US 20220193898A1

(19) **United States**
(12) **Patent Application Publication**
Aghasadeghi et al.

(10) **Pub. No.: US 2022/0193898 A1**
(43) **Pub. Date: Jun. 23, 2022**

(54) CONSTRAINED MANIPULATION OF OBJECTS

(71) Applicant: **Boston Dynamics, Inc.**, Waltham, MA (US)

(72) Inventors: **Navid Aghasadeghi**, Boston, MA (US); **Alfred Anthony Rizzi**, Waltham, MA (US); **Gina Fay**, Lexington, MA (US); **Robert Eugene Paolini**, Cambridge, MA (US)

(73) Assignee: **Boston Dynamics, Inc.**, Waltham, MA (US)

(21) Appl. No.: **17/644,642**

(22) Filed: **Dec. 16, 2021**

Related U.S. Application Data

(60) Provisional application No. 63/128,573, filed on Dec. 21, 2020.

Publication Classification

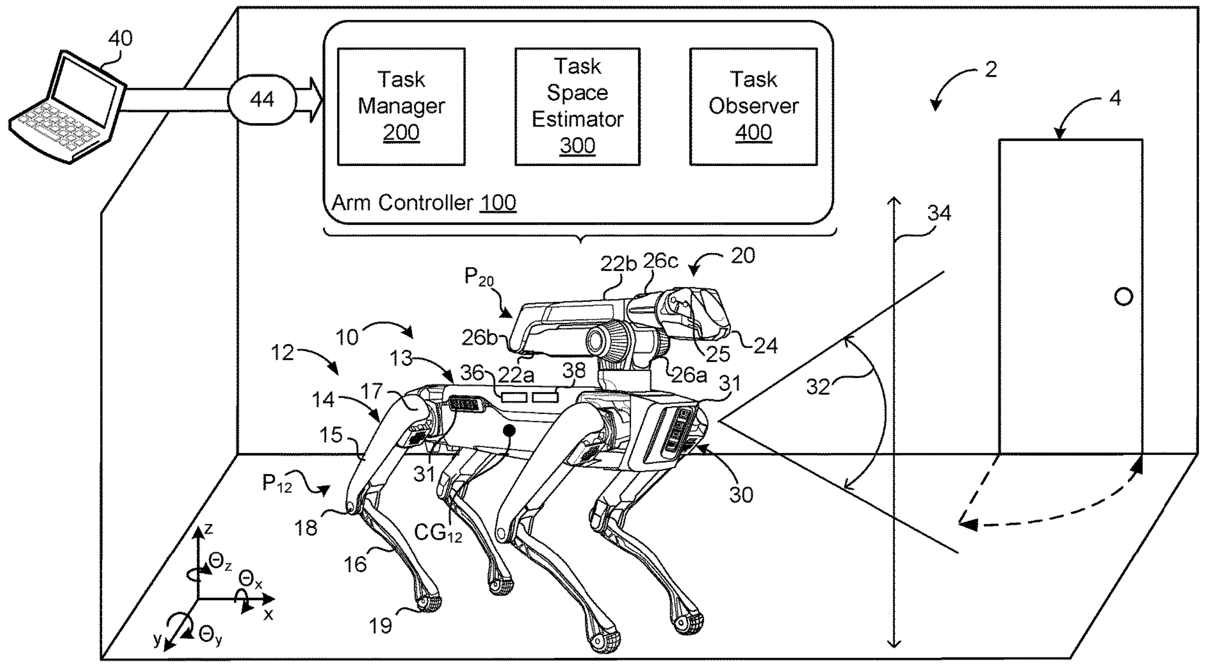
(51) **Int. Cl.**

B25J 9/16	(2006.01)
B25J 13/08	(2006.01)
G05B 19/4155	(2006.01)

(52) **U.S. Cl.**
CPC ***B25J 9/1628*** (2013.01); ***B25J 13/088*** (2013.01); *B62D 57/02* (2013.01); *G05B 2219/40269* (2013.01); ***G05B 19/4155*** (2013.01)

(57) **ABSTRACT**

A computer-implemented method executed by data processing hardware of a robot causes the data processing hardware to perform operations. The robot includes an articulated arm having an end effector engaged with a constrained object. The operations include receiving a measured task parameter set for the end effector. The measured task parameter set includes position parameters defining a position of the end effector. The operations further include determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace. The operations also include assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis. The operations include instructing the articulated arm to move the end effector along the at least one axis of freedom.



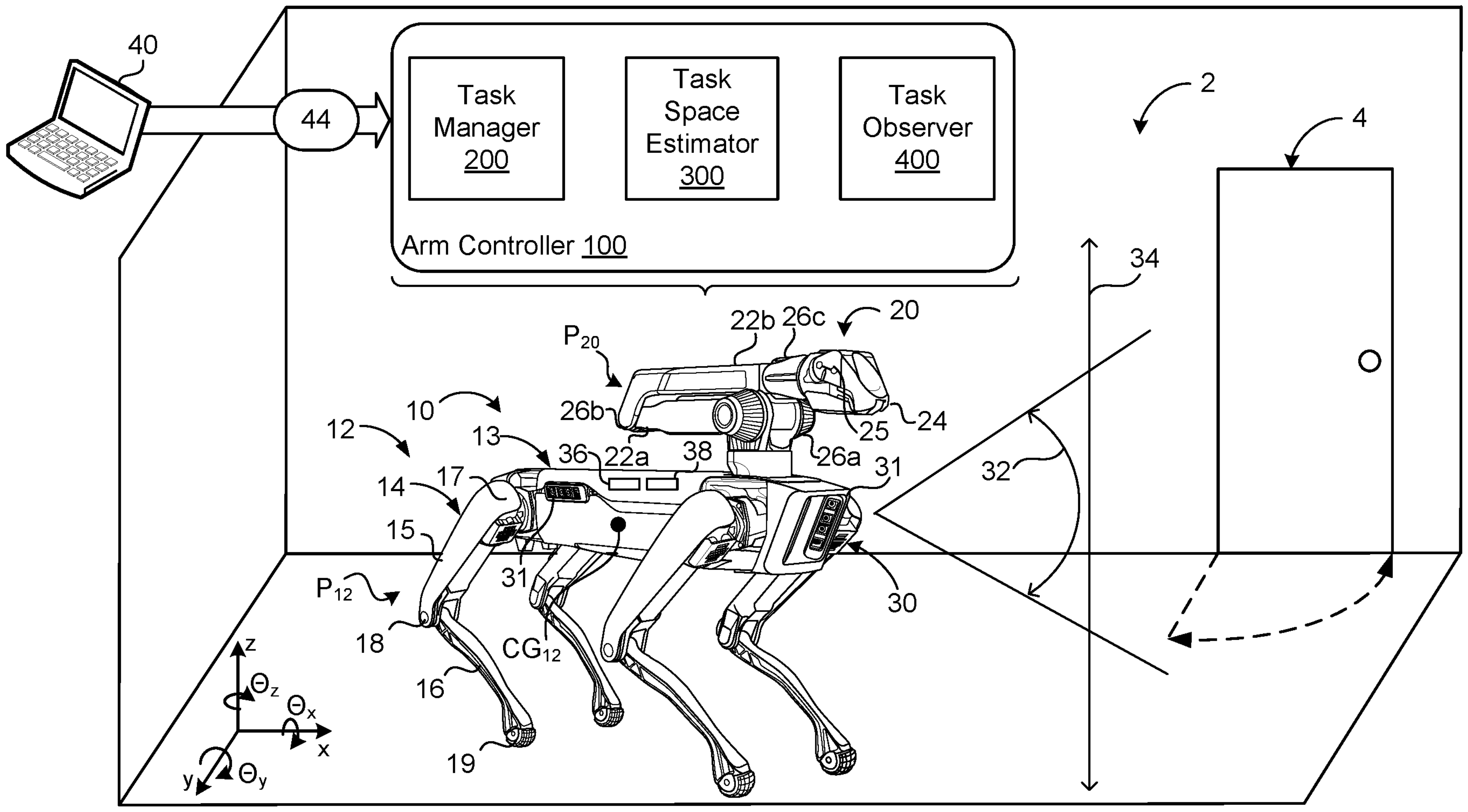

40
44
Task Manager 200
Task Space Estimator 300
Task Observer 400
Arm Controller 100
2
4
34
20
22b
26c
P_{20}
26b
10
12
13
36
22a
38
26a
25
24
31
32
17
14
15
P_{12}
31
30
18
CG_{12}
16
19
z
Θ_z
Θ_x
x
y
Θ_y

FIG. 1A

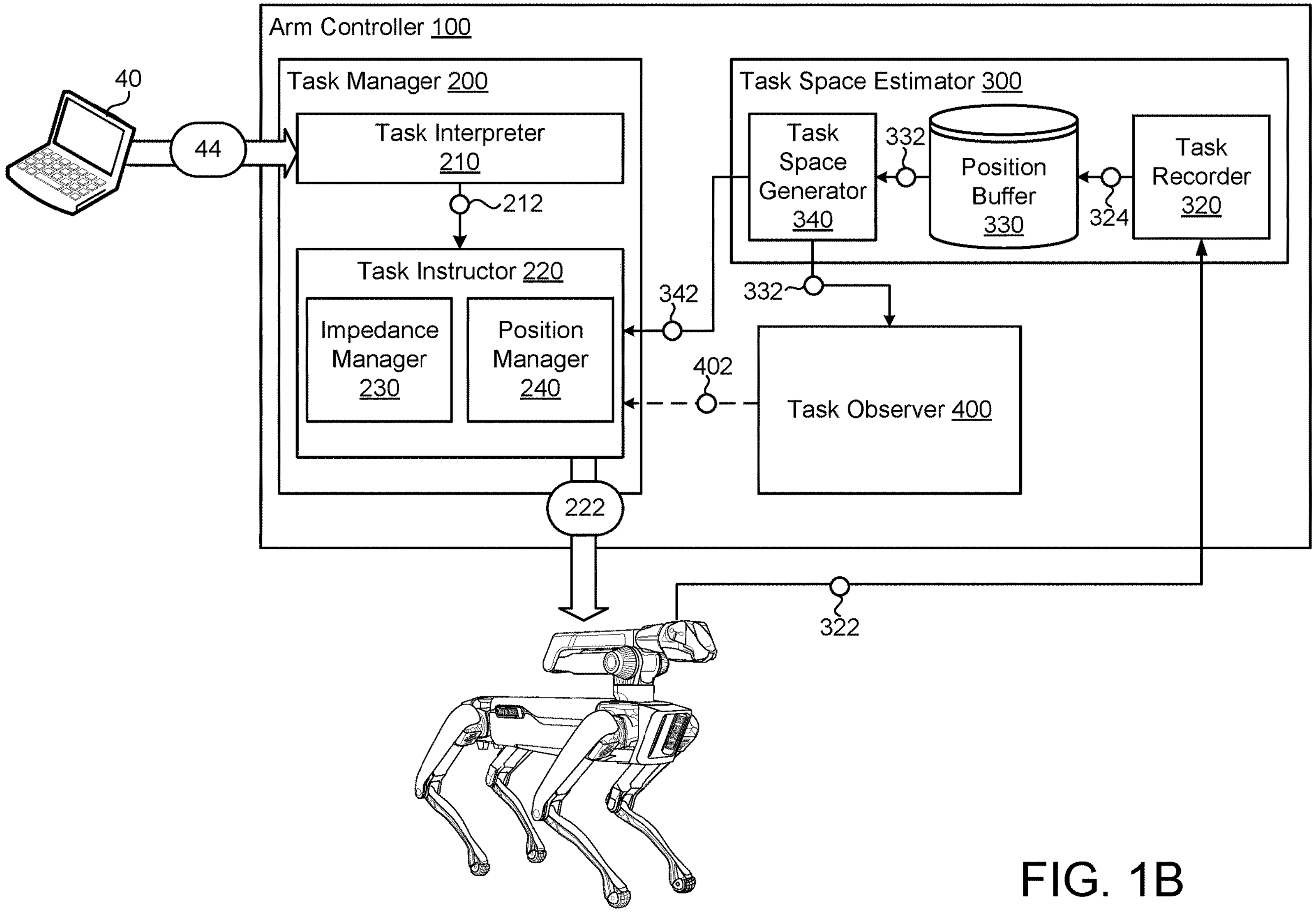

Arm Controller 100
Task Manager 200
Task Interpreter 210
212
Task Instructor 220
Impedance Manager 230
Position Manager 240
222
40
44
Task Space Estimator 300
Task Space Generator 340
332
Position Buffer 330
324
Task Recorder 320
332
342
402
Task Observer 400
322

FIG. 1B

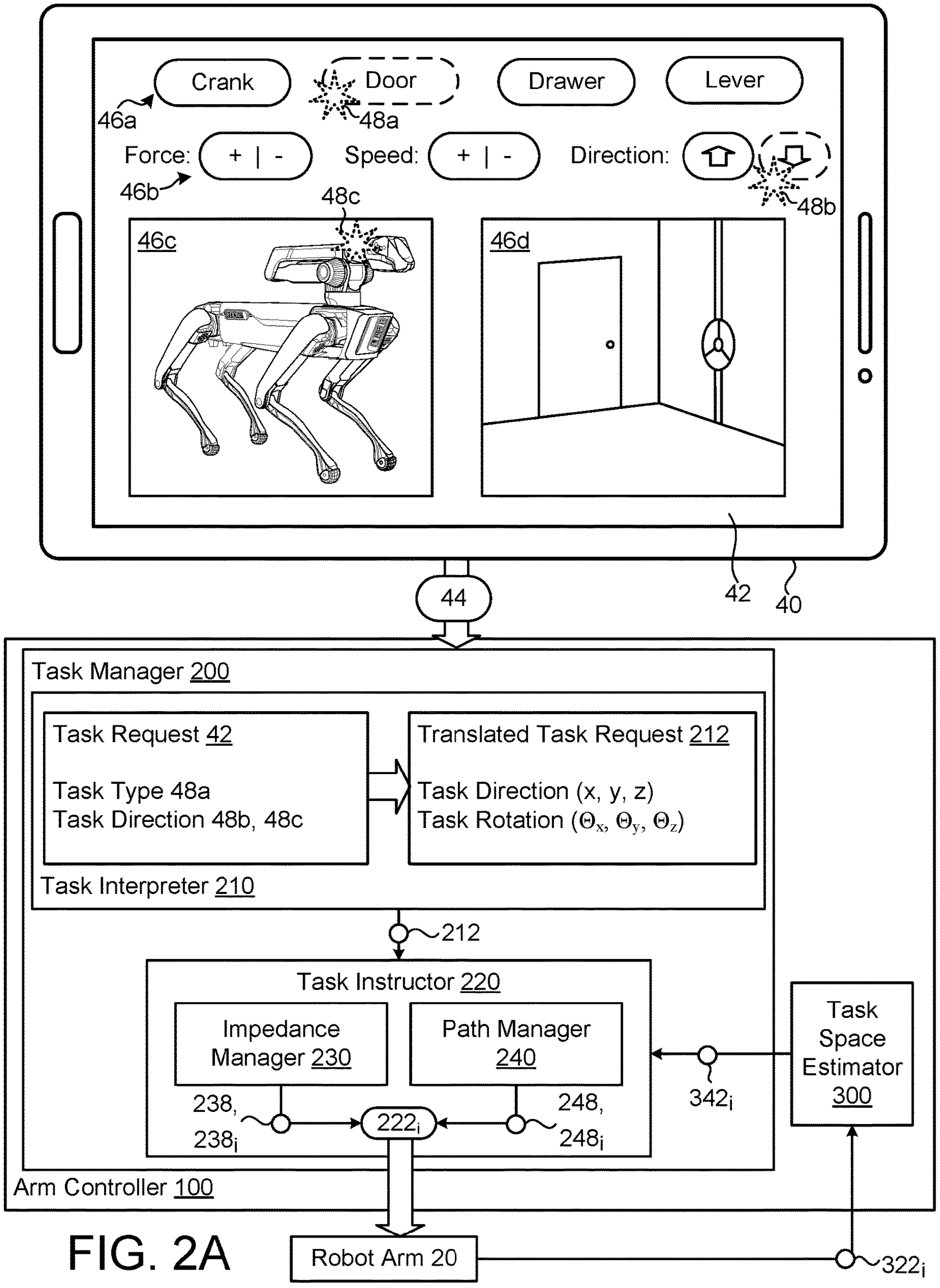

Crank
Door
Drawer
Lever
46a
48a
Force: + | -
Speed: + | -
Direction:
46b
48c
48b
46c
46d
44
42
40
Task Manager 200
Task Request 42
Task Type 48a
Task Direction 48b, 48c
Translated Task Request 212
Task Direction (x, y, z)
Task Rotation (Θ_x, Θ_y, Θ_z)
Task Interpreter 210
212
Task Instructor 220
Impedance Manager 230
Path Manager 240
238, 238_i
248, 248_i
222_i
342_i
Task Space Estimator 300
Arm Controller 100
Robot Arm 20
322_i

FIG. 2A

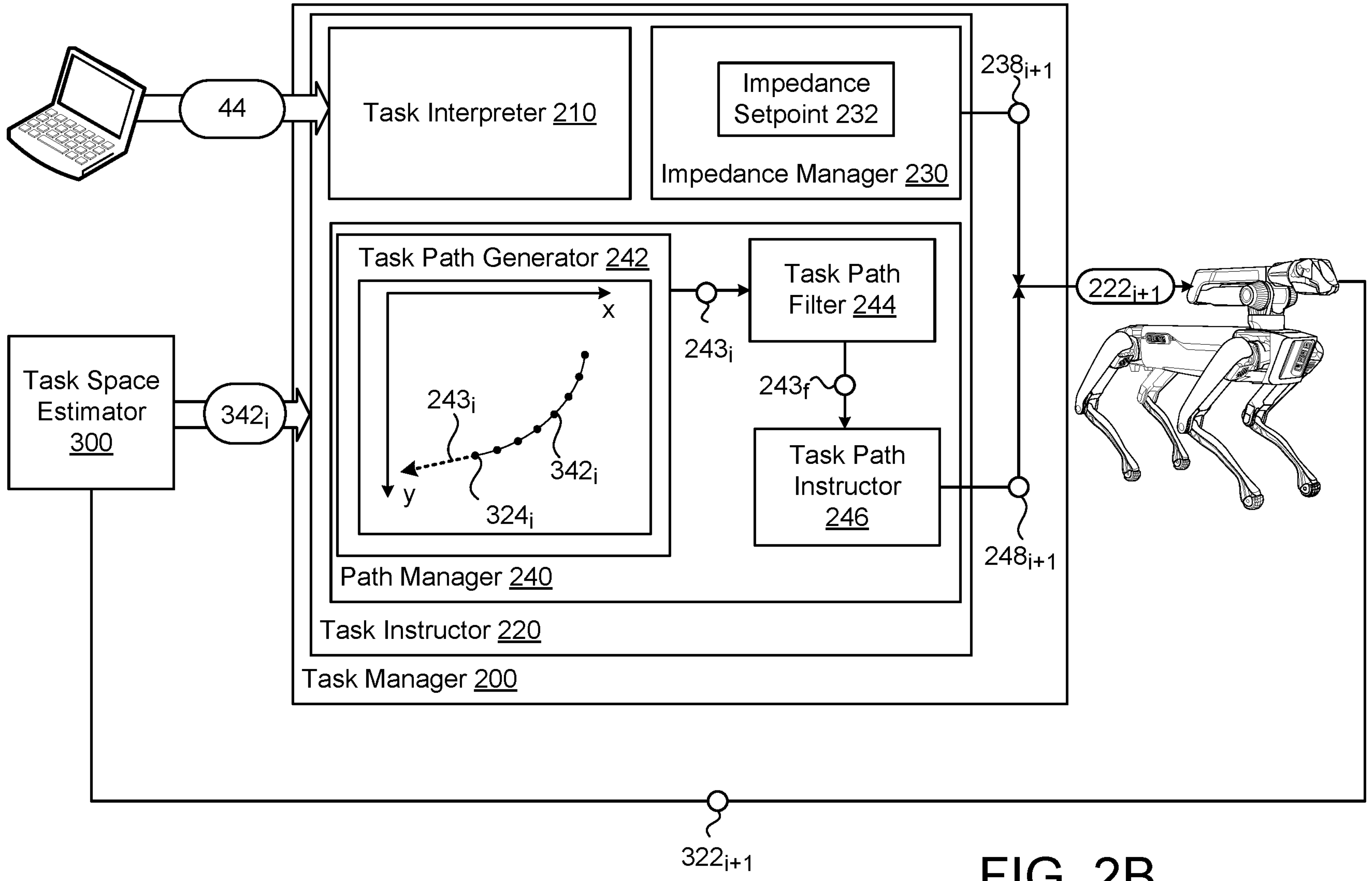

44
Task Interpreter 210
Impedance Setpoint 232
Impedance Manager 230
238_{i+1}
Task Path Generator 242
x
y
243_i
342_i
324_i
243_i
Task Path Filter 244
243_f
Task Path Instructor 246
248_{i+1}
222_{i+1}
Path Manager 240
Task Instructor 220
Task Manager 200
Task Space Estimator 300
342_i
322_{i+1}

FIG. 2B

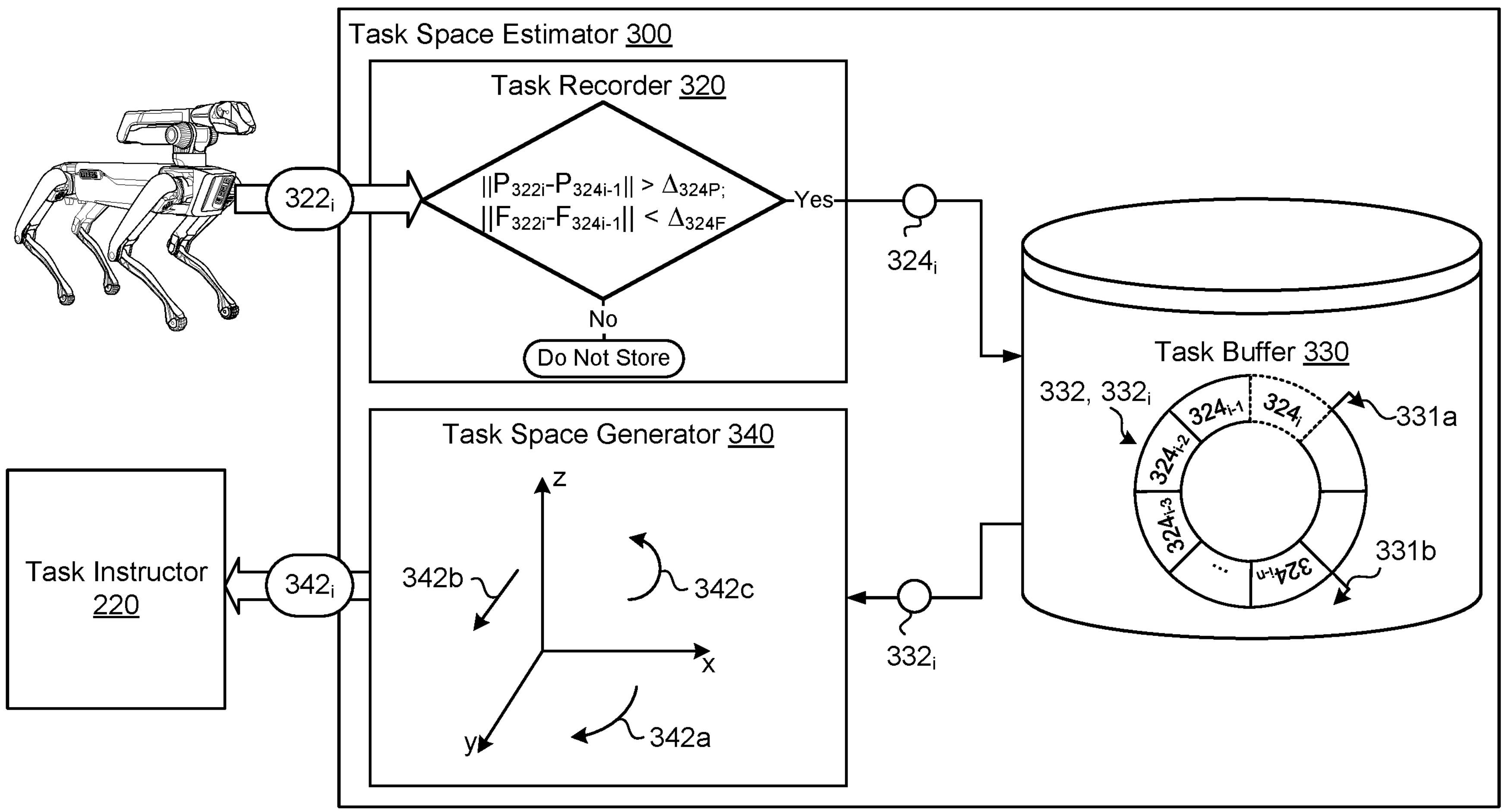

Task Space Estimator 300
Task Recorder 320
322_i
$\|P_{322i}-P_{324i-1}\| > \Delta_{324P};$
$\|F_{322i}-F_{324i-1}\| < \Delta_{324F}$
Yes
No
Do Not Store
324_i
Task Buffer 330
$332, 332_i$
324_{i-1}
324_i
324_{i-2}
324_{i-3}
...
324_{i-n}
331a
331b
Task Space Generator 340
z
x
y
342a
342b
342c
332_i
342_i
Task Instructor 220

FIG. 3

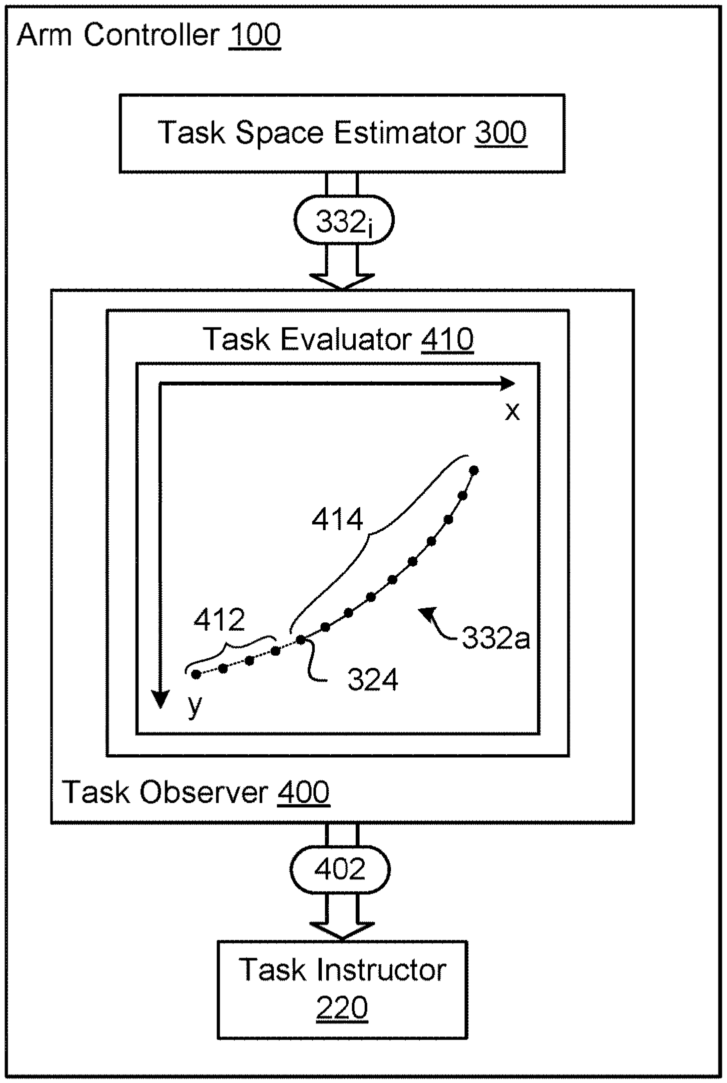

Arm Controller 100
Task Space Estimator 300
332_i
Task Evaluator 410
x
y
414
412
324
332a
Task Observer 400
402
Task Instructor 220

FIG. 4A

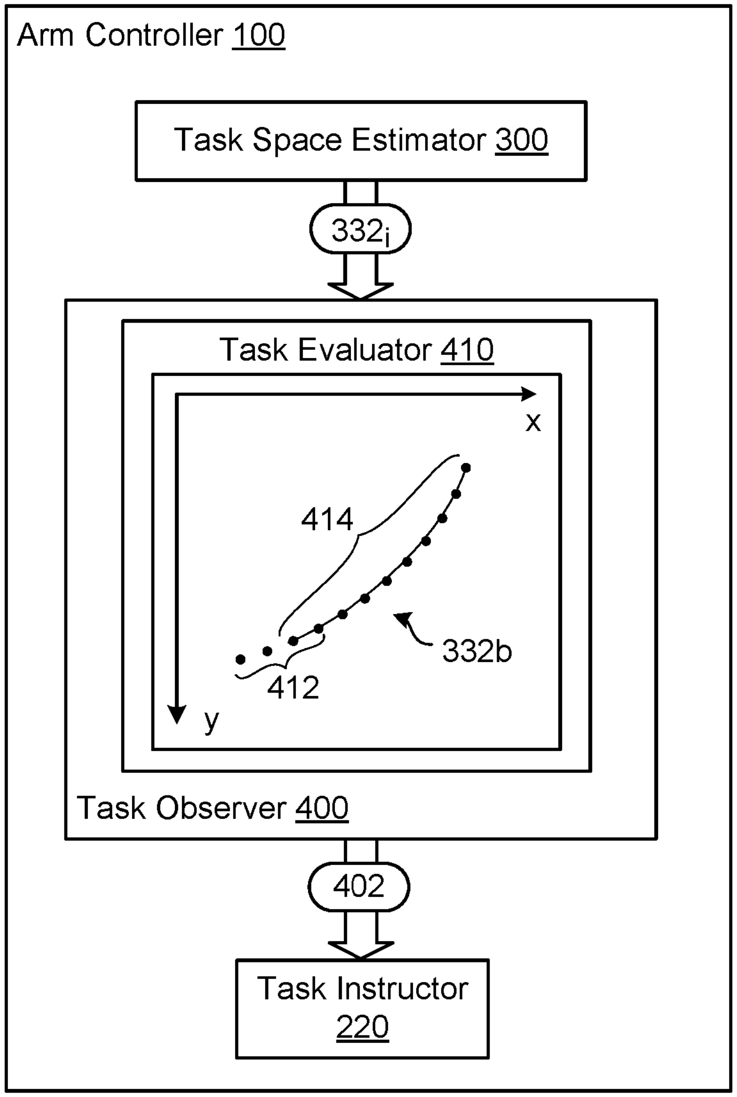

Arm Controller 100
Task Space Estimator 300
332_i
Task Evaluator 410
x
y
414
412
332b
Task Observer 400
402
Task Instructor 220

FIG. 4B

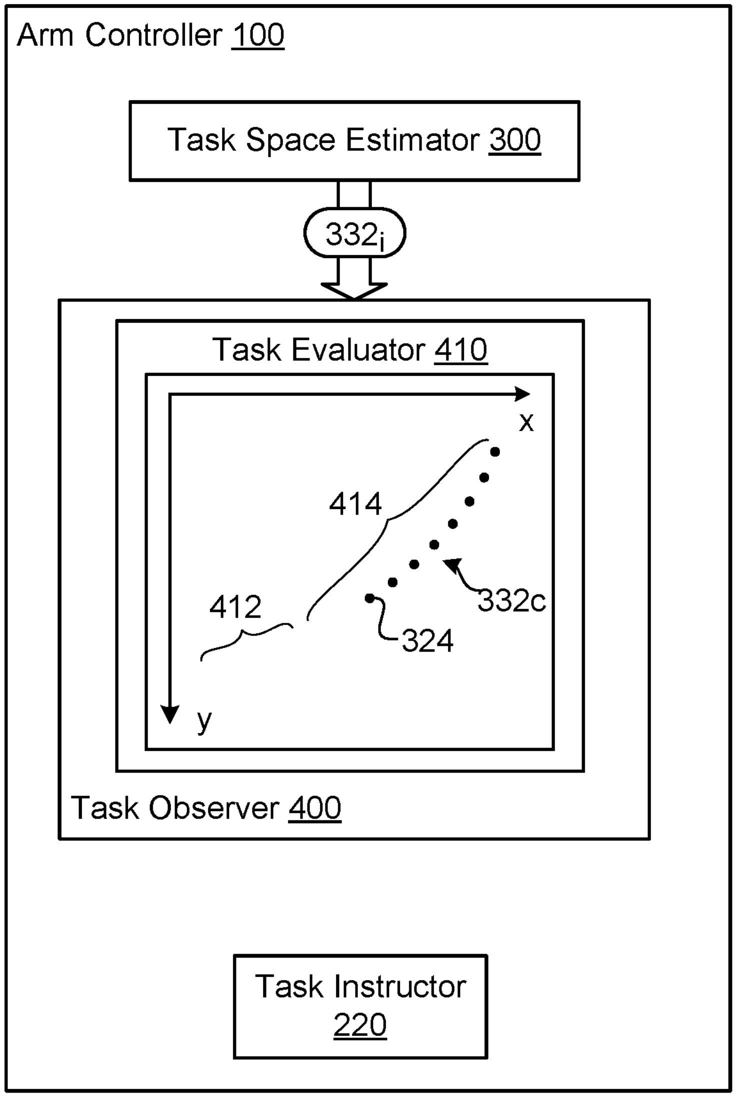

Arm Controller 100
Task Space Estimator 300
332_i
Task Evaluator 410
x
y
414
412
332c
324
Task Observer 400
Task Instructor 220

FIG. 4C

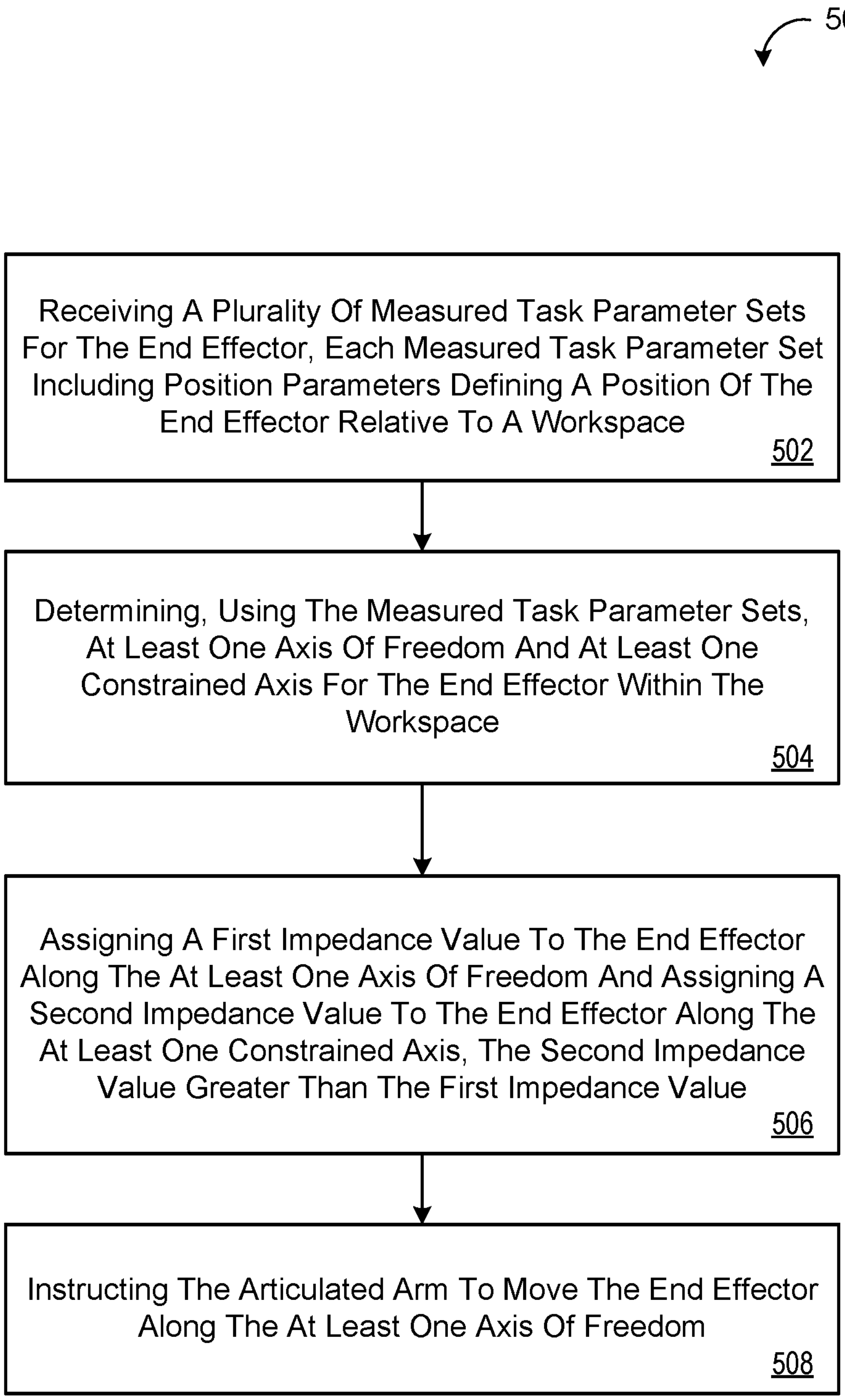

500
Receiving A Plurality Of Measured Task Parameter Sets For The End Effector, Each Measured Task Parameter Set Including Position Parameters Defining A Position Of The End Effector Relative To A Workspace
502
Determining, Using The Measured Task Parameter Sets, At Least One Axis Of Freedom And At Least One Constrained Axis For The End Effector Within The Workspace
504
Assigning A First Impedance Value To The End Effector Along The At Least One Axis Of Freedom And Assigning A Second Impedance Value To The End Effector Along The At Least One Constrained Axis, The Second Impedance Value Greater Than The First Impedance Value
506
Instructing The Articulated Arm To Move The End Effector Along The At Least One Axis Of Freedom
508

FIG. 5

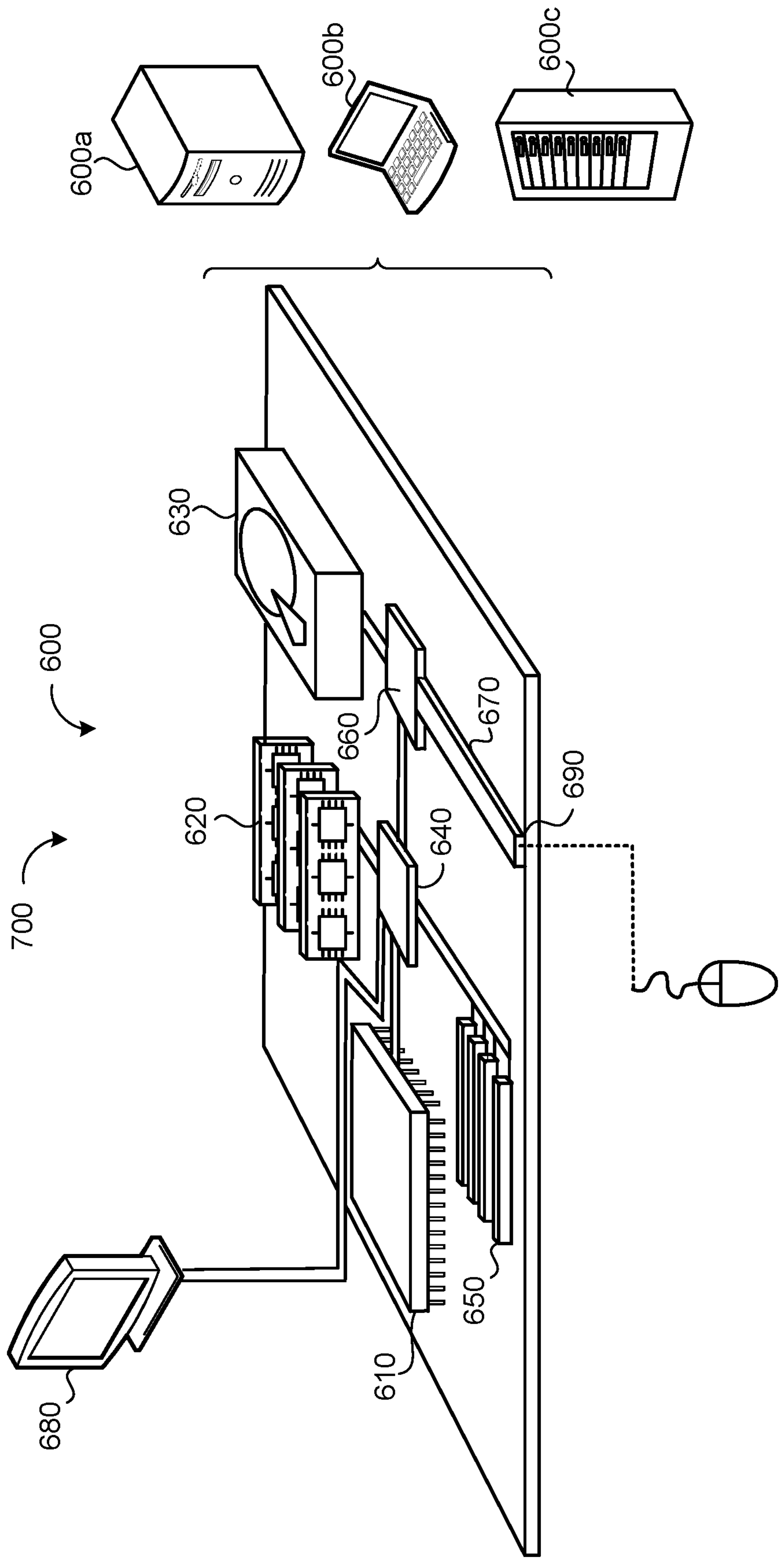


FIG. 6

CONSTRAINED MANIPULATION OF OBJECTS

CROSS REFERENCE TO RELATED APPLICATION

[0001] This U.S. patent application claims priority under 35 U.S.C. § 119(e) to U.S. Provisional Application 63/128,573, filed on Dec. 21, 2020. The disclosure of this prior application is considered part of the disclosure of this application and is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] This disclosure relates to constrained manipulation of objects using a robotic arm.

BACKGROUND

[0003] Robotic arms are increasingly being used in constrained or otherwise restricted environments to perform a variety of tasks or functions. These robotic arms often need to efficiently manipulate constrained objects, such as doors or switches, without requiring large computations. As robotic arms become more prevalent, there is a need for arm path planning that quickly determines and executes a path associated with a constrained object.

SUMMARY

[0004] One aspect of the disclosure provides a computer-implemented method. When executed by data processing hardware of a robot, the computer-implemented method causes the data processing hardware to perform operations. The robot includes an articulated arm having an end effector engaged with a constrained object. The operations include receiving a measured task parameter set for the end effector. The measured task parameter set includes position parameters defining a position of the end effector. The operations further include determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace. The operations also include assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis. Additionally, the operations include instructing the articulated arm to move the end effector along the at least one axis of freedom.

[0005] Aspects of the disclosure may include one or more of the following optional features. In some implementations, determining the at least one axis of freedom and the at least one constrained axis includes determining a task space model for the constrained object using the measured task parameter set. In some embodiments, the operations further include storing at least a portion of the measured task parameter set in a task buffer as task parameter records. In further embodiments, storing at least a portion of the measured task parameter set includes comparing at least one measured parameter from the measured parameter task set to a recorded parameter of one of the task parameter records of the task buffer and generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter satisfies a recording threshold. In even further embodiments, the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

[0006] In other further embodiments, the operations further include evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records. In other further embodiments, the operations further include evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

[0007] Another aspect of the disclosure provides a robot. The robot includes an articulated arm, data processing hardware in communication with the articulated arm, and memory hardware in communication with the data processing hardware. The articulated arm has an end effector for engaging a constrained object. The memory hardware stores instructions that when executed on the data processing hardware cause the data processing hardware to perform operations. The operations include receiving a measured task parameter set for the end effector. The measured task parameter set includes position parameters defining a position of the end effector. The operations further include determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace. The operations also include assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis. Additionally, the operations include instructing the articulated arm to move the end effector along the at least one axis of freedom.

[0008] Aspects of the disclosure may include one or more of the following optional features. In some implementations, determining the at least one axis of freedom and the at least one constrained axis includes determining a task space model for the constrained object using the measured task parameter set. In some examples, the operations further include storing at least a portion of the measured task parameter set in a task buffer as task parameter records. In further examples, storing at least a portion of the measured task parameter set includes comparing at least one measured parameter from the measured parameter task set to a recorded parameter of one of the task parameter records of the task buffer and generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter satisfies a recording threshold. In even further examples, the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

[0009] In other further examples, the operations further include evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records. In other further examples, the operations further include evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

[0010] Yet another aspect of the disclosure provides a computer program product. The computer program product is encoded on a non-transitory computer readable storage medium connected to a robot. The robot includes an articulated arm having an end effector for engaging a constrained object. The computer program product includes instructions that when executed by a data processing apparatus cause the data processing apparatus to perform operations. The operations include receiving a measured task parameter set for the end effector. The measured task parameter set includes position parameters defining a position of the end effector.

The operations further include determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace. The operations also include assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis. Furthermore, the operations include instructing the articulated arm to move the end effector along the at least one axis of freedom.

[0011] Aspects of the disclosure may include one or more of the following optional features. In some implementations, determining at the least one axis of freedom and the at least one constrained axis includes determining a task space model for the constrained object using the measured task parameter set. In some embodiments, the operations further include storing at least a portion of the measured task parameter set in a task buffer as task parameter records. In further embodiments, storing at least the portion of the measured task parameters set includes comparing at least one measured parameter from the measured parameter task set to a recorded parameter of one of the task parameter records of the task buffer and generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter exceeds a recording threshold. In even further embodiments, the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

[0012] In other further embodiments, the operations further include evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records. In other further embodiments, the operations further include evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

[0013] The details of one or more implementations of the disclosure are set forth in the accompanying drawings and the description below. Other aspects, features, and advantages will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0014] FIG. **1**A is a schematic view of an example robot executing an arm controller for executing manipulation tasks with an arm of the robot.

[0015] FIG. **1**B is a schematic view of the arm controller of FIG. **1**A.

[0016] FIG. **2**A is schematic view of a remote device and task manager of the arm controller of FIG. **1**A.

[0017] FIG. **2**B is a schematic view of a task instructor of the arm controller of FIG. **1**A.

[0018] FIG. **3** is a schematic view of a task space estimator of the arm controller of FIG. **1**A.

[0019] FIG. **4**A is a schematic view of an example operation of a task observer of the arm controller of FIG. **1**A.

[0020] FIG. **4**B is a schematic view of another example operation of the task observer of FIG. **4**A.

[0021] FIG. **4**C is a schematic view of another example operation of the task observer of FIG. **4**A.

[0022] FIG. **5** is a flowchart of an example arrangement of operations for a method of constrained object manipulation for in a robot arm.

[0023] FIG. **6** is a schematic view of an example computing device that may be used to implement the systems and methods described herein.

[0024] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0025] Many robots include multi-axis articulable appendages configured to execute complex movements for completing tasks, such as material handling or industrial operations (e.g., welding, gluing, and/or fastening). These appendages, also referred to as manipulators, typically include an end-effector or hand attached at the end of a series appendage segments or portions, which are connected to each other by one or more appendage joints. The appendage joints cooperate to configure the appendage in a variety of poses P within a space associated with the robot. Here, the term “pose” refers to the position and orientation of the appendage. For example, the pose P of the appendage may be defined by coordinates (x, y, z) of the appendage within a workspace (Cartesian space), and the orientation may be defined by angles (Ox, Oy, Oz) of the appendage within the workspace. In use, the appendage may need to manipulate partially constrained objects by applying forces to move the object along or about one or more unconstrained axes.

[0026] Referring to FIG. **1**, a robot or robotic device **10** includes a base **12** having a body **13** and two or more legs **14**. Each leg **14** may have an upper leg portion **15** and a lower leg portion **16**. The upper leg portion **15** may be attached to the body **13** at an upper joint **17** (i.e., a hip joint) and the lower leg portion **16** may be attached to the upper leg portion **15** by an intermediate joint **18** (i.e., a knee joint). Each leg **14** further includes a contact pad or foot **19** disposed at a distal end of the lower leg portion **16**, which provides a ground-contacting point for the base **12** of the robot **10**.

[0027] In some implementations, the robot **10** further includes one or more appendages, such as an articulated arm **20** or manipulator disposed on the body **13** and configured to move relative to the body **13**. Moreover, the articulated arm **20** may be interchangeably referred to as a manipulator, an appendage arm, or simply an appendage. In the example shown, the articulated arm **20** includes two arm portions **22**, **22***a*, **22***b* rotatable relative to one another and the body **13**. However, the articulated arm **20** may include more or less arm portions **22** without departing from the scope of the present disclosure. A third arm portion **24** of the articulated arm, referred to as an end effector **24** or hand **24**, may be interchangeably coupled to a distal end of the second portion **22***b* of the articulated arm **20** and may include one or more actuators **25** for gripping/grasping objects **4**.

[0028] The articulated arm **20** includes a plurality of joints **26**, **26***a*-**26***c* disposed between adjacent ones of the arm portions **22**, **24**. In the example shown, the first arm portion **22***a* is attached to the body **13** of the robot **10** by a first two-axis joint **26***a*, interchangeably referred to as a shoulder **26***a*. A single-axis joint **26***b* connects the first arm portion **22***a* to the second arm portion **22***b*. The second joint **26***b* includes a single axis of rotation and may be interchangeably referred to as an elbow **26***b* of the articulated arm **20**. A second two axis joint **26***c* connects the second arm portion **22***b* to the hand **24**, and may be interchangeably referred to as a wrist **26***c* of the articulated arm **20**. Accordingly, the joints **26** cooperate to provide the articulated arm **20** with five degrees of freedom (i.e., five axes of rotation). While the illustrated example shows a five-axis articulated arm **20**, the principles of the present disclosure are applicable to robotic arms having any number of axes. Furthermore, the principles of the present disclosure are applicable to robotic arms mounted to different types of bases, such as mobile bases including one or more wheels or stationary bases.

[0029] The robot **10** also includes a vision system **30** with at least one imaging sensor or camera **31**, each sensor or camera **31** capturing image data or sensor data of the environment **2** surrounding the robot **10** with an angle of view **32** and within a field of view **34**. The vision system **30** may be configured to move the field of view **34** by adjusting the angle of view **32** or by panning and/or tilting (either independently or via the robot **10**) the camera **31** to move the field of view **34** in any direction. Alternatively, the vision system **30** may include multiple sensors or cameras **31** such that the vision system **30** captures a generally 360-degree field of view around the robot **10**. The camera(s) **31** of the vision system **30**, in some implementations, include one or more stereo cameras (e.g., one or more RGBD stereo cameras). In other examples, the vision system **30** includes one or more radar sensors such as a scanning light-detection and ranging (LIDAR) sensor, or a scanning laser-detection and ranging (LADAR) sensor, a light scanner, a time-of-flight sensor, or any other three-dimensional (3D) volumetric image sensor (or any such combination of sensors). The vision system **30** provides image data or sensor data derived from image data captured by the cameras or sensors **31** to the data processing hardware **36** of the robot **10**. The data processing hardware **36** is in digital communication with memory hardware **38** and, in some implementations, may be a remote system. The remote system may be a single computer, multiple computers, or a distributed system (e.g., a cloud environment) having scalable/elastic computing resources and/or storage resources.

[0030] In the example shown, the robot **10** executes an arm controller **100** on the data processing hardware **36** of the robot. In some implementations, at least a portion of the arm controller **100** executes on a remote device **40** in communication with the robot **10**. For instance, a model **342** of the constrained task space may be computed on a remote device **40** and a control system executing on the robot **10** may receive the model and determine the limited torque requests using the model. Optionally, the arm controller **100** may execute on a remote device **40** and the remote device **40** may provide an object manipulation task request **44** to the robot **10** to move/control the articulated arm **20** for manipulating a constrained object **4**.

[0031] The arm controller **100** of the robot **10** controls moving the articulated arm **20** between arm poses P_{20}. For instance, the articulated arm **20** may need to move from a start pose P_{20} to a target pose P_{20} when the robot **10** is executing the task request **44**. For instance, in a scenario when the robot **10** needs to open a door while navigating in an environment, the robot arm controller **100** will need to move the articulated arm **20** from a first pose P_{20} where the door is in a closed position to a second pose P_{20} where the door is in an open position.

[0032] The arm controller **100** may include a task manager **200**, a task space estimator **300**, and a task observer **400**. The task manager **200** receives or obtains a task request **44** for manipulating a constrained object **4** and generates task instructions **222** (FIG. **2**A) including impedance parameters **238** (e.g., stiffness) (FIG. **2**A) and path parameters **248** (e.g., force, position) (FIG. **2**A) for executing the task request **44**. The task space estimator **300** is configured to receive a measured task parameter set **322** (e.g., position, force, speed) (FIGS. **2**A-**3**) from the robot arm **20** and generate a task space model **342** (FIG. **3**) defining the degrees of freedom and/or constraints associated with the task request **44**. The task observer **400** evaluates the measured task parameter set **322** of the robot arm **20** during execution of the task request **44** to determine whether the robot arm **20** is successfully executing the task request **44**.

[0033] Movements and poses of the robot **10** and robot appendages **14**, **20** may be defined in terms of a robot workspace based on a Cartesian coordinate system. In the example of the robot **10** provided in FIG. **1**A, the robot workspace may be defined by six dimensions including the translational axes x, y, z and rotational axes Θ_x, Θ_y, Θ_z (SE(3) manifolds). As discussed below, actions of the robot **10** and/or the robot arm **20** may be defined using lower-dimensional spaces or manifolds including less axes than the number of axes (six) of the workspace. For example, the task request **44** may be constrained to a single axis within the workspace so that path parameters **248** can be efficiently computed along the single axis. Appendages **14**, **20** of the robot **10** may also be described in terms of a joint space, which refers to a space representing all possible combinations of joint configurations of a robot appendage, and is directly related to the number of degrees of freedom of the robot appendage. For instance, a robot arm having n degrees of freedom will have an n-dimensional joint space. In the present example, the articulated arm has five degrees of freedom defining a five-dimensional joint space.

[0034] With reference to FIG. **1**B, the task manager **200** includes a task interpreter **210** configured to receive or obtain task requests **44** from the remote device **40** and generate translated task requests **212** executable by the arm controller **100** to move the robot arm **20**. The task manager **200** further includes a task instructor **220** configured to generate task instructions **222** using the translated task request **212** and/or the task space model **342** provided by the task space estimator **300**. Thus, as described later, the task manager **200** generally receives a task request **44** and generates task instructions **222** that comply with the task space model **342**.

[0035] FIG. **2**A shows an example operation of the task interpreter **210** generating the translated task request **212** based on the task request **44** received or obtained from the remote device **40**. A user may interact with a user interface **42** displayed on a screen in communication with the remote device **40** to select task characteristics **46**, **46*a-d*** for the task request **44**. For example, the user interface **42** may graphically display one or more buttons **46*a*** for defining the type of object **4** (e.g., crank, door, drawer, etc.) for manipulation and one or more buttons **46*b*** for selecting task parameters (e.g., speed, force, direction, etc.). The user interface **42** may also include a force application window **46*c*** including a graphical representation of the robot **10** or the robot arm **20**. A user may use touch gestures to indicate a general direction for moving the robot arm **20** or applying force, such as an initial direction of movement for the arm **20** or initial direction for applying force. For example, a user may swipe to the left from the displayed end effector **24** to indicate a desired initial movement of the end effector **24** to the left. The user interface **42** also includes a task location window

46*d* for displaying a location in the robot environment **2** corresponding to the initial movement of the end effector **24**. The task location window **46***d* may be based on the image data from the camera **31** of the robot **10**. Thus, a user can select a location within the field of view **34** of the robot **10**.

[0036] In the example of FIG. **2**A, the user interface **42** is shown with user inputs or selection points **48**-*c* corresponding to each of the task characteristics **46**. For example, the inputs **48** include a first input **48***a* identifying the constrained task type **46***a* as being a door and a second input **48***b* selecting that task parameters **46***b* include moving the arm **20** in a downward or backward direction depending on the constrained task type **46***a* (e.g., a door would be backward, a switch would be downward). At a third input **48***c*, the user selects the right-hand side of the end effector **24** for applying the selected task parameters **46***b*. Thus, the task request **44** generally includes characteristics **46** for pulling a door handle in a backward direction using the end effector **24**.

[0037] While presenting the user interface **42** with gesture-based buttons **46***a*, **46***b* and selection windows **46***c*, **46***d* simplifies user control of the robot arm **20** by presenting an intuitive interface, the task characteristics **46** included in the task request **44** may not be directly executable by the robot arm **20**. Accordingly, the task interpreter **210** of the task manager **200** receives the task request **44** and translates the task characteristics **46** into translational and/or rotational coordinates based on the robot workspace. For example, the task interpreter **210** may translate the user-selected backward direction **48***b* associated with the task type **46***a* of opening a door into movements along the x-y plane of the workspace (FIG. **1**). The translated task request **212** is then sent to the task instructor **220**.

[0038] Referring to FIG. **1**B, the task instructor **220** includes an impedance manager **230** for assigning the impedance parameters **238** to each Cartesian dimension of the end effector **24** and a path manager **240** for assigning instructed path parameters **248** to the end effector **24**. The task instructor **220** generates task instructions **222** including the impedance parameters **238** and the path parameters **248**. In FIG. **2**A, the task instructor **220** receives the translated task request **212** and generates an initial iteration of task instructions $\mathbf{222}_i$ for moving the robot arm **20** according to the task request **44**. The initial iteration of the task instructions $\mathbf{222}_i$ may be based on user-defined task parameters **238**, **248** and/or the translated task request **212** since the task space model **342** associated with the task request **44** (i.e., the degrees of freedom associated with the task type **46***a*) is undefined. For example, the impedance manager **230** may assign a user-defined impedance setpoint **232** as the instructed impedance parameters $\mathbf{238}_i$ until the task space estimator **300** has sufficient data about the task request **44** to define the task space model **342** for the task request **44**. The path manager **240** may generate path parameters $\mathbf{248}_i$ based on the directional and rotational coordinates of the translated task request **212** corresponding to the one or more of the user inputs **48***a*-**48***c*. The initial task instructions $\mathbf{222}_i$ are then sent to the robot arm **20** for initiating the task request **44**.

[0039] Referring to FIG. **1**B, the robot arm **20** includes one or more sensors that measure the position, force, and/or velocity parameters of the robot arm **20** after each iteration of the task instructions **222** and generate a corresponding measured task parameter set **322**. Each iteration of the measured task parameter set **322** is sent to the task space estimator **300**, which uses the measured task parameter set **322** to generate the task space model **342** associated with the task request **44**. Additionally or alternatively, the task space estimator **300** may receive or obtain the path parameters **248** from the task instructor **220** to generate the task space model **342** using the same steps as described below with respect to the measured task parameter set **322**.

[0040] In FIGS. **1**B and **3**, a task recorder **320** receives each iteration of the measured task parameter set **322** and analyzes the received iteration of the measured task parameter set $\mathbf{322}_i$ to determine whether to store the measured task parameter set $\mathbf{322}_i$ as a new task parameter record **324** at a task buffer **330**, e.g., residing on the memory hardware **38** of the robot **10**. Generally, the task recorder **320** is configured to generate and store a new task parameter record **324** based on force parameters F_{322} and/or position parameters P_{322} of the measured task parameter set $\mathbf{322}_i$. Thus, task parameter records **324** are not added to the task buffer **330** for every iteration of the measured task parameter set **322**, but only when a value of one of the parameters of the measured task parameter set $\mathbf{322}_i$ satisfies a recording threshold Δ_{324}.

[0041] In one example, the task recorder **320** generates a new task parameter record **324** based on a measured position parameter P_{322i}. The task recorder **320** stores the first iteration of the measured task parameter set $\mathbf{322}_{i1}$ as an initial task parameter record $\mathbf{324}_{i+1}$. For subsequent iterations, the task recorder **320** compares the measured position parameter P_{322i} of the received iteration of the measured task parameter set $\mathbf{322}_i$ against a position parameter P_{324i-1} of the last-stored task parameter record $\mathbf{324}_{i-1}$ in the task buffer **330**. The task recorder **320** generates and stores, in the task buffer **330**, a new task parameter record $\mathbf{324}_i$ when the measured position parameter P_{322i} of the measured task parameter set $\mathbf{322}_i$ is different from the stored position parameter P_{324i-1} of the last-stored task parameter record $\mathbf{324}_{i-1}$ by a position-based recording threshold Δ_{324P} set by the robot user. Thus, task parameter records **324** are not added to the task buffer **330** for every iteration of the task request **44**, but only when the change in position exceeds the recording threshold Δ_{324P}.

[0042] Additionally or alternatively, the task recorder **320** may optionally generate and add task parameter records **324** based on the measured force parameter F_{322i} of the measured task parameter set $\mathbf{322}_i$. The task recorder **320** compares a measured force parameter F_{322i} of the received iteration of the measured task parameter set $\mathbf{322}_i$ against a stored force parameter F_{324i-1} of the last-stored task parameter record $\mathbf{324}_{i-1}$ in the task buffer **330**. The task recorder **320** generates and stores a new task parameter record $\mathbf{324}_i$ when the measured force parameter F_{322i} of the measured task parameter set $\mathbf{322}_i$ is different from the stored force parameter F_{324i-1} of the last-stored task parameter record $\mathbf{324}_{i-1}$ by a force-based recording threshold Δ_{324F} calibrated by the robot operator.

[0043] As with position-based recording, force-based recording results in records **324** being added to the task buffer **330** only when the record **324** represents movement along a path associated with the task (e.g, horizontal arc of a door, vertical arc of a switch). In other words, the force-based recording may filter measured task parameter set $\mathbf{322}_i$ where the measured forces F_{322i} are too high, as high forces may be associated with a constrained axis of the manipulated object **4**. For example, where the task request **44** is associated with pulling a switch along a vertically-oriented arcuate switch path, the switch may have a known pull-force (i.e., the force threshold F_{thresh}) associated with the arcuate switch path.

[0044] Using position-based task parameter recording and/or force-based task parameter recording ensures that task parameter records **324** that are added to the task buffer **330** represent actual changes in position of the robot arm **20** along the axes of freedom of the object **4**, which can then be evaluated by the task space generator **340** to determine actual movement of the robot arm **20** and the task space model **342**. In contrast, using time-based or velocity-based storage may result in storage of a large number of records **324** associated with a relatively small change in position and/or undesired changes in position along constrained axes of the object **4**.

[0045] Referring to FIG. **3**, the task buffer **330** is configured as a circular buffer **330** having a fixed-sized queue and a first-in-first-out (FIFO) data characteristic. Thus, the task buffer **330** only stores a task parameter record set **332** including a predetermined number n of measured position parameter records **324**. When a new task parameter record $\mathbf{324}_i$ is added to the task buffer **330** by the task recorder **320**, then the last task parameter record $\mathbf{324}_{i-n}$ is deleted from the task buffer **330**. In the illustrated example, the task parameter record set **332** includes six populated task parameter records $\mathbf{324}_{i-n}$ arranged between a write pointer **331***a* and a read pointer **331***b*. New task parameter records **324** are added to vacant record slots in front of the write pointer **331***a*. As discussed below, the size of the task parameter record set **332** (i.e., the number of task parameter records **324**) may be tuned by a user depending on desired characteristics of the task space model **342** and/or task instructions **222**. Larger task parameter record sets **332** will provide a more complete representation of the task position history, but including too many task parameter records **324** may result in the older task parameter records **324** inaccurately representing the current trajectory of the end effector **24**.

[0046] When a new task parameter record $\mathbf{324}_i$ is added to the task buffer **330**, the task space generator **340** obtains the current task parameter record set $\mathbf{332}_i$ including the added task parameter record **324** and generates a task space model **342** based on the task parameter record set $\mathbf{332}_i$. The task space generator **340** evaluates the task parameter record set $\mathbf{332}_i$ to determine main axes or a plane along which the task parameter records **324** of the task parameter record set $\mathbf{332}_i$ are best-fit. For example, in FIG. **3**, the task space generator **340** may decompose the task parameter record set $\mathbf{332}_i$, which includes the task parameter records **324** including translational (x, y, z) and/or rotational (Θ_x, Θ_y, Θ_z) coordinates, into a lower-dimensional (e.g., one-dimensional, two-dimensional) task space model **342**. The task space generator **340** may use singular value decomposition (SVD) to determine the axes or plane of freedom associated with the task parameter record set $\mathbf{332}_i$. For example, where the task request **44** includes opening a door **4**, the task space generator **340** decomposes the task parameter record set $\mathbf{332}_i$ into a two-dimensional task space model **342***a* projected to the x-y plane of the workspace. In other examples, such as opening a drawer, the task space generator **340** generates a single-dimensional task space model **342***b* projected to a single axis of the workspace (e.g., the y-axis). In yet another example where the task request **44** is associated with turning a wheel or crank, the task space generator **340** generates a two-dimensional task space model **342***c* on the x-z plane.

[0047] Each iteration of the task space model $\mathbf{342}_i$ is sent to or obtained by the task instructor **220**, which uses the iteration of the task space model $\mathbf{342}_i$ to generate a new iteration of task instructions $\mathbf{222}_{i+1}$ for the robot arm **20**. As shown in FIG. 2B, the task space model $\mathbf{342}_i$ includes an arcuate path based on the task parameter record set $\mathbf{332}_i$ (FIG. 3B) that is constrained to the x-y plane. Thus, as discussed previously, the task space model $\mathbf{342}_i$ provides the path manager **240** with a lower-dimensional task space for determining a new iteration of task instructions $\mathbf{222}_{i+1}$.

[0048] With reference to FIG. 2B, the path manager **240** includes a task path estimator **242** and a task path filter **244**. The task path estimator **242** evaluates the task space model **342** and generates a task path model **243**, $\mathbf{243}_i$ representing the estimated task direction for the next iteration of the task instructions $\mathbf{222}_{i+1}$. Generating the task path model $\mathbf{243}_i$ for each iteration may include further decomposing the task space model $\mathbf{342}_i$ into the lower-dimensional task path model **243** (i.e., from two-dimensional plane to one-dimensional axis). In one example, the task path model **243** includes a one-dimensional tangent to the two-dimensional task space model $\mathbf{342}_i$. Here, the task path generator **242** extrapolates the task instructions **222** to determine the current location of the end effector **24**. The tangent is then taken at the point of the task space model $\mathbf{342}_i$ associated with the current position of the end effector **24**.

[0049] The task path filter **244** determines whether the task path model $\mathbf{243}_i$ complies with a path model quality threshold. For example, the task path filter **244** may compare or fit the task path model $\mathbf{243}_i$ to a previous iteration of the task path model $\mathbf{243}_{i-1}$ and/or the task space model $\mathbf{342}_i$ to determine the quality of the task path model $\mathbf{243}_i$. Where the task path model $\mathbf{243}_i$ exceeds a threshold value (e.g. an error value), the task path filter **244** may discard the current iteration of the task path model $\mathbf{243}_i$ and select the previous iteration of the task path model $\mathbf{243}_{i-1}$.

[0050] The task path filter **244** sends the filtered task path model **243***f* (i.e., either current or previous iteration of the task path model $\mathbf{243}_i$, $\mathbf{243}_{i-1}$) to the task path instructor **246**, which generates new path parameters $\mathbf{248}_{i+1}$ for the robot arm **20**. The new path parameters $\mathbf{248}_{i+1}$ are based on the lower-dimensional filtered task path model **243***f* and include force or position parameters for moving the end effector **24** along the direction of the filtered task path model **243***f*. Thus, the task path instructor **246** simply applies a force along the direction tangent to the path (e.g., door arc) associated with the task request **44**. By decomposing the measured task parameter sets **322** (e.g., three-dimension) into the task space model **342** (e.g., two-dimensional) and then into the task path model **243** (e.g., one-dimensional), the task instructor **220** can quickly compute the task instructions **222** for manipulating the constrained object **4** by applying forces only along the axes of freedom of the object **4**.

[0051] The impedance manager **230** receives or obtains the task space model $\mathbf{342}_i$ from the task space estimator **300** and determines the impedance (i.e., stiffness) of the end effector **24** of the robot arm **20** for the current task request **44**. Generally, the impedance manager **230** is configured to evaluate the task space model $\mathbf{342}_i$ and to assign lower impedance to the end effector **24** along axes that the arm controller **100** expects the end effector **24** to travel and to assign higher impedance along axes that the arm controller **100** expects the end effector **24** to be constrained. Thus, in the present example where the task space model $\mathbf{342}_i$ lies along the x-y plane, the impedance manager **230** assigns relatively low impedance values (i.e., joint stiffness) to the end effector **24** along the x-axis and the y-axis. Conversely, the impedance manager **230** assigns a relatively high impedance value (i.e., stiffness) along the z-axis since the task space model $\mathbf{342}_i$ indicates that the object **4** is constrained along z-direction. Assigning low impedance values along the free axes of the task space model $\mathbf{342}_i$ and high impedance values along the constrained axes of the task space model $\mathbf{342}_i$ allows the end effector **24** to rotate or pivot along the task path as the arm **20** executes the task request **44** (e.g., opening the door) while maintaining stiffness along directions that are not expected to have movement.

[0052] In addition to evaluating the task path model $\mathbf{243}_i$ to determine the next iteration of impedance parameters $\mathbf{238}_{i+1}$, the impedance manager **230** may consider other inputs in determining the impedance parameters $\mathbf{238}_{i+1}$. For example, the impedance manager **230** may select impedance parameters $\mathbf{238}_{i+1}$ based on the impedance setpoint **232** provided by the user. Additionally or alternatively, the impedance manager **230** may consider the input task parameters **48***b* received from the remote device **40**. Where the impedance manager **230** does not receive or obtain the task space model **342**, such as in the first iteration of task instructions $\mathbf{222}_{i+1}$, or where the impedance manager **230** has a relatively low confidence in the task space model **342**, such as when the task space model **342** has a high level of noise or inconsistent task parameter records **324**, the impedance manager **230** may give greater weight to the impedance setpoint **232** and/or the input task parameters **48***b* to determine the impedance values for the next iteration of impedance parameters $\mathbf{238}_{i+1}$. For example, where the user selects a leftward direction **48***b* corresponding to the x-y plane for the force and the impedance manager **230** does not have a task path model $\mathbf{243}_i$, the impedance manager **230** will determine and assign impedance values based on the task input **48***b* and/or the impedance setpoint **232**.

[0053] The new iteration of the impedance parameters $\mathbf{238}_{i+1}$ and the path parameters $\mathbf{248}_{i+1}$ based on the current iteration of the task space model $\mathbf{342}_i$ are sent to the robot arm **20** as a new iteration of task instructions $\mathbf{222}_{i+1}$. As provided above, the path parameters $\mathbf{248}_{i+1}$ simply instruct the robot arm **20** to move the end effector **24** along the free direction of lower-dimensional task path model **243** while the impedance parameters $\mathbf{238}_{i+1}$ instruct the end effector **24** to have a low stiffness along the modeled task space model **342** and a high stiffness transverse to the modeled task space model **342**. Thus, the task instructions $\mathbf{222}_{i+1}$ allow the end effector **24** to execute the task request **44** with minimal computation required at arm controller **100** by allowing the end effector **24** to follow the constrained path of the object **4** associated with the task request **44** (e.g., door arc, wheel circle, drawer axis). After the robot **10** executes the task instructions $\mathbf{222}_{i+1}$, a new iteration of a measured task parameter set $\mathbf{322}_{i+1}$ is transmitted to the task space estimator **300** for evaluation and generating an updated iteration of the task space model $\mathbf{342}_{i+1}$.

[0054] Referring to FIGS. **1**B and **4**A, the task observer **400** is configured to continuously evaluate the measured task parameter set **322** (e.g., position, speed) of the end effector **24** to determine whether the end effector **24** is still engaged with the constrained object **4**. The task observer **400** then generates a task status **402** identifying whether the end effector **24** is engaged and transmits the task status **402** to the task instructor **220**. When the task status **402** indicates that the end effector **24** is disengaged from the constrained object **4**, the task instructor **220** may terminate the task request **44** and cease generating new task instructions **222** and revert to a safe stopping behavior until the end effector **24** can be reengaged with the object **4**.

[0055] Generally, the task observer **400** evaluates the most-recent task parameter records $\mathbf{324}_i$ and determines the task status **402** based on whether the measured position P_{322} and/or velocity V_{322} of the end effector **24** satisfies task criteria. For example, the task observer **400** may evaluate the task parameter records **324** to determine whether the end effector **24** fits the task space model **342** and/or satisfies a velocity threshold V_{thresh}.

[0056] In FIG. 4A, the task observer **400** includes a task evaluator **410** that receives the current task parameter record set $\mathbf{332}_i$ and/or the task space model $\mathbf{342}_i$ including the task parameter records **324**. The task evaluator **410** may segregate the task parameter records **324** into an evaluation record set **412** including a number of the most-recent task parameter records **324** and a model record set **414** including a number of the records **324** immediately preceding the records **324** selected for the evaluation record set **412**. In the illustrated example, the evaluation record set **412** includes the four most-recent task parameter records **324** and the model record set **414** includes the ten task parameter records **324** preceding the evaluation record set **412**. However, the task evaluator **410** may be tuned to select any number of records **324** for the evaluation record set **412** and the model record set **414**. The task evaluator **410** checks a fit of the evaluation record set **412** against the model record set **414**. When the quality of the fit of the evaluation record set **412** exceeds a fit quality threshold, the task evaluator **410** determines that the end effector **24** is disengaged (e.g., lost grip) from the constrained object **4**.

[0057] The task evaluator **410** may be configured to determine a quality of the current iteration of the task parameter record set $\mathbf{332}_i$ and to evaluate the task parameter record set $\mathbf{332}_i$ based on the quality. For example, as previously discussed, the task evaluator **410** segregates the task parameter records set $\mathbf{332}_i$ into an evaluation record set **412** including a first number of task parameter records **324** and a model record set **414** including a second number of task parameter records **324**. The task evaluator **410** determines the quality of the task parameter record set $\mathbf{332}_i$ based on whether the task parameter record set $\mathbf{332}_i$ includes a quantity of task parameter records **324** needed to populate the evaluation record set **412** and the model record set **414**. Depending on the number of records **324** in the task parameter record set $\mathbf{332}_i$, the task evaluator **410** designates the task parameter record set $\mathbf{332}_i$ as an optimal task parameter record set **332***a* (FIG. 4A), a sufficient record set **332***b* (FIG. 4B), or an insufficient task parameter record set **332***c* (FIG. 4C).

[0058] In the present example, the evaluation record set **412** includes four (4) slots for task parameter records **324** and the model record set **414** includes ten (10) slots for task parameter records **324**. The task evaluator **410** determines that a task parameter record set $\mathbf{332}_i$ is an optimal task parameter record set **332***a* (FIG. 4A) when the task parameter record set $\mathbf{332}_i$ includes enough task parameter records **324** to populate each of the record sets **412**, **414** with unique task parameter records **324** (i.e., no shared task parameter records between record sets **412**, **414**). Thus, in the present example, an optimal task parameter record set **332***a* includes at least fourteen (14) task parameter records **324** for uniquely populating the record sets **412**, **414**, as shown in FIG. **4**A.

[0059] In another scenario (FIG. 4B), the task evaluator **410** determines that a current iteration of the task parameter record set $\mathbf{332}_i$ is a sufficient task parameter record set **332***b* (i.e., less than optimal) when the task parameter record set $\mathbf{332}_i$ includes at least enough task parameter records **324** to populate each of the evaluation record set **412** and the model record set **414**, but with some of the task parameter records **324** of the sufficient task parameter record set **332***b* being shared between the record sets **412**, **414**. For example, where a sufficient task parameter record set **332***b* includes twelve (12) task parameter records **324** (FIG. 4B), the evaluation record set **412** is populated with the four most-recent task parameter records **324** (e.g., records 1-4) and the model record set **414** is populated with the ten (10) oldest records (e.g., records 3-12) such that the third and fourth records **324** of the sufficient task parameter record set **332***b* are included in both record sets **412**, **414**.

[0060] In another scenario (FIG. 4C), the task evaluator **410** determines that a current iteration of the task parameter record set $\mathbf{332}_i$ is an insufficient task parameter record set **332***c* (i.e., cannot be evaluated) when the task parameter record set $\mathbf{332}_i$ does not include enough task parameter records **324** to populate the model record set **414** and provide at least one unique record to the evaluation record set **412**. In the present example, the task parameter record set **332***i* would need at least eleven task parameter records **324** to populate the ten (10) slots of the model record set **414** and provide one (1) unique record to the evaluation record set **412**. For example, where an insufficient task parameter record set **332***c* includes eight (8) task parameter records **324** (FIG. 4C), the task evaluator **410** will not evaluate the task parameter record set **332***c*. If the task parameter record set $\mathbf{332}_i$ is an insufficient task parameter record set **332***c*, the task observer **400** does not evaluate the insufficient task parameter record set **332***c*.

[0061] With respect to FIG. **4**A, where the positions included in the task parameter records **324** of evaluation record set **412** do not fit the model record set **414**, the task observer **400** determines that the end effector **24** has prematurely disengaged the constrained object **4** (i.e., lost grip) and generates a task status **402** indicating that end effector **24** has failed to complete the task request **44**. In some examples, the task observer **400** evaluates the overall quality of a fit of the evaluation record set **412** by determining the mean squared error of the fit of the evaluation record set **412** to the model record set **414**. Where the mean squared error exceeds a threshold fit error value, the task observer **400**, generates the task status **402** indicating that the end effector **24** has become disengaged from the constrained object **4**. Additionally or alternatively, the task observer **400** may determine that the end effector **24** has disengaged the object **4** when a measured speed or velocity V_{322} of the end effector **24** exceeds a threshold velocity V_{thresh}. The threshold velocity V_{thresh} may be a fixed value or may be based on the measured velocities V_{322} included in the task parameter records **324** of the model record set **414**.

[0062] FIG. **5** is a flowchart of an example arrangement of operations for a method **500** for manipulating constrained objects using a robot arm **20**. The method **500** may be a computer implemented method executed by data processing hardware of the robot **10**, which causes the data processing hardware to perform operations. At operation **502**, the method **500** includes receiving a plurality of measured task parameter sets **322** for the end effector **24**, each measured task parameter set **322** including position parameters P_{322} defining a position of the end effector **24** relative to a workspace **2**. The method **500**, at operation **504**, also includes determining, using the measured task parameter sets **322**, at least one axis of freedom x, y, z, Θ_x, Θ_y, Θ_z and at least one constrained axis x, y, z, Θ_x, Θ_y, Θ_z for the end effector **24** within the workspace **2**. Another operation **506** of the method includes assigning a first impedance value **238** to the end effector **24** along the at least one axis of freedom x, y, z, Θ_x, Θ_y, Θ_z and assigning a second impedance value **238** to the end effector **24** along the at least one constrained axis x, y, z, Θ_x, Θ_y, Θ_z, the second impedance value **238** greater than the first impedance value **238**. At operation **508**, the method **500** includes instructing the articulated arm **20** to move the end effector **24** along the at least one axis of freedom x, y, z, Θ_x, Θ_y, Θ_z.

[0063] FIG. **6** is schematic view of an example computing device **600** that may be used to implement the systems and methods described in this document. The computing device **600** is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. The components shown here, their connections and relationships, and their functions, are meant to be exemplary only, and are not meant to limit implementations of the inventions described and/or claimed in this document.

[0064] The computing device **600** includes a processor **610**, memory **620**, a storage device **630**, a high-speed interface/controller **640** connecting to the memory **620** and high-speed expansion ports **650**, and a low speed interface/controller **660** connecting to a low speed bus **670** and a storage device **630**. Each of the components **610**, **620**, **630**, **640**, **650**, and **660**, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor **610** can process instructions for execution within the computing device **600**, including instructions stored in the memory **620** or on the storage device **630** to display graphical information for a graphical user interface (GUI) on an external input/output device, such as display **680** coupled to high speed interface **640**. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices **600** may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0065] The memory **620** stores information non-transitorily within the computing device **600**. The memory **620** may be a computer-readable medium, a volatile memory unit(s), or non-volatile memory unit(s). The non-transitory memory **620** may be physical devices used to store programs (e.g., sequences of instructions) or data (e.g., program state information) on a temporary or permanent basis for use by the computing device **600**. Examples of non-volatile memory include, but are not limited to, flash memory and read-only memory (ROM)/programmable read-only memory (PROM)/erasable programmable read-only memory (EPROM)/electronically erasable programmable read-only memory (EEPROM) (e.g., typically used for firmware, such as boot programs). Examples of volatile memory include, but are not limited to, random access memory (RAM), dynamic random access memory (DRAM), static random access memory (SRAM), phase change memory (PCM) as well as disks or tapes.

[0066] The storage device **630** is capable of providing mass storage for the computing device **600**. In some implementations, the storage device **630** is a computer-readable medium. In various different implementations, the storage device **630** may be a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. In additional implementations, a computer program product **700** is tangibly embodied in an information carrier. The computer program product contains instructions that, when executed, perform one or more methods, such as those described above. The information carrier is a computer- or machine-readable medium, such as the memory **620**, the storage device **630**, or memory on processor **610**.

[0067] The high speed controller **640** manages bandwidth-intensive operations for the computing device **600**, while the low speed controller **660** manages lower bandwidth-intensive operations. Such allocation of duties is exemplary only. In some implementations, the high-speed controller **640** is coupled to the memory **620**, the display **680** (e.g., through a graphics processor or accelerator), and to the high-speed expansion ports **650**, which may accept various expansion cards (not shown). In some implementations, the low-speed controller **660** is coupled to the storage device **630** and a low-speed expansion port **690**. The low-speed expansion port **690**, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet), may be coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

[0068] The computing device **600** may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server **600***a* or multiple times in a group of such servers **600***a*, as a laptop computer **600***b*, or as part of a rack server system **600***c*.

[0069] Various implementations of the systems and techniques described herein can be realized in digital electronic and/or optical circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations can include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

[0070] These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and can be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms "machine-readable medium" and "computer-readable medium" refer to any computer program product, non-transitory computer readable medium, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term "machine-readable signal" refers to any signal used to provide machine instructions and/or data to a programmable processor.

[0071] The processes and logic flows described in this specification can be performed by one or more programmable processors, also referred to as data processing hardware, executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0072] To provide for interaction with a user, one or more aspects of the disclosure can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube), LCD (liquid crystal display) monitor, or touch screen for displaying information to the user and optionally a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's client device in response to requests received from the web browser.

[0073] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the disclosure. Accordingly, other implementations are within the scope of the following claims.

What is claimed is:

1. A computer-implemented method that, when executed by data processing hardware of a robot including an articulated arm having an end effector engaged with a constrained object, causes the data processing hardware to perform operations comprising:

receiving a measured task parameter set for the end effector, the measured task parameter set including position parameters defining a position of the end effector;

determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace;

assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis; and

instructing the articulated arm to move the end effector along the at least one axis of freedom.

2. The method of claim **1**, wherein determining the at least one axis of freedom and the at least one constrained axis comprises determining a task space model for the constrained object using the measured task parameter set.

3. The method of claim **1**, wherein the operations further comprise storing at least a portion of the measured task parameter set in a task buffer as task parameter records.

4. The method of claim **3**, wherein storing at least a portion of the measured task parameter set comprises:

comparing at least one measured parameter from the measured task parameter set to a recorded parameter of one of the task parameter records of the task buffer; and

generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter satisfies a recording threshold.

5. The method of claim **4**, wherein the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

6. The method of claim **5**, wherein the operations further comprise evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records.

7. The method of claim **3**, wherein the operations further comprise evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

8. A robot comprising:

an articulated arm having an end effector for engaging a constrained object;

data processing hardware in communication with the articulated arm; and

memory hardware in communication with the data processing hardware, the memory hardware storing instructions that when executed on the data processing hardware cause the data processing hardware to perform operations comprising:

receiving a measured task parameter set for the end effector, the measured task parameter set including position parameters defining a position of the end effector;

determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace;

assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis; and

instructing the articulated arm to move the end effector along the at least one axis of freedom.

9. The robot of claim **8**, wherein determining the at least one axis of freedom and the at least one constrained axis comprises determining a task space model for the constrained object using the measured task parameter set.

10. The robot of claim **8**, wherein the operations further comprise storing at least a portion of the measured task parameter set in a task buffer as task parameter records.

11. The robot of claim **10**, wherein storing at least a portion of the measured task parameter set comprises:

comparing at least one measured parameter from the measured task parameter set to a recorded parameter of one of the task parameter records of the task buffer; and

generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter satisfies a recording threshold.

12. The robot of claim **11**, wherein the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

13. The robot of claim **12**, wherein the operations further comprise evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records.

14. The robot of claim **10**, wherein the operations further comprise evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

15. A computer program product encoded on a non-transitory computer readable storage medium connected to a robot including an articulated arm having an end effector for engaging a constrained object, the computer program product comprising instructions that when executed by a data processing apparatus cause the data processing apparatus to perform operations comprising:

receiving a measured task parameter set for the end effector, the measured task parameter set including position parameters defining a position of the end effector;

determining, using the measured task parameter set, at least one axis of freedom and at least one constrained axis for the end effector within a workspace;

assigning a first impedance value to the end effector along the at least one axis of freedom and assigning a second impedance value to the end effector along the at least one constrained axis; and

instructing the articulated arm to move the end effector along the at least one axis of freedom.

16. The product of claim **15**, wherein determining at the least one axis of freedom and the at least one constrained axis comprises determining a task space model for the constrained object using the measured task parameter set.

17. The product of claim **15**, wherein the operations further comprise storing at least a portion of the measured task parameter set in a task buffer as task parameter records.

18. The product of claim **17**, wherein storing at least the portion of the measured task parameters set comprises:

comparing at least one measured parameter from the measured task parameter set to a recorded parameter of one of the task parameter records of the task buffer; and

generating a new task parameter record when a difference between the at least one measured parameter and the recorded parameter exceeds a recording threshold.

19. The product of claim **18**, wherein the measured parameter and the recorded parameter each include a respective position parameter and/or a respective velocity parameter.

20. The product of claim **17**, wherein the operations further comprise evaluating the position parameters of the task parameter records to determine the at least one axis of freedom associated with the task parameter records.

21. The product of claim **17**, wherein the operations further comprise evaluating the task parameter records to determine whether the end effector is engaged with the constrained object.

* * * * *