompanying drawings, wherein:

**[0009]** FIG. **1** shows a pose manipulator deployed as a cloud service:

**[0010]** FIG. **2** shows an initial pose of a kinematic tree, target poses of four joints, and an updated pose of the kinematic tree:

**[0011]** FIG. **3** shows an architecture of an example inverse kinematics neural network:

**[0012]** FIG. **4** is a flow diagram of a method of updating a pose using a pose manipulator such as that of FIG. **1** or **3**:

**[0013]** FIG. **5** is a flow diagram of a method of creating training data to train a pose manipulator:

**[0014]** FIG. **6** is a flow diagram of a method of training a pose manipulator;

**[0015]** FIG. **7** illustrates an exemplary computing-based device in which examples of a pose manipulator are implemented.

**[0016]** Like reference numerals are used to designate like parts in the accompanying drawings.

## DETAILED DESCRIPTION

**[0017]** The detailed description provided below in connection with the appended drawings is intended as a description of the present examples and is not intended to represent the only forms in which the present examples are constructed or utilized. The description sets forth the functions of the examples and the sequence of operations for constructing and operating the examples. However, the same or equivalent functions and sequences may be accomplished by different examples.

**[0018]** As mentioned above, updating the pose of an articulated object in the light of sensor data about the object (which may be sparse) is difficult. One approach is to use inverse kinematics. Inverse kinematics is where the pose of one or more end effectors of a kinematic tree are known, and it is desired to update the rest of the kinematic tree to be consistent with the pose of the end effectors. An end effector is a leaf node of a kinematic tree, such as an ankle or wrist in the case of a human skeleton. One of the reasons that inverse kinematics is difficult is that there are constraints on how the articulated entity can be arranged and these constraints are difficult and complex to express mathematically using rules. In an example, where the kinematic tree is a human skeleton, the constraints include things like, the foot cannot go behind the back. Traditional methods for solving inverse kinematics problems often involve complex mathematical formulations or require artists and specialized riggers involvement to create rigs (where a rig is a set of constraints and relationships between different parts of an object, such as wrist can only bend in particular ways and palm cannot intersect with fingers) that inverse kinematics can work on to achieve natural looking and plausible poses, and may not always guarantee optimal or real-time solutions.

**[0019]** The present technology presents an innovative deep learning based approach to approximating inverse kinematics for real time and accurate pose manipulation. By harnessing the power of a neural network with attention mechanism, neural networks can efficiently and accurately estimate joint angles given an arbitrary number of target joint positions and rotations. A novel data augmentation regime is described to create training data for training the neural network. The resulting system can seamlessly be utilized as a real time inverse kinematics solution that enables non-artists to be able to manipulate poses. It also enables other tasks such as motion generation driven by wearable devices.

**[0020]** FIG. **1** is a schematic diagram of a pose manipulator **100** deployed in the cloud. The pose manipulator is for use with one class or type of kinematic tree, such as a human body, or a motor vehicle. It is computer implemented and comprises an inverse kinematics neural network **102**, a processor **104**, a memory **106**, and an initial pose and optional reference shape **108**. The pose manipulator **100** receives as input one or more target joint poses. It returns an updated pose **116** of all the joints in the kinematic tree.

**[0021]** The pose manipulator **100** receives the target poses via communications network **124** from a client device. A non-exhaustive list of types of client device which may be used is: game console **110**, smart phone **122**, head worn computer **114**, file creation tool **120**. Each target pose is a 3D position and 3D orientation (6 degree of freedom pose). The updated pose is a 3D position and 3D orientation for each joint of the kinematic tree. In some examples the updated pose of the articulated object is an approximation of inverse kinematics applied to the kinematic tree using the target poses.

**[0022]** It is not essential for the pose manipulator **100** to be deployed as a cloud service. In some cases the pose manipulator is functionality deployed in an end user computing device such as a smart phone, laptop computer, wearable computer or desktop computer. The functionality of the pose manipulator **100** is distributed between more than one entity in some cases.

**[0023]** The pose manipulator updates a pose of a plurality of joints of a kinematic tree of an articulated object. This is useful for a variety of purposes such as film creation, avatar manipulation, full body pose estimation from sparse sensor data and more. The method comprises receiving, for each of the joints in the kinematic tree, an initial pose. Having an initial pose gives the method something to start from. The initial pose is not only to have a place to start from (e.g. the method could start from fully neutral standing pose with arms down by the hips), but also to constrain the end outcome. In some examples it is desired for the end pose to be like the start pose with exception that a joint position is changed, such as a hand is raised for example. The method computes a single first embedding vector by encoding the initial poses in an embedding space. The first embedding vector is a concise and hence efficient way to represent the initial poses. For each of some but not all of the joints in the kinematic tree, a target pose is received. The method is able to receive the target pose from user input or from sensor data or from another computing process. The method computes a single second embedding vector representing the target poses in the embedding space. The second embedding vector is a concise representation which improves efficiency. The process comprises modifying the first embedding vector using the second embedding vector to form a third embedding vector. Because the first and second embedding vectors are in the same space it is possible to modify the first embedding vector (representing the initial poses) in the light of the target poses (second embedding vector). This provides a convenient and effective way to update the pose that gives accurate results which are plausible (agree with ways the articulated entity can be manipulated naturally). There is no need to use complex rules to express constraints. The method decodes the third embedding vector to produce the updated pose of the articulated object. The third embedding vector is a concise representation that can be sent with low bandwidth. It can be decoded to produce the position and orientation values for each of the joints in the kinematic tree.

**[0024]** Being able to encode the initial poses and the target poses into the same embedding space enables the target poses to be used to update the initial poses in an efficient and effective manner.

**[0025]** The inverse kinematics neural network improves the functioning of the underlying computing device by enabling target poses to be used to update initial poses in an efficient and effective manner.

**[0026]** Alternatively, or in addition, the functionality of the pose manipulator described herein is performed, at least in part, by one or more hardware logic components. For example, and without limitation, illustrative types of hardware logic components that are optionally used include Field-programmable Gate Arrays (FPGAs), Application-specific Integrated Circuits (ASICs), Application-specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), Graphics Processing Units (GPUs).

**[0027]** The task of the pose manipulator **100** is to generate a novel pose y given an initial pose x and an arbitrary number of targets T=[j1, j2 . . . ] such that the novel pose is a modified version of the initial pose that closely matches the targets. The poses x and y are defined with joint rotations of joints in the body kinematic tree. Each target is a 6-DoF representation of an end-effector joint (i.e., the leaves of the kinematic tree).

**[0028]** FIG. **2** shows an initial pose **200** of a kinematic tree (also referred to as a reference pose), target poses of four joints **204**, **206**, and an updated pose **208** of the kinematic tree. The initial pose is a standing human body. The target poses are poses of ankle joints and wrist joints of the human body where only the right wrist has moved as compared to the initial pose **200**. The updated pose **208** shows the standing human body with the right hand raised and the right arm and shoulder pose adjusted to agree and where the rest of the joints of the body are unchanged.

**[0029]** FIG. **3** shows an architecture of an example inverse kinematics neural network **102**. It comprises a first encoder **302**, a second encoder **312**, a decoder **306** and a transformer comprising a transformer encoder **314** and a transformer decoder **304**. In this example the architecture comprises five components, all trained end-to-end: an initial pose embedding module (the first encoder **302**), a target embedding module (the second encoder **312**), a self-attention mechanism (the transformer encoder **314**) to learn the relation between various components of the target signal, a cross-attention module (the transformer decoder **304**) to learn how to manipulate the initial pose given the target signal, and finally a pose decoder (decoder **306**) to generate the resulting pose.

**[0030]** The first encoder **302** computes an embedding representation of the initial pose that is then consumable by the transformer decoder **304** (cross-attention module). The first encoder receives as input the initial pose **300**, where the pose global orientation is considered fixed. The pose is defined as the local rotation of all joints in the body, each represented relative to its parent according to the kinematic tree of human skeleton. This is a standard representation of the human pose. For a pose of N joints, each joint is represented with the 6D rotation representation, resulting a N×6 tensor. Along with this representation, additionally provide the global rotation and position of each joint. This is achieved by computing the forward kinematics (FK) on the given pose. Similar to local pose, the global rotation of each joint is also represented in 6D and the positions are represented in 3D coordinate system, where the root join is positioned at the center of coordinate system facing forward. Combined together, the input to the first encoder **302** is a tensor of N×(6+6+3) for a single input pose. The first encoder is based on a multi-layer perceptron (MLP), acting on each joint's representation independently and in parallel. It aims at computing the embedding representation in a d dimensional embedding space.

**[0031]** The second encoder **312** computes an embedding representation of the target poses that is then consumable by the transformer encoder **314**. The second encoder **312** receives as input the 6-DoF representation (position and rotation) of an arbitrary number of target joints **310**. The number of target joints could be between 1 and the number of end effector joints (NEE). Similar to the first encoder, here the rotations are represented in 6D and positions are in 3D. Thus each target 6-DoF is represented as a 9D input. The second encoder is based on a multi-layer perceptron (MLP), acting on each joint's representation independently and in parallel. It aims at computing the embedding representation in a d dimensional embedding space. Note that, the model is trained only once and can be used for any subset of target joints.

**[0032]** The transformer encoder **314** learn the relation between different target poses. Unlike the initial pose representation in which each joint is represented relative to its parent in the body kinematic tree, and thus are strongly related to each other, target joints are represented globally and independently. Thus, it is of crucial importance to learn how they correlate and what potential poses they may represent. To this end, explicitly learn the relation between different target signals. For this purpose, use a transformer encoder **314** which utilizes a self-attention mechanism. The benefit of such design choice is two-fold: (1) to effectively transform each joint's embedding such that the resulting feature vector contains information about all other joints in relation to it, and (2) to effectively handle arbitrary number of input target joints by design. The resulting features are more expressive representation of target joints in an embedding space of similar dimension.

**[0033]** The transformer decoder **304** learns the cross-attention between the target joints' representations and that of the initial pose. Once expressive representation of the target joints are achieved, it is possible to learn how to modify the initial pose such that the resulting pose matches the target. This requires learning to attend to different parts of the initial pose to be able to change the pose of different body parts. This is achieved by explicitly learning the cross-attention between the target joints' representations and that of the initial pose. A transformer decoder **304** layer (which inherently utilizes cross-attention), learn how to translate the initial pose to a novel pose such that (1) if the target joint indicates changing a body part of the initial pose, it is reflected in the resulting representation, and (2) if it does not indicate a change in the initial pose, the initial pose representation remains unchanged.

**[0034]** In an example, modifying the first embedding vector using the second embedding vector comprises concatenating the first embedding vector and the second embedding vector to form a concatenated embedding vector and decoding the concatenated embedding vector using a decoder neural network.

**[0035]** The decoder **306** is used to decode the transformed embedding vector into the updated pose. Once the transformer decoder **304** computes the features, they serve as the input to a multi-layer perceptron (MLP) **306** that transforms the d dimensional feature representation of each joint to a 6D rotation representation, constructing a novel pose.

**[0036]** The architecture of FIG. **3** is modified in some examples as now explained.

**[0037]** In some cases shape parameters are used in addition to poses of the joints of the kinematic tree. The shape parameters are shape parameters of body parts of the kinematic tree. The shape parameters are concatenated to the inputs to the encoders and the architecture is the same except that the sizes of the embedding vectors may be larger since shape also has to be taken into account. Using shape parameters is particularly beneficial for human body pose estimation where body shape influences poses which are plausible.

**[0038]** The first embedding vector is in a multi-dimensional space and the second embedding vector is in the same multi-dimensional space. This is achieved either by end-to-end training of the neural network or by using the same encoder to produce the first and second embedding vectors.

**[0039]** FIG. **4** is a flow diagram of a method of updating a pose using a pose manipulator such as that of FIG. **1** or **3**. An initial pose and optional shape parameter values of an articulated body is received **400**. Target joint poses are received **402**. Optionally the target joint poses are received from a wearable inertial measurement unit IMU signal. In some examples the target joint poses are received **402** from a graphical user interface where a user drags and drops an icon or uses another user interface tool to specify the target joint poses.

**[0040]** The target joint poses and the initial poses are input **404** to the model (the inverse kinematics neural network of FIG. **3**) which outputs **406** an update pose of the articulated body. The process optionally repeats from operation **400**. The updated pose is provided **408** to a downstream process such as a film or video game **410**, a telepresence application **412**, a full body pose from wearable head mounted display HMD application.

**[0041]** FIG. **5** is a flow diagram of a method of creating training data to train a pose manipulator such as that of FIG. **1**. A library **500** of poses of the type of articulated entity is available. In the case of human body pose this is a library of human body poses. In the case of a motor vehicle this is a library of poses of motor vehicles. To prepare a sample to train the inverse kinematics neural network, two poses are sampled from the library **500**, such as by random sampling or other sampling methods. This gives sample pose x **502** and sample pose z **504**. Then normalize each pose such that (1) the root appears in the center of the coordinate system, and (2) the root of the pose, (i.e., the pelvis in the case of a human body), is facing forward. With that, the two poses are globally originated similarly. Since the pose constructed as a kinematic tree, once can replace part of the kinematic tree of a pose that leads to a leaf node with that of another pose. For instance, if one defines the left arm as left shoulder, elbow; and wrist, it is possible to replace these joints' rotations with same joints rotations of another pose, resulting in a new pose where everything except for the left arm is identical to the original pose. The same process is possible for any part of the kinematic tree. As illustrated in FIG. **5** randomly sample a pose from pose library **500** to serve as x. Randomly sample another pose z that is different from x. Randomly select **508** one or more body parts leading to a subset of leaf joints. Create a new pose y by copying **506** x and replacing the sampled body part from z. Compute **510** forward kinematics to come up with the 6-DoF of the modified leaf joints, resulting in the target T.

**[0042]** This results in the triplet [x, y, T]. Store **512** the triplet and check if there are enough triplets in the store. If the store is not yet full repeat the process. Otherwise end **516**.

**[0043]** FIG. **6** is a flow diagram of a method of training a pose manipulator. The process comprises accessing training data **600** where the training data comprises triplets as explained with reference to FIG. **5**. A training example is accessed **602** and end to end supervised training **604** is carried out on the inverse kinematics neural network. The training uses backpropagation with a loss function **606** as explained below. If convergence is reached (where the neural network weights change only a very small amount in a training iteration) the training ends and the model is stored **610**. If convergence is not reached at check **608** the process repeats from operation **602** for another training example.

**[0044]** To optimize the parameters of the model, the following loss function may be used, encouraging the model to predict a pose that is close to y, given x and T.

$$L = L_{local} + L_{world} + L_{EE}$$

**[0045]** The first term, $L_{local}$, is the reconstruction loss between the direct prediction of the model and y. The second term, $L_{world}$, is the reconstruction loss between the prediction of the model and y in the world coordinates after computing forward kinematics on both poses. Finally, LEE, is the reconstruction loss between the 6-DoFs of the predicted end effectors and T. All reconstruction losses are LI loss.

**[0046]** The inverse kinematics neural network of FIG. **3** was tested empirically. The Euclidean distance between the generated poses end effector 6-DoFs and that of the target signal were computed. For the left wrist the error was 3.2 cm, for the right wrist 3.17 cm for the left ankle 3.15 cm, for the right ankle 3.16 cm.

**[0047]** FIG. **7** illustrates various components of an exemplary computing-based device **700** which are implemented as any form of a computing and/or electronic device (such as a smart phone, HMD, self-driving vehicle, laptop computer, data center compute node), and in which examples of a pose manipulator are implemented in some examples.

**[0048]** Computing-based device **700** comprises one or more processors **714** which are microprocessors, controllers or any other suitable type of processors for processing computer executable instructions to control the operation of the device in order to manipulate the pose of an articulated entity having a kinematic tree. In some examples, for example where a system on a chip architecture is used, the processors **714** include one or more fixed function blocks (also referred to as accelerators) which implement a part of the method of any of FIGS. **4** to **6** in hardware (rather than software or firmware). A pose manipulator **702** is deployed in the computing-based device **700** and comprises an inverse kinematics neural network **704**. Platform software comprising an operating system **706** or any other suitable platform software is provided at the computing-based device to enable application software **710** to be executed on the device. Data store **708** holds poses, kinematic trees, shape parameters or other data.

**[0049]** The computer executable instructions are provided using any computer-readable media that is accessible by computing based device **700**. Computer-readable media includes, for example, computer storage media such as memory **712** and communications media. Computer storage media, such as memory **712**, includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or the like. Computer storage media includes, but is not limited to, random access memory (RAM), read only memory (ROM), erasable programmable read only memory (EPROM), electronic erasable programmable read only memory (EEPROM), flash memory or other memory technology, compact disc read only memory (CD-ROM), digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transmission medium that is used to store information for access by a computing device. In contrast, communication media embody computer readable instructions, data structures, program modules, or the like in a modulated data signal, such as a carrier wave, or other transport mechanism. As defined herein, computer storage media does not include communication media. Therefore, a computer storage medium should not be interpreted to be a propagating signal per se. Although the computer storage media (memory **712**) is shown within the computing-based device **700** it will be appreciated that the storage is, in some examples, distributed or located remotely and accessed via a network or other communication link (e.g. using communication interface **716**).

**[0050]** The computing-based device optionally has a display device **720**. The display information may provide a graphical user interface. The computing-based device optionally has a capture device **718** such as for capturing inertial measurement unit IMU signals or other data.

**[0051]** Alternatively or in addition to the other examples described herein, examples include any combination of the following clauses:

**[0052]** Clause A. A computer-implemented method of updating a pose of a plurality of joints of a kinematic tree of an articulated object, the method comprising:

- **[0053]** receiving, for each of the joints in the kinematic tree, an initial pose;
- **[0054]** computing a single first embedding vector by encoding the initial poses in an embedding space;
- **[0055]** for each of some but not all of the joints in the kinematic tree, receiving a target pose;
- **[0056]** computing a single second embedding vector representing the target poses in the embedding space;

**[0057]** modifying the first embedding vector using the second embedding vector to form a third embedding vector;

**[0058]** decoding the third embedding vector to produce the updated pose of the articulated object.

**[0059]** Clause B. The method of clause A wherein the updated pose of the articulated object is an approximation of inverse kinematics applied to the kinematic tree using the target poses.

**[0060]** Clause C. The method of clause A or clause B wherein computing the first embedding vector comprises:

**[0061]** concatenating the initial poses; and

**[0062]** using a first encoder neural network to encode the concatenated initial poses producing the first embedding vector.

**[0063]** Clause D. The method of any preceding clause wherein computing the second embedding vector comprises concatenating the target poses and using a second encoder neural network to encode the concatenated target poses.

**[0064]** Clause E. The method of clause D wherein the second encoder neural network is different from the first encoder neural network.

**[0065]** Clause F. The method of clause C wherein the first encoder neural network is a multi-layer perceptron.

**[0066]** Clause G. The method of clause D wherein the second encoder neural network is a multi-layer perceptron.

**[0067]** Clause H. The method of any preceding clause wherein the first embedding vector is in a multi-dimensional space and the second embedding vector is in the same multi-dimensional space.

**[0068]** Clause I. The method of any preceding clause wherein modifying the first embedding vector using the second embedding vector comprises concatenating the first embedding vector and the second embedding vector to form a concatenated embedding vector and decoding the concatenated embedding vector using a decoder neural network.

**[0069]** Clause J. The method of clause I wherein the decoder neural network comprises a transformer neural network.

**[0070]** Clause K. The method of clause I wherein the decoder neural network comprises a transformer neural network and a decoder neural network.

**[0071]** Clause L. The method of any preceding clause comprising applying self-attention to the first embedding vector using a transformer neural network prior to using the first embedding vector to modify the second embedding vector.

**[0072]** Clause M. The method of any preceding clause comprising applying cross attention to the first embedding vector and the second embedding vector using a transformer neural network.

**[0073]** Clause N. The method of any preceding clause comprising receiving values of shape parameters of the articulated object and computing the first embedding vector by encoding both the initial poses and the shape parameter values such that the updated pose of the articulated object takes into account the shape parameter values.

**[0074]** Clause O. The method of any preceding clause comprising using neural networks to compute the first embedding vector, the second embedding vector and the third embedding vector and to decode the third embedding vector, where the neural networks are trained end-to-end using supervised learning.

**[0075]** Clause P. The method of clause O comprising carrying out the supervised learning using training examples, each training example comprising a triplet X, Y, T where X is first pose of the kinematic tree of the articulated object, Y is a second pose of the kinematic tree of the articulated object created by copying X and replacing one or more leaf joints of X using Z, where Z is another pose of the kinematic tree of the articulated object, and where T is the pose of the replaced leaf joints computed using forward kinematics.

**[0076]** Clause Q. The method of any preceding clause comprising presenting the kinematic tree of the initial pose in a graphical user interface and receiving user input moving a leaf joint to specify a target pose.

**[0077]** Clause R. The method of any preceding clause comprising using the updated pose of the articulated object for any of: enabling a non-artist to update pose of an avatar, task-specific pose augmentation, augmenting an upper body motion by modifying lower body pose or vice versa, full body pose estimation given signals from a wearable device, full body pose estimation from inertial measurement unit sensors worn only on wrists and ankles.

**[0078]** Clause S. A computer storage medium having computer-executable instructions that, when executed by a computing system, direct the computing system to perform operations comprising:

**[0079]** receiving, for each of the joints in the kinematic tree, an initial pose;

**[0080]** computing a single first embedding vector by encoding the initial poses in an embedding space;

**[0081]** for each of some but not all of the joints in the kinematic tree, receiving a target pose;

**[0082]** computing a single second embedding vector representing the target poses in the embedding space;

**[0083]** modifying the first embedding vector using the second embedding vector to form a third embedding vector, by using a transformer neural network;

**[0084]** decoding the third embedding vector to produce the updated pose of the articulated object.

**[0085]** Clause T. An apparatus comprising:

**[0086]** a processor;

**[0087]** a memory storing instructions that, when executed by the processor, perform a method, comprising:

**[0088]** receiving, for each of the joints in the kinematic tree, an initial pose;

**[0089]** computing a single first embedding vector by encoding the initial poses;

**[0090]** for each of some but not all of the joints in the kinematic tree, receiving a target pose;

**[0091]** computing a single second embedding vector representing the target poses;

**[0092]** modifying the first embedding vector using the second embedding vector to form a third embedding vector;

**[0093]** decoding the third embedding vector to produce the updated pose of the articulated object.

**[0094]** The term 'computer' or 'computing-based device' is used herein to refer to any device with processing capability such that it executes instructions. Those skilled in the art will realize that such processing capabilities are incorporated into many different devices and therefore the terms 'computer' and 'computing-based device' each include personal computers (PCs), servers, mobile telephones (including smart phones), tablet computers, set-top boxes, media players, games consoles, personal digital assistants, wearable computers, and many other devices.

**[0095]** The methods described herein are performed, in some examples, by software in machine readable form on a tangible storage medium e.g. in the form of a computer program comprising computer program code means adapted to perform all the operations of one or more of the methods described herein when the program is run on a computer and where the computer program may be embodied on a computer readable medium. The software is suitable for execution on a parallel processor or a serial processor such that the method operations may be carried out in any suitable order, or simultaneously.

**[0096]** Those skilled in the art will realize that storage devices utilized to store program instructions are optionally distributed across a network. For example, a remote computer is able to store an example of the process described as software. A local or terminal computer is able to access the remote computer and download a part or all of the software to run the program. Alternatively, the local computer may download pieces of the software as needed, or execute some software instructions at the local terminal and some at the remote computer (or computer network). Those skilled in the art will also realize that by utilizing conventional techniques known to those skilled in the art that all, or a portion of the software instructions may be carried out by a dedicated circuit, such as a digital signal processor (DSP), programmable logic array, or the like.

**[0097]** Any range or device value given herein may be extended or altered without losing the effect sought, as will be apparent to the skilled person.

**[0098]** Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

**[0099]** It will be understood that the benefits and advantages described above may relate to one embodiment or may relate to several embodiments. The embodiments are not limited to those that solve any or all of the stated problems or those that have any or all of the stated benefits and advantages. It will further be understood that reference to 'an' item refers to one or more of those items.

**[0100]** The operations of the methods described herein may be carried out in any suitable order, or simultaneously where appropriate. Additionally, individual blocks may be deleted from any of the methods without departing from the scope of the subject matter described herein. Aspects of any of the examples described above may be combined with aspects of any of the other examples described to form further examples without losing the effect sought.

**[0101]** The term 'comprising' is used herein to mean including the method blocks or elements identified, but that such blocks or elements do not comprise an exclusive list and a method or apparatus may contain additional blocks or elements.

**[0102]** It will be understood that the above description is given by way of example only and that various modifications may be made by those skilled in the art. The above specification, examples and data provide a complete description of the structure and use of exemplary embodiments. Although various embodiments have been described above with a certain degree of particularity, or with reference to one or more individual embodiments, those skilled in the art could make numerous alterations to the disclosed embodiments without departing from the scope of this specification.

What is claimed is:

**1**. A computer-implemented method of updating a pose of a plurality of joints of a kinematic tree of an articulated object, the method comprising:

receiving, for each of the joints in the kinematic tree, an initial pose;

computing a single first embedding vector by encoding the initial poses in an embedding space;

for each of some but not all of the joints in the kinematic tree, receiving a target pose;

computing a single second embedding vector representing the target poses in the embedding space;

modifying the first embedding vector using the second embedding vector to form a third embedding vector;

decoding the third embedding vector to produce the updated pose of the articulated object.

**2**. The method of claim **1** wherein the updated pose of the articulated object is an approximation of inverse kinematics applied to the kinematic tree using the target poses.

**3**. The method of claim **1** wherein computing the first embedding vector comprises:

concatenating the initial poses; and

using a first encoder neural network to encode the concatenated initial poses producing the first embedding vector.

**4**. The method of claim **3** wherein the first encoder neural network is a multi-layer perceptron.

**5**. The method of claim **1** wherein computing the second embedding vector comprises concatenating the target poses and using a second encoder neural network to encode the concatenated target poses.

**6**. The method of claim **5** wherein the second encoder neural network is different from the first encoder neural network.

**7**. The method of claim **5** wherein the second encoder neural network is a multi-layer perceptron.

**8**. The method of claim **1** wherein the first embedding vector is in a multi-dimensional space and the second embedding vector is in the same multi-dimensional space.

**9**. The method of claim **1** wherein modifying the first embedding vector using the second embedding vector comprises concatenating the first embedding vector and the second embedding vector to form a concatenated embedding vector and decoding the concatenated embedding vector using a decoder neural network.

**10**. The method of claim **9** wherein the decoder neural network comprises a transformer neural network.

**11**. The method of claim **9** wherein the decoder neural network comprises a transformer neural network and a decoder neural network.

**12**. The method of claim **1** comprising applying self-attention to the first embedding vector using a transformer neural network prior to using the first embedding vector to modify the second embedding vector.

**13**. The method of claim **1** comprising applying cross attention to the first embedding vector and the second embedding vector using a transformer neural network.

**14**. The method of claim **1** comprising receiving values of shape parameters of the articulated object and computing the first embedding vector by encoding both the initial poses and the shape parameter values such that the updated pose of the articulated object takes into account the shape parameter values.

**15**. The method of claim **1** comprising using neural networks to compute the first embedding vector, the second embedding vector and the third embedding vector and to decode the third embedding vector, where the neural networks are trained end-to-end using supervised learning.

**16**. The method of claim **15** comprising carrying out the supervised learning using training examples, each training example comprising a triplet X, Y, T where X is first pose of the kinematic tree of the articulated object, Y is a second pose of the kinematic tree of the articulated object created by copying X and replacing one or more leaf joints of X using Z, where Z is another pose of the kinematic tree of the articulated object, and where T is the pose of the replaced leaf joints computed using forward kinematics.

**17**. The method of claim **1** comprising presenting the kinematic tree of the initial pose in a graphical user interface and receiving user input moving a leaf joint to specify a target pose.

**18**. The method of claim **1** comprising using the updated pose of the articulated object for any of: enabling a non-artist to update pose of an avatar, task-specific pose augmentation, augmenting an upper body motion by modifying lower body pose or vice versa, full body pose estimation given signals from a wearable device, full body pose estimation from inertial measurement unit sensors worn only on wrists and ankles.

**19**. A computer storage medium having computer-executable instructions that, when executed by a computing system, direct the computing system to perform operations comprising:

receiving, for each of the joints in the kinematic tree, an initial pose;

computing a single first embedding vector by encoding the initial poses in an embedding space;

for each of some but not all of the joints in the kinematic tree, receiving a target pose;

computing a single second embedding vector representing the target poses in the embedding space;

modifying the first embedding vector using the second embedding vector to form a third embedding vector, by using a transformer neural network;

decoding the third embedding vector to produce the updated pose of the articulated object.

**20**. An apparatus comprising:

a processor;

a memory storing instructions that, when executed by the processor, perform a method, comprising:

receiving, for each of the joints in the kinematic tree, an initial pose;

computing a single first embedding vector by encoding the initial poses;

for each of some but not all of the joints in the kinematic tree, receiving a target pose;

computing a single second embedding vector representing the target poses;

modifying the first embedding vector using the second embedding vector to form a third embedding vector;

decoding the third embedding vector to produce the updated pose of the articulated object.

* * * * *