US 20190283245A1

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

(10) Pub. No.: US 2019/0283245 A1
(43) Pub. Date: Sep. 19, 2019

(54) DEEP MACHINE LEARNING METHODS AND APPARATUS FOR ROBOTIC GRASPING

(71) Applicant: **Google LLC**, Mountain View, CA (US)

(72) Inventors: **Sergey Levine**, Berkeley, CA (US); **Peter Pastor Sampedro**, San Francisco, CA (US); **Alex Krizhevsky**, San Jose, CA (US)

(21) Appl. No.: **16/234,272**

(22) Filed: **Dec. 27, 2018**

Related U.S. Application Data

(63) Continuation of application No. 15/377,280, filed on Dec. 13, 2016, now Pat. No. 10,207,402.

(60) Provisional application No. 62/303,139, filed on Mar. 3, 2016.

Publication Classification

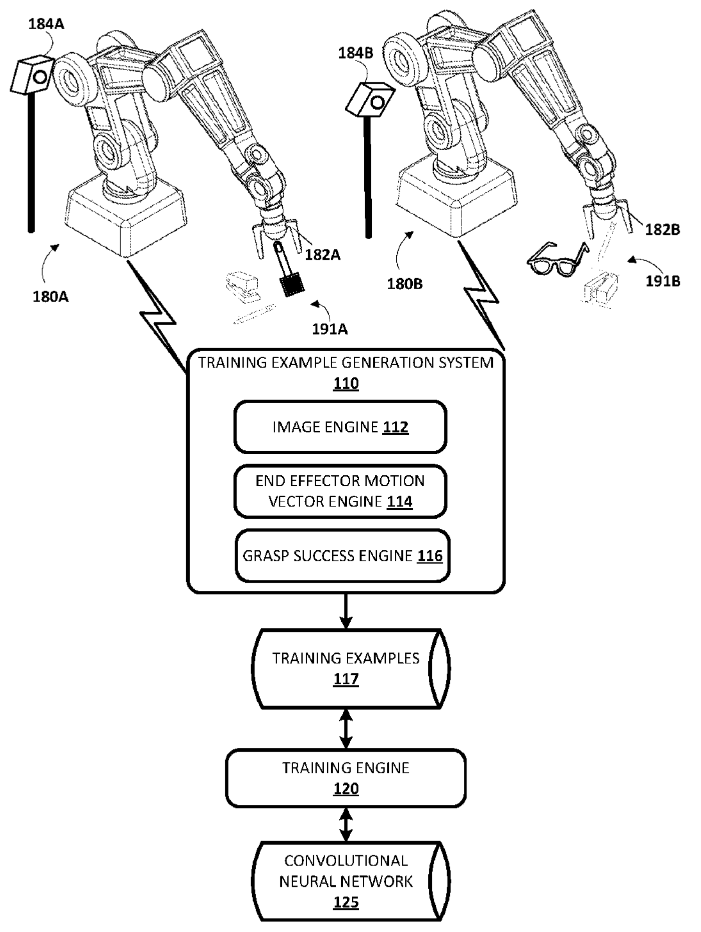
(51) **Int. Cl.**

B25J 9/16	(2006.01)
G06N 3/08	(2006.01)

(52) **U.S. Cl.**
CPC ***B25J 9/163*** (2013.01); ***G06N 3/08*** (2013.01); ***B25J 9/1664*** (2013.01); ***B25J 9/1697*** (2013.01)

(57) **ABSTRACT**

Deep machine learning methods and apparatus related to manipulation of an object by an end effector of a robot. Some implementations relate to training a deep neural network to predict a measure that candidate motion data for an end effector of a robot will result in a successful grasp of one or more objects by the end effector. Some implementations are directed to utilization of the trained deep neural network to servo a grasping end effector of a robot to achieve a successful grasp of an object by the grasping end effector. For example, the trained deep neural network may be utilized in the iterative updating of motion control commands for one or more actuators of a robot that control the pose of a grasping end effector of the robot, and to determine when to generate grasping control commands to effectuate an attempted grasp by the grasping end effector.



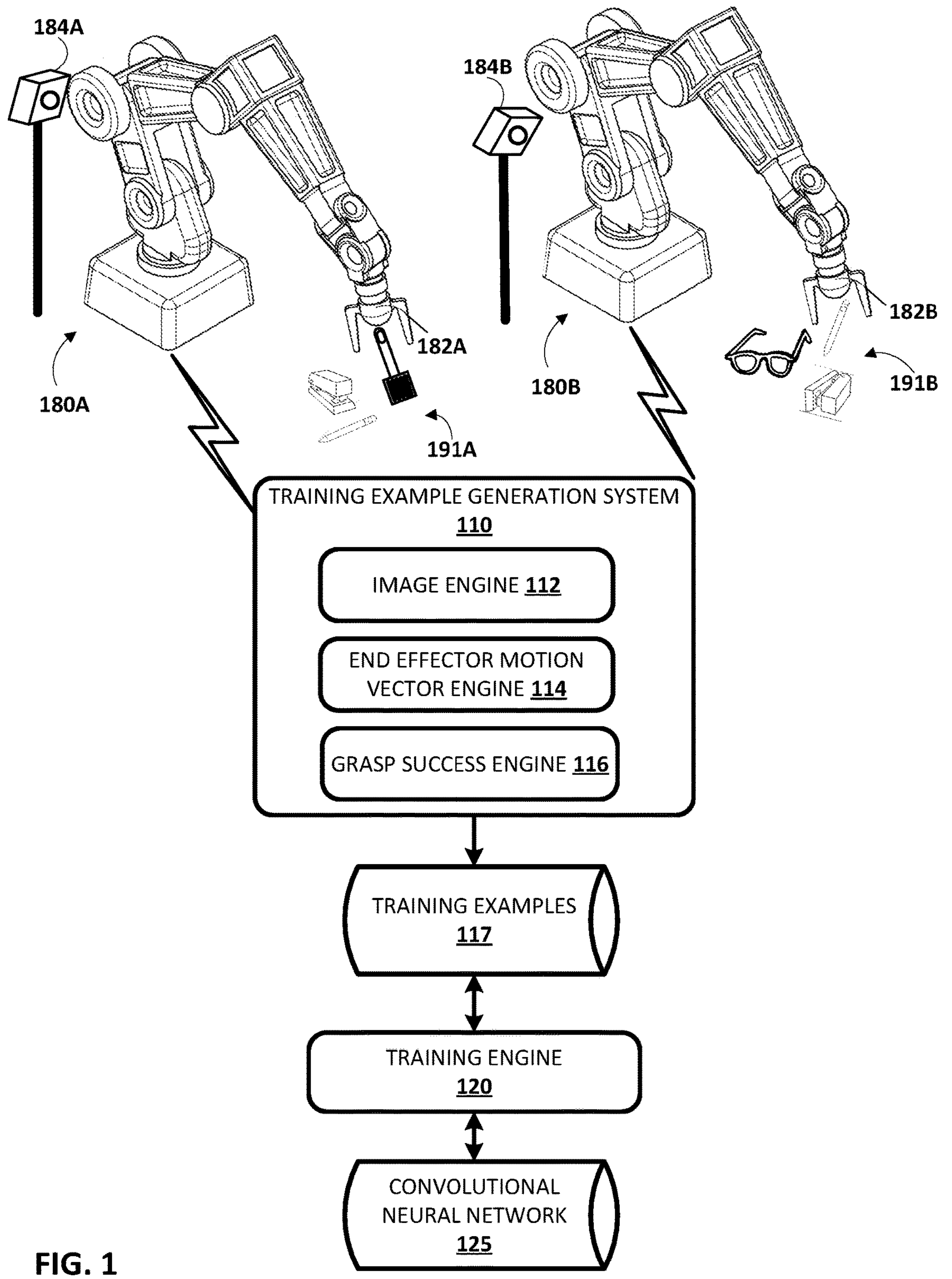

184A
184B
182A
182B
180A
180B
191A
191B
TRAINING EXAMPLE GENERATION SYSTEM 110
IMAGE ENGINE 112
END EFFECTOR MOTION VECTOR ENGINE 114
GRASP SUCCESS ENGINE 116
TRAINING EXAMPLES 117
TRAINING ENGINE 120
CONVOLUTIONAL NEURAL NETWORK 125

FIG. 1

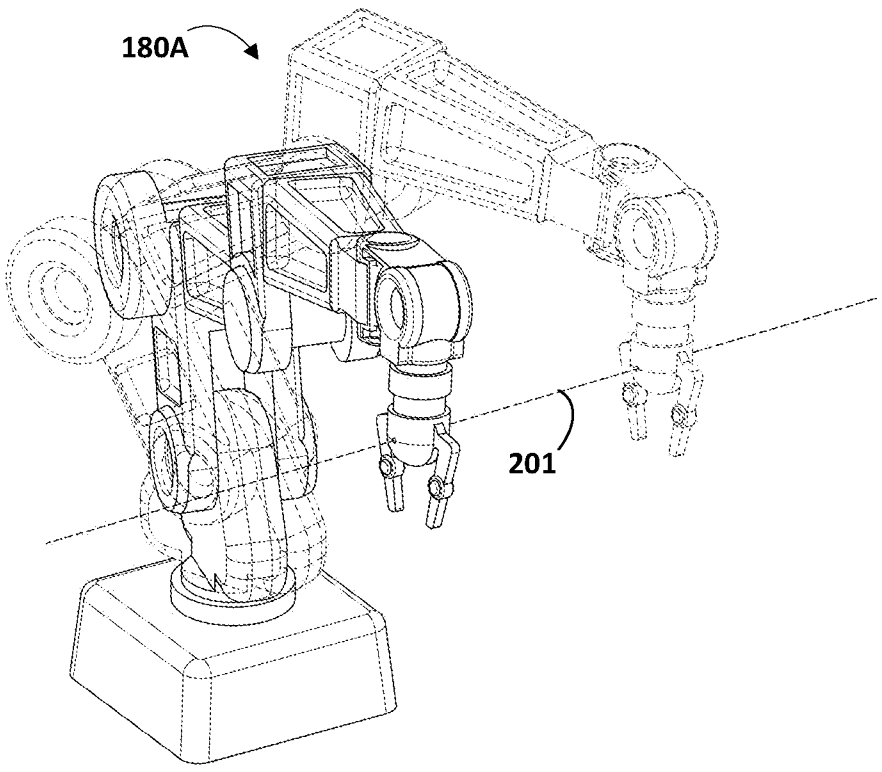


FIG. 2

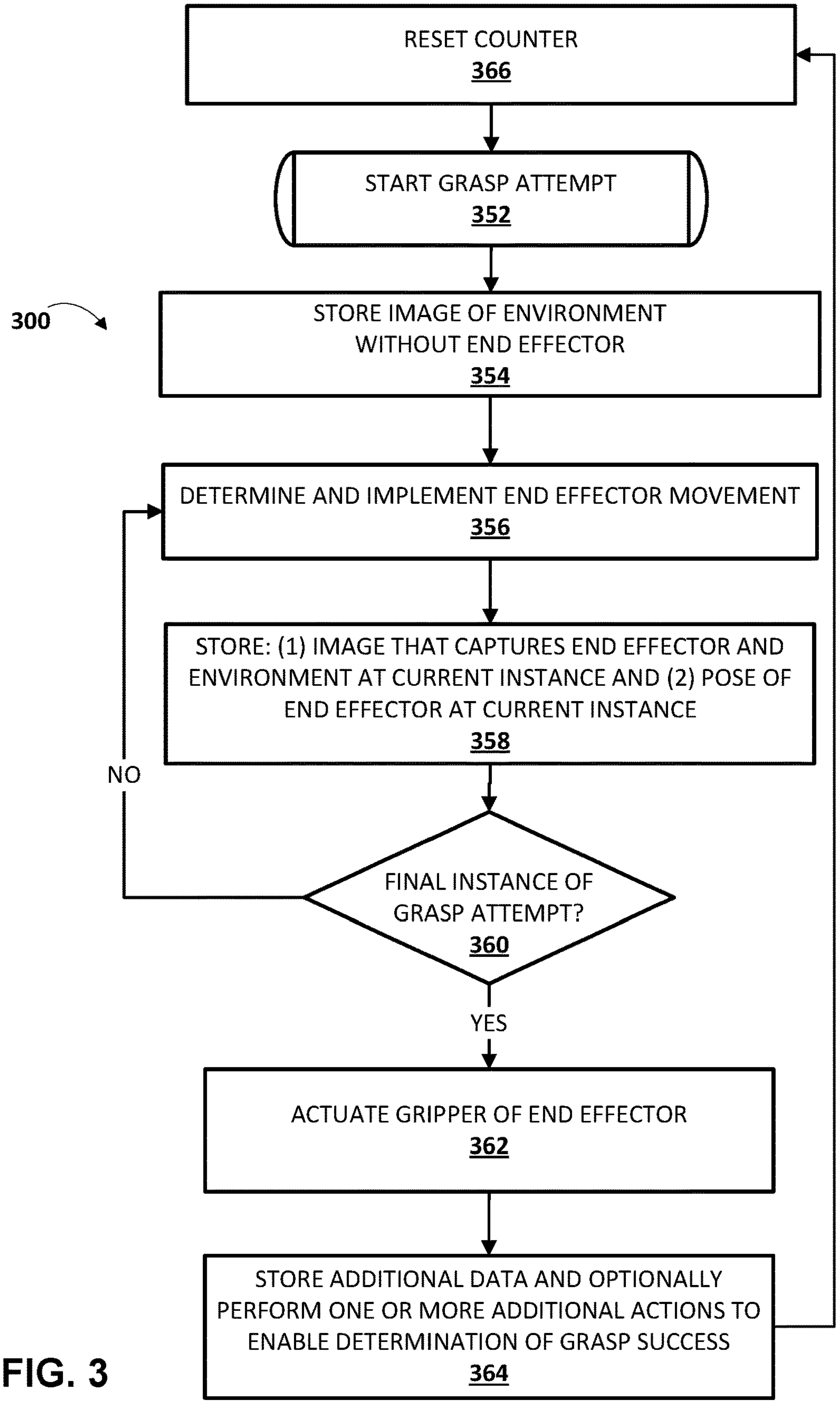

RESET COUNTER
366
START GRASP ATTEMPT
352
300
STORE IMAGE OF ENVIRONMENT WITHOUT END EFFECTOR
354
DETERMINE AND IMPLEMENT END EFFECTOR MOVEMENT
356
STORE: (1) IMAGE THAT CAPTURES END EFFECTOR AND ENVIRONMENT AT CURRENT INSTANCE AND (2) POSE OF END EFFECTOR AT CURRENT INSTANCE
358
NO
FINAL INSTANCE OF GRASP ATTEMPT?
360
YES
ACTUATE GRIPPER OF END EFFECTOR
362
STORE ADDITIONAL DATA AND OPTIONALLY PERFORM ONE OR MORE ADDITIONAL ACTIONS TO ENABLE DETERMINATION OF GRASP SUCCESS
364

FIG. 3

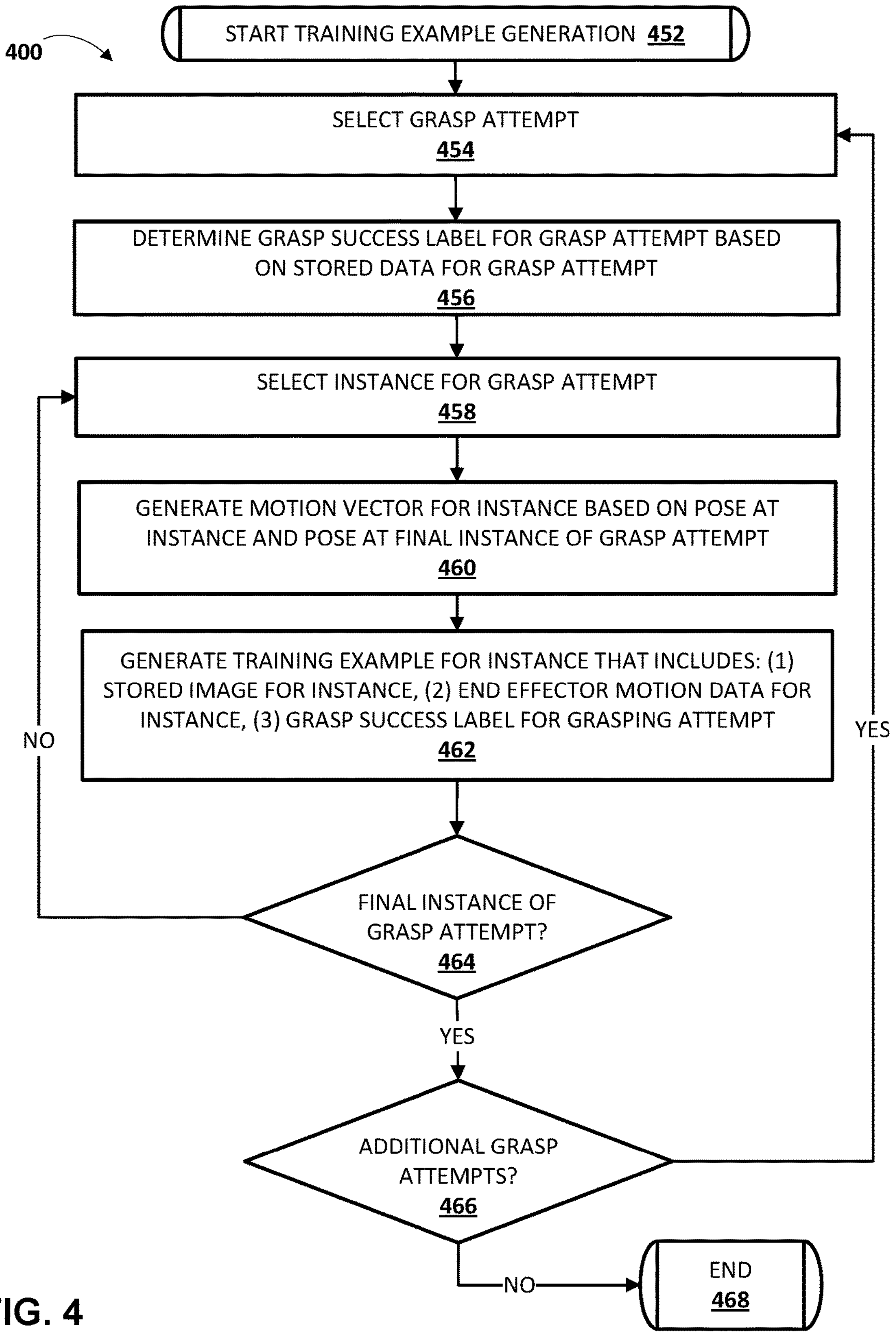

400
START TRAINING EXAMPLE GENERATION 452
SELECT GRASP ATTEMPT 454
DETERMINE GRASP SUCCESS LABEL FOR GRASP ATTEMPT BASED ON STORED DATA FOR GRASP ATTEMPT 456
SELECT INSTANCE FOR GRASP ATTEMPT 458
GENERATE MOTION VECTOR FOR INSTANCE BASED ON POSE AT INSTANCE AND POSE AT FINAL INSTANCE OF GRASP ATTEMPT 460
GENERATE TRAINING EXAMPLE FOR INSTANCE THAT INCLUDES: (1) STORED IMAGE FOR INSTANCE, (2) END EFFECTOR MOTION DATA FOR INSTANCE, (3) GRASP SUCCESS LABEL FOR GRASPING ATTEMPT 462
NO
YES
FINAL INSTANCE OF GRASP ATTEMPT? 464
YES
ADDITIONAL GRASP ATTEMPTS? 466
NO
END 468

FIG. 4

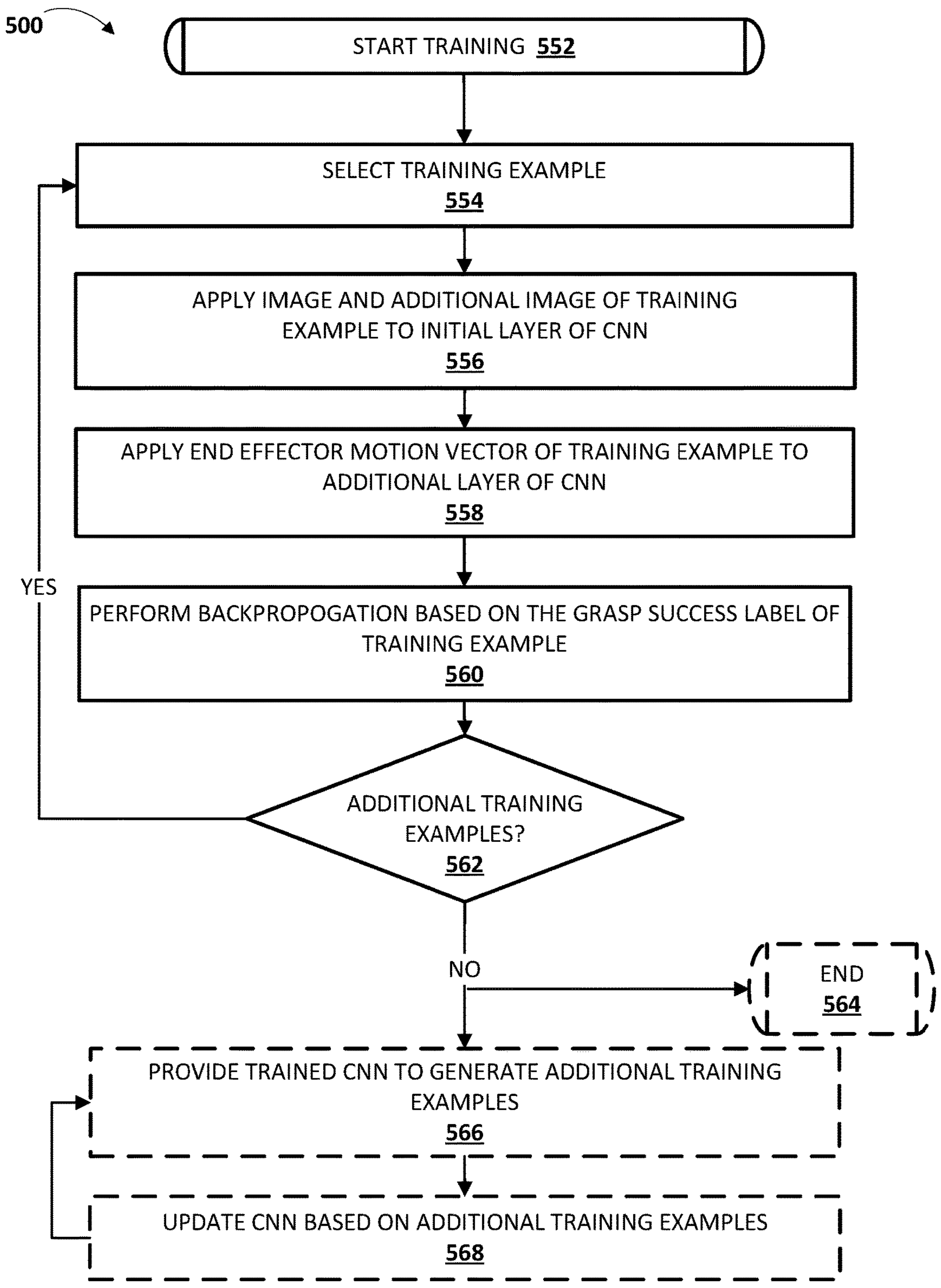

500
START TRAINING 552
SELECT TRAINING EXAMPLE 554
APPLY IMAGE AND ADDITIONAL IMAGE OF TRAINING EXAMPLE TO INITIAL LAYER OF CNN 556
APPLY END EFFECTOR MOTION VECTOR OF TRAINING EXAMPLE TO ADDITIONAL LAYER OF CNN 558
PERFORM BACKPROPOGATION BASED ON THE GRASP SUCCESS LABEL OF TRAINING EXAMPLE 560
ADDITIONAL TRAINING EXAMPLES? 562
YES
NO
END 564
PROVIDE TRAINED CNN TO GENERATE ADDITIONAL TRAINING EXAMPLES 566
UPDATE CNN BASED ON ADDITIONAL TRAINING EXAMPLES 568

FIG. 5

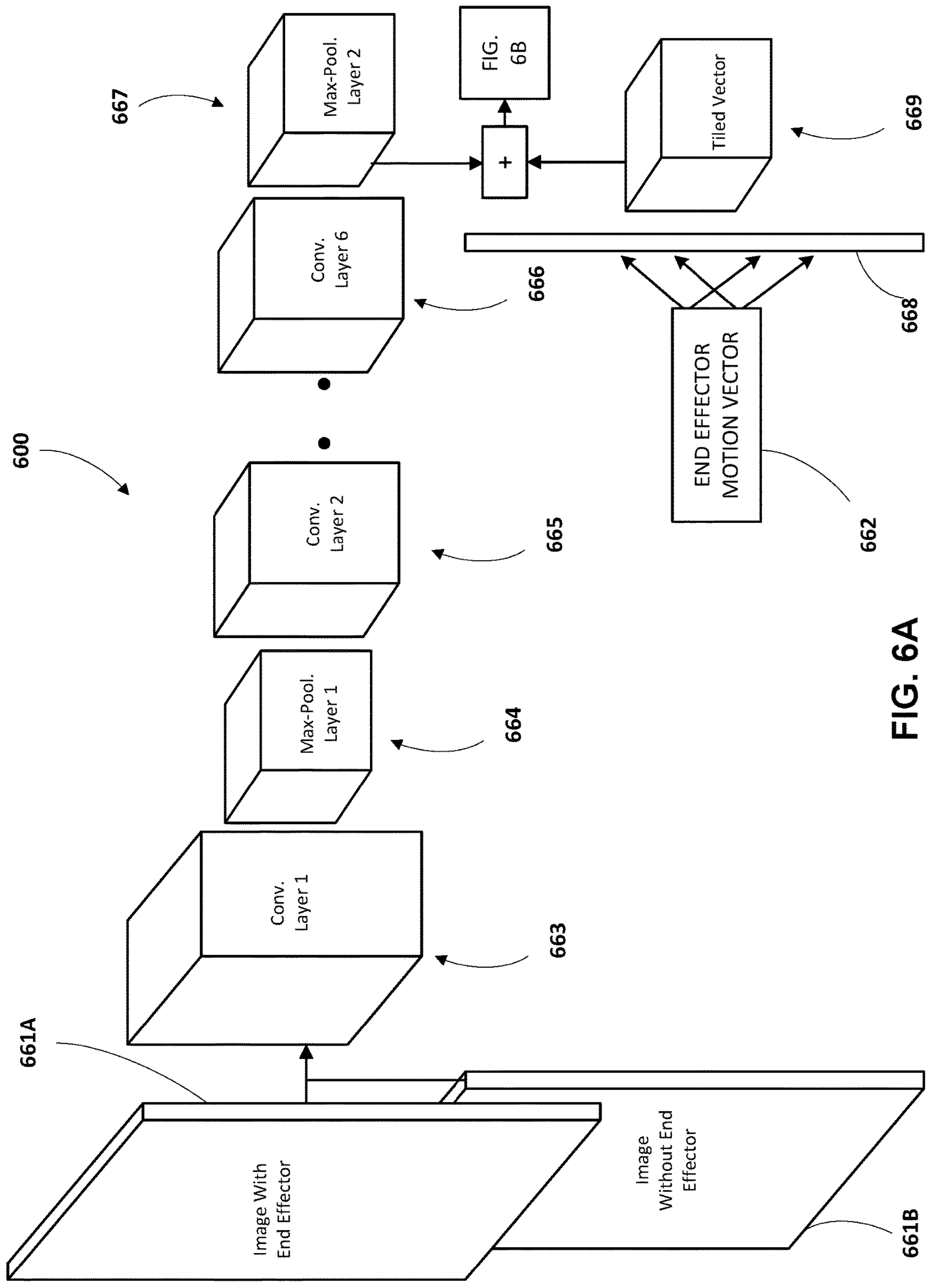

600
661A
Image With End Effector
Image Without End Effector
661B
Conv. Layer 1
663
Max-Pool. Layer 1
664
Conv. Layer 2
665
Conv. Layer 6
666
Max-Pool. Layer 2
667
FIG. 6B
+
Tiled Vector
669
END EFFECTOR MOTION VECTOR
662
668

FIG. 6A

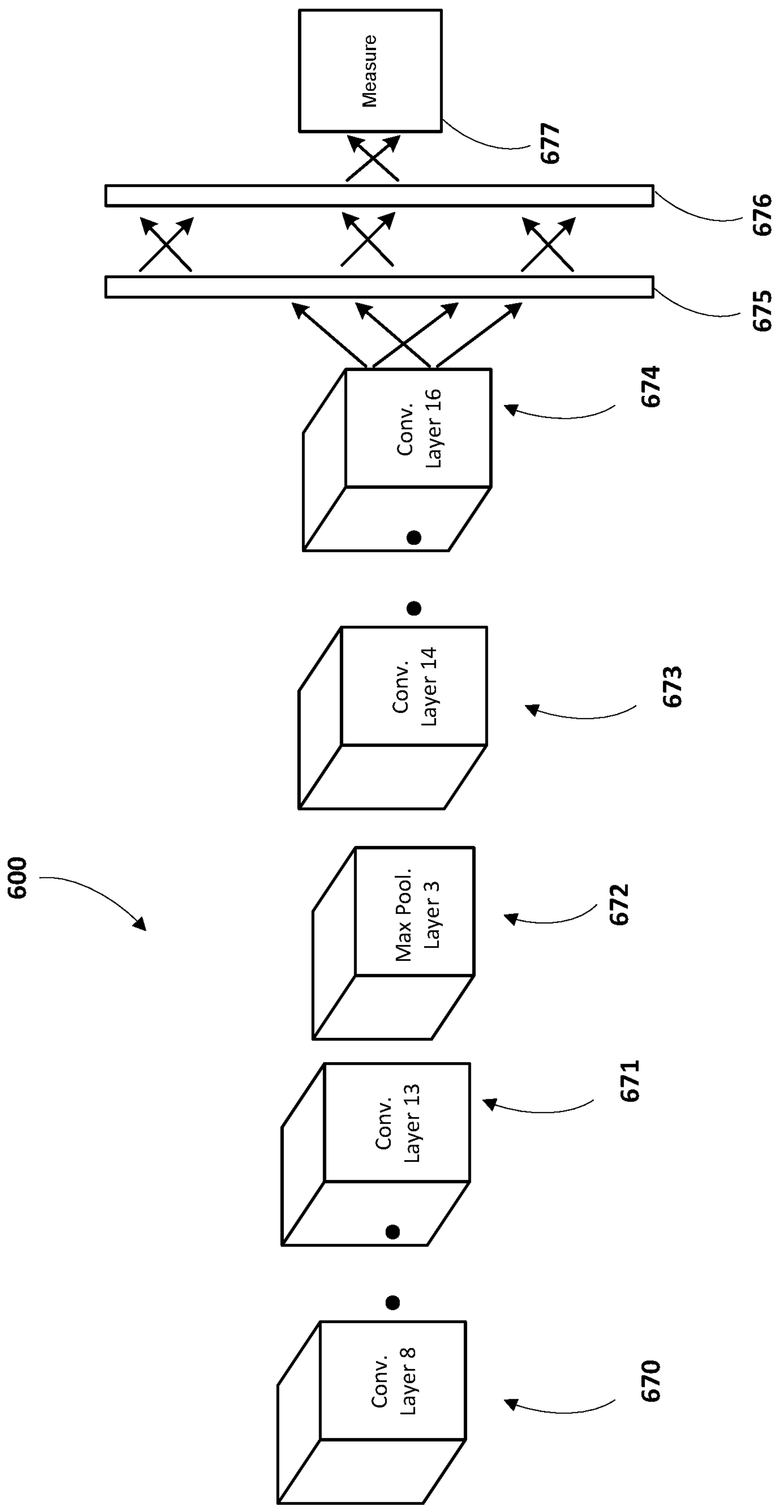

Measure
677
676
675
Conv. Layer 16
674
Conv. Layer 14
673
Max Pool. Layer 3
672
600
Conv. Layer 13
671
Conv. Layer 8
670

FIG. 6B

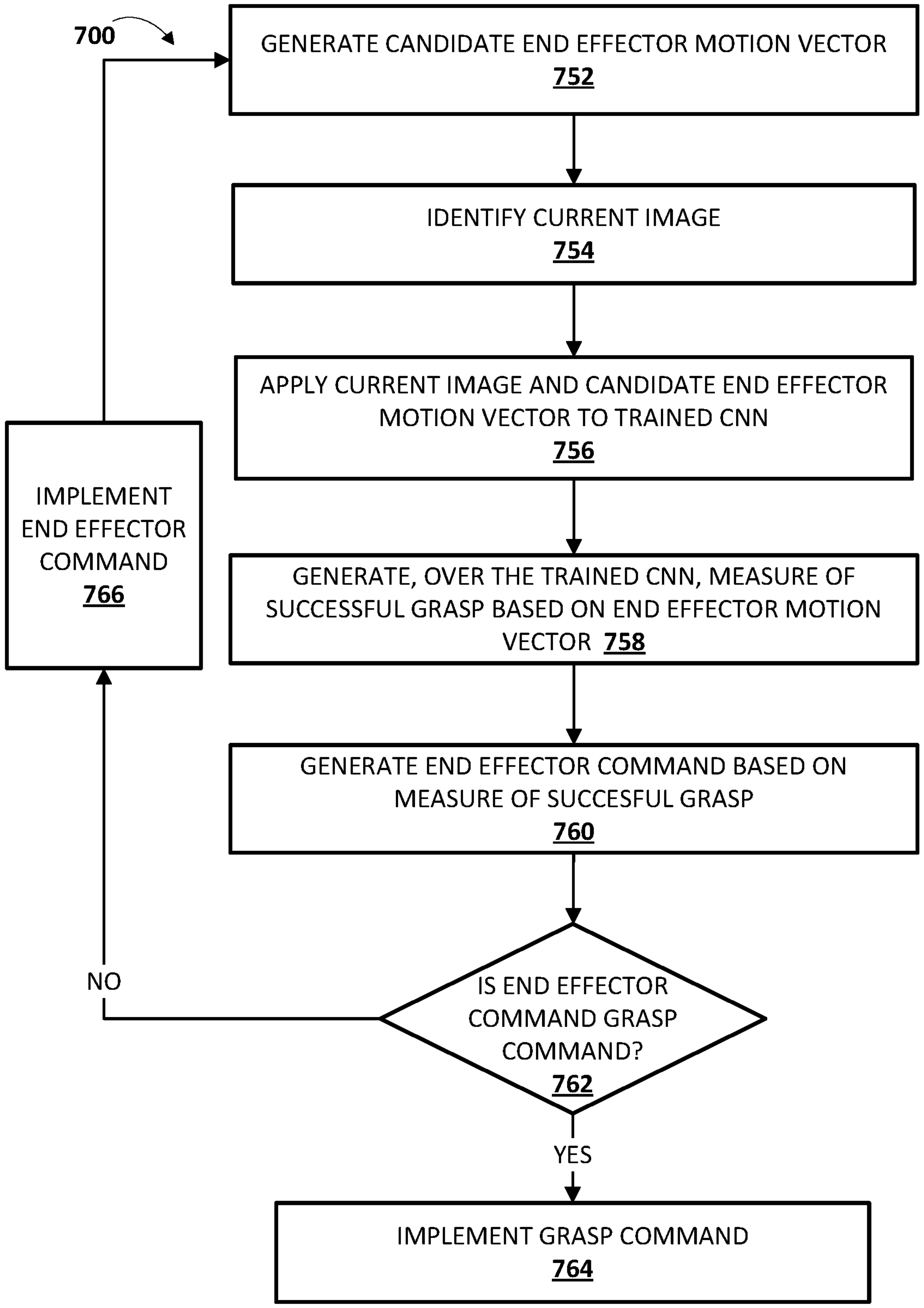

700
GENERATE CANDIDATE END EFFECTOR MOTION VECTOR 752
IDENTIFY CURRENT IMAGE 754
APPLY CURRENT IMAGE AND CANDIDATE END EFFECTOR MOTION VECTOR TO TRAINED CNN 756
GENERATE, OVER THE TRAINED CNN, MEASURE OF SUCCESSFUL GRASP BASED ON END EFFECTOR MOTION VECTOR 758
GENERATE END EFFECTOR COMMAND BASED ON MEASURE OF SUCCESFUL GRASP 760
IS END EFFECTOR COMMAND GRASP COMMAND? 762
NO
YES
IMPLEMENT END EFFECTOR COMMAND 766
IMPLEMENT GRASP COMMAND 764

FIG. 7A

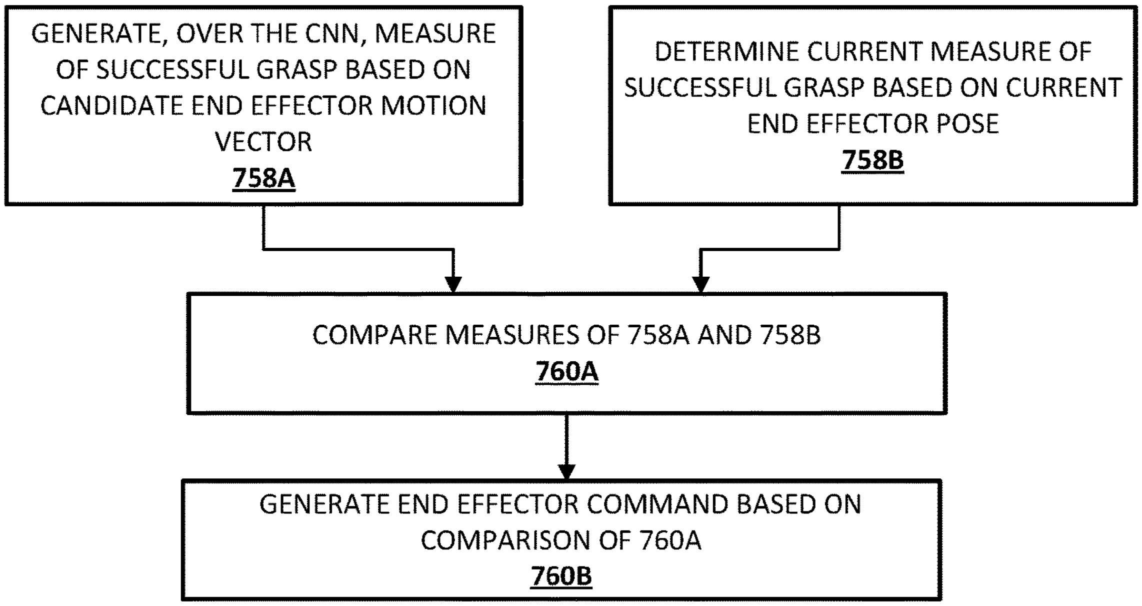

GENERATE, OVER THE CNN, MEASURE OF SUCCESSFUL GRASP BASED ON CANDIDATE END EFFECTOR MOTION VECTOR
758A
DETERMINE CURRENT MEASURE OF SUCCESSFUL GRASP BASED ON CURRENT END EFFECTOR POSE
758B
COMPARE MEASURES OF 758A AND 758B
760A
GENERATE END EFFECTOR COMMAND BASED ON COMPARISON OF 760A
760B

FIG. 7B

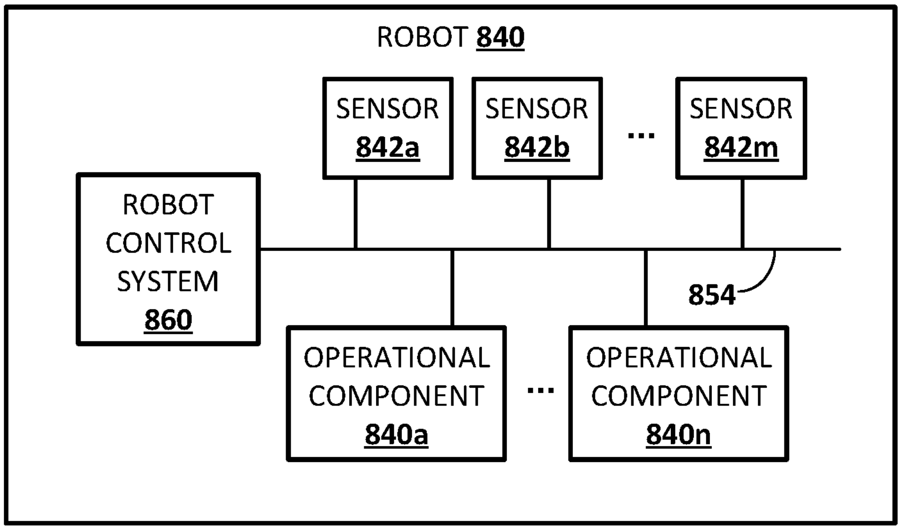

ROBOT 840
SENSOR 842a
SENSOR 842b
...
SENSOR 842m
ROBOT CONTROL SYSTEM 860
854
OPERATIONAL COMPONENT 840a
...
OPERATIONAL COMPONENT 840n

FIG. 8

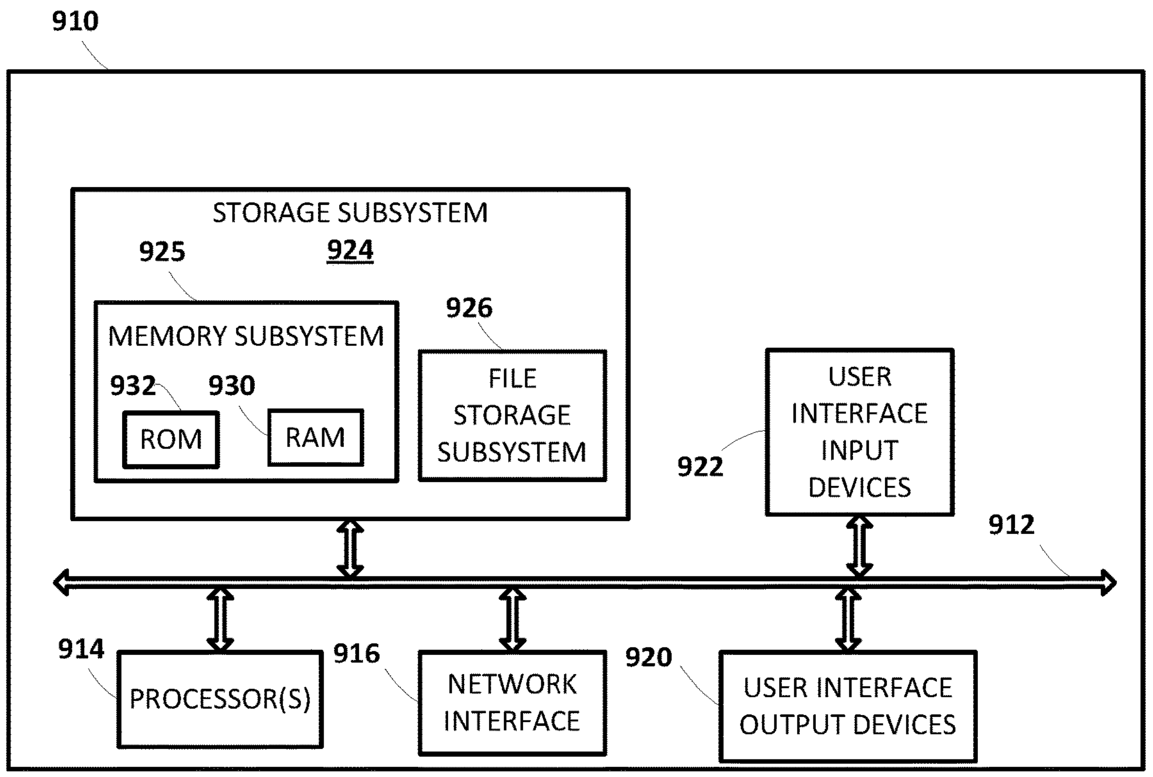

910
STORAGE SUBSYSTEM
924
925
MEMORY SUBSYSTEM
932
930
ROM
RAM
926
FILE STORAGE SUBSYSTEM
922
USER INTERFACE INPUT DEVICES
912
914
PROCESSOR(S)
916
NETWORK INTERFACE
920
USER INTERFACE OUTPUT DEVICES

FIG. 9

DEEP MACHINE LEARNING METHODS AND APPARATUS FOR ROBOTIC GRASPING

BACKGROUND

[0001] Many robots are programmed to utilize one or more end effectors to grasp one or more objects. For example, a robot may utilize a grasping end effector such as an "impactive" gripper or "ingressive" gripper (e.g., physically penetrating an object using pins, needles, etc.) to pick up an object from a first location, move the object to a second location, and drop off the object at the second location. Some additional examples of robot end effectors that may grasp objects include "astrictive" end effectors (e.g., using suction or vacuum to pick up an object) and one or more "contigutive" end effectors (e.g., using surface tension, freezing or adhesive to pick up an object), to name just a few.

SUMMARY

[0002] This specification is directed generally to deep machine learning methods and apparatus related to manipulation of an object by an end effector of a robot. Some implementations are directed to training a deep neural network, such as a convolutional neural network (also referred to herein as a "CNN"), to predict the probability that candidate motion data for an end effector of a robot will result in a successful grasp of one or more objects by the end effector. For example, some implementations enable applying, as input to a trained deep neural network, at least: (1) a candidate motion vector that defines a candidate motion (if any) of a grasping end effector of a robot and (2) an image that captures at least a portion of the work space of the robot; and generating, based on the applying, at least a measure that directly or indirectly indicates the probability that the candidate motion vector will result in a successful grasp. The predicted probability may then be used in servoing performance of grasp attempts by a robot having a grasping end effector, thereby improving the ability of the robot to successfully grasp objects in its environment.

[0003] Some implementations are directed to utilization of the trained deep neural network to servo a grasping end effector of a robot to achieve a successful grasp of an object by the grasping end effector. For example, the trained deep neural network may be utilized in the iterative updating of motion control commands for one or more actuators of a robot that control the pose of a grasping end effector of the robot, and to determine when to generate grasping control commands to effectuate an attempted grasp by the grasping end effector. In various implementations, utilization of the trained deep neural network to servo the grasping end effector may enable fast feedback to robotic perturbations and/or motion of environmental object(s) and/or robustness to inaccurate robotic actuation(s).

[0004] In some implementations, a method is provided that includes generating a candidate end effector motion vector that defines motion to move a grasping end effector of a robot from a current pose to an additional pose. The method further includes identifying a current image that is captured by a vision sensor associated with the robot and that captures the grasping end effector and at least one object in an environment of the robot. The method further includes applying the current image and the candidate end effector motion vector as input to a trained convolutional neural network and generating, over the trained convolutional neural network, a measure of successful grasp of the object with application of the motion. The measure is generated based on the application of the image and the end effector motion vector to the trained convolutional neural network. The method optionally further includes generating an end effector command based on the measure and providing the end effector command to one or more actuators of the robot. The end effector command may be a grasp command or an end effector motion command.

[0005] This method and other implementations of technology disclosed herein may each optionally include one or more of the following features.

[0006] In some implementations, the method further includes determining a current measure of successful grasp of the object without application of the motion, and generating the end effector command based on the measure and the current measure. In some versions of those implementations, the end effector command is the grasp command and generating the grasp command is in response to determining that comparison of the measure to the current measure satisfies a threshold. In some other versions of those implementations, the end effector command is the end effector motion command and generating the end effector motion command includes generating the end effector motion command to conform to the candidate end effector motion vector. In yet other versions of those implementations, the end effector command is the end effector motion command and generating the end effector motion command includes generating the end effector motion command to effectuate a trajectory correction to the end effector. In some implementations, determining the current measure of successful grasp of the object without application of the motion includes: applying the image and a null end effector motion vector as input to the trained convolutional neural network; and generating, over the trained convolutional neural network, the current measure of successful grasp of the object without application of the motion.

[0007] In some implementations, the end effector command is the end effector motion command and conforms to the candidate end effector motion vector. In some of those implementations, providing the end effector motion command to the one or more actuators moves the end effector to a new pose, and the method further includes: generating an additional candidate end effector motion vector defining new motion to move the grasping end effector from the new pose to a further additional pose; identifying a new image captured by a vision sensor associated with the robot, the new image capturing the end effector at the new pose and capturing the objects in the environment; applying the new image and the additional candidate end effector motion vector as input to the trained convolutional neural network; generating, over the trained convolutional neural network, a new measure of successful grasp of the object with application of the new motion, the new measure being generated based on the application of the new image and the additional end effector motion vector to the trained convolutional neural network; generating a new end effector command based on the new measure, the new end effector command being the grasp command or a new end effector motion command; and providing the new end effector command to one or more actuators of the robot.

[0008] In some implementations, applying the image and the candidate end effector motion vector as input to the trained convolutional neural network includes: applying the image as input to an initial layer of the trained convolutional neural network; and applying the candidate end effector motion vector to an additional layer of the trained convolutional neural network. The additional layer may be downstream of the initial layer. In some of those implementations, applying the candidate end effector motion vector to the additional layer includes: passing the end effector motion vector through a fully connected layer of the convolutional neural network to generate end effector motion vector output; and concatenating the end effector motion vector output with upstream output. The upstream output is from an immediately upstream layer of the convolutional neural network that is immediately upstream of the additional layer and that is downstream from the initial layer and from one or more intermediary layers of the convolutional neural network. The initial layer may be a convolutional layer and the immediately upstream layer may be a pooling layer.

[0009] In some implementations, the method further includes identifying an additional image captured by the vision sensor and applying the additional image as additional input to the trained convolutional neural network. The additional image may capture the one or more environmental objects and omit the robotic end effector or include the robotic end effector in a different pose than that of the robotic end effector in the image. In some of those implementations, applying the image and the additional image to the convolutional neural network includes concatenating the image and the additional image to generate a concatenated image, and applying the concatenated image as input to an initial layer of the convolutional neural network.

[0010] In some implementations, generating the candidate end effector motion vector includes generating a plurality of candidate end effector motion vectors and performing one or more iterations of cross-entropy optimization on the plurality of candidate end effector motion vectors to select the candidate end effector motion vector from the plurality of candidate end effector motion vectors.

[0011] In some implementations, a method is provided that includes identifying a plurality of training examples generated based on sensor output from one or more robots during a plurality of grasp attempts by the robots. Each of the training examples including training example input and training example output. The training example input of each of the training examples includes: an image for a corresponding instance of time of a corresponding grasp attempt of the grasp attempts, the image capturing a robotic end effector and one or more environmental objects at the corresponding instance of time; and an end effector motion vector defining motion of the end effector to move from an instance of time pose of the end effector at the corresponding instance of time to a final pose of the end effector for the corresponding grasp attempt. The training example output of each of the training examples includes a grasp success label indicative of success of the corresponding grasp attempt. The method further includes training the convolutional neural network based on the training examples.

[0012] This method and other implementations of technology disclosed herein may each optionally include one or more of the following features.

[0013] In some implementations, the training example input of each of the training examples further includes an additional image for the corresponding grasp attempt. The additional image may capture the one or more environmental objects and omit the robotic end effector or include the robotic end effector in a different pose than that of the robotic end effector in the image. In some implementations, training the convolutional neural network includes applying, to the convolutional neural network, the training example input of a given training example of the training examples. In some of those implementations, applying the training example input of the given training example includes: concatenating the image and the additional image of the given training example to generate a concatenated image; and applying the concatenated image as input to an initial layer of the convolutional neural network.

[0014] In some implementations, training the convolutional neural network includes applying, to the convolutional neural network, the training example input of a given training example of the training examples. In some of those implementations, applying the training example input of the given training example includes: applying the image of the given training example as input to an initial layer of the convolutional neural network; and applying the end effector motion vector of the given training example to an additional layer of the convolutional neural network. The additional layer may be downstream of the initial layer. In some of those implementations, applying the end effector motion vector to the additional layer includes: passing the end effector motion vector through a fully connected layer to generate end effector motion vector output and concatenating the end effector motion vector output with upstream output. The upstream output may be from an immediately upstream layer of the convolutional neural network that is immediately upstream of the additional layer and that is downstream from the initial layer and from one or more intermediary layers of the convolutional neural network. The initial layer may be a convolutional layer and the immediately upstream layer may be a pooling layer.

[0015] In some implementations, the end effector motion vector defines motion of the end effector in task-space.

[0016] In some implementations, the training examples include: a first group of the training examples generated based on output from a plurality of first robot sensors of a first robot during a plurality of the grasp attempts by the first robot; and a second group of the training examples generated based on output from a plurality of second robot sensors of a second robot during a plurality of the grasp attempts by the second robot. In some of those implementations: the first robot sensors include a first vision sensor generating the images for the training examples of the first group; the second robot sensors include a second vision sensor generating the images for the training examples of the second group; and a first pose of the first vision sensor relative to a first base of the first robot is distinct from a second pose of the second vision sensor relative to a second base of the second robot.

[0017] In some implementations, the grasp attempts on which a plurality of training examples are based each include a plurality of random actuator commands that randomly move the end effector from a starting pose of the end effector to the final pose of the end effector, then grasp with the end effector at the final pose. In some of those implementations, the method further includes: generating additional grasp attempts based on the trained convolutional neural network; identifying a plurality of additional training examples based on the additional grasp attempts; and updating the convolutional neural network by further training of the convolutional network based on the additional training examples.

[0018] In some implementations, the grasp success label for each of the training examples is either a first value indicative of success or a second value indicative of failure.

[0019] In some implementations, the training comprises performing backpropagation on the convolutional neural network based on the training example output of the plurality of training examples.

[0020] Other implementations may include a non-transitory computer readable storage medium storing instructions executable by a processor (e.g., a central processing unit (CPU) or graphics processing unit (GPU)) to perform a method such as one or more of the methods described above. Yet another implementation may include a system of one or more computers and/or one or more robots that include one or more processors operable to execute stored instructions to perform a method such as one or more of the methods described above.

[0021] It should be appreciated that all combinations of the foregoing concepts and additional concepts described in greater detail herein are contemplated as being part of the subject matter disclosed herein. For example, all combinations of claimed subject matter appearing at the end of this disclosure are contemplated as being part of the subject matter disclosed herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] FIG. **1** illustrates an example environment in which grasp attempts may be performed by robots, data associated with the grasp attempts may be utilized to generate training examples, and/or the training examples may be utilized to train a convolutional neural network.

[0023] FIG. **2** illustrates one of the robots of FIG. **1** and an example of movement of a grasping end effector of the robot along a path.

[0024] FIG. **3** is a flowchart illustrating an example method of performing grasp attempts and storing data associated with the grasp attempts.

[0025] FIG. **4** is a flowchart illustrating an example method of generating training examples based on data associated with grasp attempts of robots.

[0026] FIG. **5** is a flow chart illustrating an example method of training a convolutional neural network based on training examples.

[0027] FIGS. **6**A and **6**B illustrate an architecture of an example convolutional neural network.

[0028] FIG. **7**A is a flowchart illustrating an example method of utilizing a trained convolutional neural network to servo a grasping end effector.

[0029] FIG. **7**B is a flowchart illustrating some implementations of certain blocks of the flowchart of FIG. **7**A.

[0030] FIG. **8** schematically depicts an example architecture of a robot.

[0031] FIG. **9** schematically depicts an example architecture of a computer system.

DETAILED DESCRIPTION

[0032] Some implementations of the technology described herein are directed to training a deep neural network, such as a CNN, to enable utilization of the trained deep neural network to predict a measure indicating the probability that candidate motion data for a grasping end effector of a robot will result in a successful grasp of one or more objects by the end effector. In some implementations, the trained deep neural network accepts an image (I_b) generated by a vision sensor and accepts an end effector motion vector (v_t), such as a task-space motion vector. The application of the image (I_b) and the end effector motion vector (v_t) to the trained deep neural network may be used to generate, over the deep neural network, a predicted measure that executing command(s) to implement the motion defined by motion vector (v_t), and subsequently grasping, will produce a successful grasp. Some implementations are directed to utilization of the trained deep neural network to servo a grasping end effector of a robot to achieve a successful grasp of an object by the grasping end effector. Additional description of these and other implementations of the technology is provided below.

[0033] With reference to FIGS. **1**-**6**B, various implementations of training a CNN are described. FIG. **1** illustrates an example environment in which grasp attempts may be performed by robots (e.g., robots **180**A, **180**B, and/or other robots), data associated with the grasp attempts may be utilized to generate training examples, and/or the training examples may be utilized to train a CNN.

[0034] Example robots **180**A and **180**B are illustrated in FIG. **1**. Robots **180**A and **180**B are "robot arms" having multiple degrees of freedom to enable traversal of grasping end effectors **182**A and **182**B along any of a plurality of potential paths to position the grasping end effectors **182**A and **182**B in desired locations. For example, with reference to FIG. **2**, an example of robot **180**A traversing its end effector along a path **201** is illustrated. FIG. **2** includes a phantom and non-phantom image of the robot **180**A showing two different poses of a set of poses struck by the robot **180**A and its end effector in traversing along the path **201**. Referring again to FIG. **1**, robots **180**A and **180**B each further controls the two opposed "claws" of their corresponding grasping end effector **182**A, **182**B to actuate the claws between at least an open position and a closed position (and/or optionally a plurality of "partially closed" positions).

[0035] Example vision sensors **184**A and **184**B are also illustrated in FIG. **1**. In FIG. **1**, vision sensor **184**A is mounted at a fixed pose relative to the base or other stationary reference point of robot **180**A. Vision sensor **184**B is also mounted at a fixed pose relative to the base or other stationary reference point of robot **1806**. As illustrated in FIG. **1**, the pose of the vision sensor **184**A relative to the robot **180**A is different than the pose of the vision sensor **184**B relative to the robot **1806**. As described herein, in some implementations this may be beneficial to enable generation of varied training examples that can be utilized to train a neural network that is robust to and/or independent of camera calibration. Vision sensors **184**A and **184**B are sensors that can generate images related to shape, color, depth, and/or other features of object(s) that are in the line of sight of the sensors. The vision sensors **184**A and **184**B may be, for example, monographic cameras, stereographic cameras, and/or 3D laser scanner. A 3D laser scanner includes one or more lasers that emit light and one or more sensors that collect data related to reflections of the emitted light. A 3D laser scanner may be, for example, a time-of-flight 3D laser scanner or a triangulation based 3D laser scanner and may include a position sensitive detector (PSD) or other optical position sensor.

[0036] The vision sensor **184**A has a field of view of at least a portion of the workspace of the robot **180**A, such as the portion of the workspace that includes example objects **191**A. Although resting surface(s) for objects **191**A are not illustrated in FIG. **1**, those objects may rest on a table, a tray, and/or other surface(s). Objects **191**A include a spatula, a stapler, and a pencil. In other implementations more objects, fewer objects, additional objects, and/or alternative objects may be provided during all or portions of grasp attempts of robot **180**A as described herein. The vision sensor **184**B has a field of view of at least a portion of the workspace of the robot **1806**, such as the portion of the workspace that includes example objects **191**B. Although resting surface(s) for objects **191**B are not illustrated in FIG. **1**, they may rest on a table, a tray, and/or other surface(s). Objects **191**B include a pencil, a stapler, and glasses. In other implementations more objects, fewer objects, additional objects, and/or alternative objects may be provided during all or portions of grasp attempts of robot **1806** as described herein.

[0037] Although particular robots **180**A and **180**B are illustrated in FIG. **1**, additional and/or alternative robots may be utilized, including additional robot arms that are similar to robots **180**A and **180**B, robots having other robot arm forms, robots having a humanoid form, robots having an animal form, robots that move via one or more wheels (e.g., self-balancing robots), submersible vehicle robots, an unmanned aerial vehicle ("UAV"), and so forth. Also, although particular grasping end effectors are illustrated in FIG. **1**, additional and/or alternative end effectors may be utilized, such as alternative impactive grasping end effectors (e.g., those with grasping "plates", those with more or fewer "digits"/"claws"), "ingressive" grasping end effectors, "astrictive" grasping end effectors, or "contigutive" grasping end effectors, or non-grasping end effectors. Additionally, although particular mountings of vision sensors **184**A and **184**B are illustrated in FIG. **1**, additional and/or alternative mountings may be utilized. For example, in some implementations, vision sensors may be mounted directly to robots, such as on non-actuable components of the robots or on actuable components of the robots (e.g., on the end effector or on a component close to the end effector). Also, for example, in some implementations, a vision sensor may be mounted on a non-stationary structure that is separate from its associated robot and/or may be mounted in a non-stationary manner on a structure that is separate from its associated robot.

[0038] Robots **180**A, **180**B, and/or other robots may be utilized to perform a large quantity of grasp attempts and data associated with the grasp attempts may be utilized by the training example generation system **110** to generate training examples. In some implementations, all or aspects of training example generation system **110** may be implemented on robot **180**A and/or robot **180**B (e.g., via one or more processors of robots **180**A and **180**B). For example, robots **180**A and **180**B may each include an instance of the training example generation system **110**. In some implementations, all or aspects of training example generation system **110** may be implemented on one or more computer systems that are separate from, but in network communication with, robots **180**A and **1808**.

[0039] Each grasp attempt by robot **180**A, **180**B, and/or other robots consists of T separate time steps or instances. At each time step, a current image (I_t^i) captured by the vision sensor of the robot performing the grasp attempt is stored, the current pose (p_t^i) of the end effector is also stored, and the robot chooses a path (translational and/or rotational) along which to next move the gripper. At the final time step T, the robot actuates (e.g., closes) the gripper and stores additional data and/or performs one or more additional actions to enable evaluation of the success of the grasp. The grasp success engine **116** of training example generation system **110** evaluates the success of the grasp, generating a grasp success label (ℓ_i).

[0040] Each grasp attempt results in T training examples, represented by (I_t^i,$p_T^i-p_t^i$,ℓ_i). That is, each training example includes at least the image observed at that time step (I_t^i), the end effector motion vector ($p_T^i-p_t^i$) from the pose at that time step to the one that is eventually reached (the final pose of the grasp attempt), and the grasp success label (ℓ_i) of the grasp attempt. Each end effector motion vector may be determined by the end effector motion vector engine **114** of training example generation system **110**. For example, the end effector motion vector engine **114** may determine a transformation between the current pose and the final pose of the grasp attempt and use the transformation as the end effector motion vector. The training examples for the plurality of grasp attempts of a plurality of robots are stored by the training example generation system **110** in training examples database **117**.

[0041] The data generated by sensor(s) associated with a robot and/or the data derived from the generated data may be stored in one or more non-transitory computer readable media local to the robot and/or remote from the robot. In some implementations, the current image may include multiple channels, such as a red channel, a blue channel, a green channel, and/or a depth channel. Each channel of an image defines a value for each of a plurality of pixels of the image such as a value from 0 to 255 for each of the pixels of the image. In some implementations, each of the training examples may include the current image and an additional image for the corresponding grasp attempt, where the additional image does not include the grasping end effector or includes the end effector in a different pose (e.g., one that does not overlap with the pose of the current image). For instance, the additional image may be captured after any preceding grasp attempt, but before end effector movement for the grasp attempt begins and when the grasping end effector is moved out of the field of view of the vision sensor. The current pose and the end effector motion vector from the current pose to the final pose of the grasp attempt may be represented in task-space, in joint-space, or in another space. For example, the end effector motion vector may be represented by five values in task-space: three values defining the three-dimensional (3D) translation vector, and two values representing a sine-cosine encoding of the change in orientation of the end effector about an axis of the end effector. In some implementations, the grasp success label is a binary label, such as a "0/successful" or "1/not successful" label. In some implementations, the grasp success label may be selected from more than two options, such as 0, 1, and one or more values between 0 and 1. For example, "0" may indicate a confirmed "not successful grasp", "1" may indicate a confirmed successful grasp, "0.25" may indicate a "most likely not successful grasp" and "0.75" may indicate a "most likely successful grasp."

[0042] The training engine **120** trains a CNN **125**, or other neural network, based on the training examples of training examples database **117**. Training the CNN **125** may include iteratively updating the CNN **125** based on application of the training examples to the CNN **125**. For example, the current image, the additional image, and the vector from the current pose to the final pose of the grasp attempt of the training examples may be utilized as training example input; and the grasp success label may be utilized as training example output. The trained CNN **125** is trained to predict a measure indicating the probability that, in view of current image (and optionally an additional image, such as one that at least partially omits the end effector), moving a gripper in accordance with a given end effector motion vector, and subsequently grasping, will produce a successful grasp.

[0043] FIG. **3** is a flowchart illustrating an example method **300** of performing grasp attempts and storing data associated with the grasp attempts. For convenience, the operations of the flow chart are described with reference to a system that performs the operations. This system may include one or more components of a robot, such as a processor and/or robot control system of robot **180**A, **180**B, **840**, and/or other robot. Moreover, while operations of method **300** are shown in a particular order, this is not meant to be limiting. One or more operations may be reordered, omitted or added.

[0044] At block **352**, the system starts a grasp attempt. At block **354**, the system stores an image of an environment without an end effector present in the image. For example, the system may move the grasping end effector out of the field of view of the vision sensor (i.e., not occluding the view of the environment) and capture an image at an instance when the grasping end effector is out of the field of view. The image may then be stored and associated with the grasp attempt.

[0045] At block **356**, the system determines and implements an end effector movement. For example, the system may generate one or more motion commands to cause one or more of the actuators that control the pose of the end effector to actuate, thereby changing the pose of the end effector.

[0046] In some implementations and/or iterations of block **356**, the motion command(s) may be random within a given space, such as the work-space reachable by the end effector, a restricted space within which the end effector is confined for the grasp attempts, and/or a space defined by position and/or torque limits of actuator(s) that control the pose of the end effector. For example, before initial training of a neural network is completed, the motion command(s) generated by the system at block **356** to implement end effector movement may be random within a given space. Random as used herein may include truly random or pseudo-random.

[0047] In some implementations, the motion command(s) generated by the system at block **356** to implement end effector movement may be based at least in part on a current version of a trained neural network and/or based on other criteria. In some implementations, in the first iteration of block **356** for each grasp attempt, the end effector may be "out of position" based on it being moved out of the field of view at block **354**. In some of those implementations, prior to the first iteration of block **356** the end effector may be randomly or otherwise moved "back into position". For example, the end effector may be moved back to a set "starting position" and/or moved to a randomly selected position within a given space.

[0048] At block **358**, the system stores: (1) an image that captures the end effector and the environment at the current instance of the grasp attempt and (2) the pose of the end effector at the current instance. For example, the system may store a current image generated by a vision sensor associated with the robot and associate the image with the current instance (e.g., with a timestamp). Also, for example the system may determine the current pose of the end effector based on data from one or more joint position sensors of joints of the robot whose positions affect the pose of the robot, and the system may store that pose. The system may determine and store the pose of the end effector in task-space, joint-space, or another space.

[0049] At block **360**, the system determines whether the current instance is the final instance for the grasping attempt. In some implementations, the system may increment an instance counter at block **352**, **354**, **356**, or **358** and/or increment a temporal counter as time passes—and determine if the current instance is the final instance based on comparing a value of the counter to a threshold. For example, the counter may be a temporal counter and the threshold may be 3 seconds, 4 seconds, 5 seconds, and/or other value. In some implementations, the threshold may vary between one or more iterations of the method **300**.

[0050] If the system determines at block **360** that the current instance is not the final instance for the grasping attempt, the system returns to blocks **356**, where it determines and implements another end effector movement, then proceeds to block **358** where it stores an image and the pose at the current instance. Through multiple iterations of blocks **356**, **358**, and **360** for a given grasp attempt, the pose of the end effector will be altered by multiple iterations of block **356**, and an image and the pose stored at each of those instances. In many implementations, blocks **356**, **358**, **360**, and/or other blocks may be performed at a relatively high frequency, thereby storing a relatively large quantity of data for each grasp attempt.

[0051] If the system determines at block **360** that the current instance is the final instance for the grasping attempt, the system proceeds to block **362**, where it actuates the gripper of the end effector. For example, for an impactive gripper end effector, the system may cause one or more plates, digits, and/or other members to close. For instance, the system may cause the members to close until they are either at a fully closed position or a torque reading measured by torque sensor(s) associated with the members satisfies a threshold.

[0052] At block **364**, the system stores additional data and optionally performs one or more additional actions to enable determination of the success of the grasp of block **360**. In some implementations, the additional data is a position reading, a torque reading, and/or other reading from the gripping end effector. For example, a position reading that is greater than some threshold (e.g., 1 cm) may indicate a successful grasp.

[0053] In some implementations, at block **364** the system additionally and/or alternatively: (1) maintains the end effector in the actuated (e.g., closed) position and moves (e.g., vertically and/or laterally) the end effector and any object that may be grasped by the end effector; (2) stores an image that captures the original grasping position after the end effector is moved; (3) causes the end effector to "drop" any object that is being grasped by the end effector (optionally after moving the gripper back close to the original grasping position); and (4) stores an image that captures the original grasping position after the object (if any) has been dropped. The system may store the image that captures the original grasping position after the end effector and the object (if any) is moved and store the image that captures the original grasping position after the object (if any) has been dropped—and associate the images with the grasp attempt. Comparing the image after the end effector and the object (if any) is moved to the image after the object (if any) has been dropped, may indicate whether a grasp was successful. For example, an object that appears in one image but not the other may indicate a successful grasp.

[0054] At block **366**, the system resets the counter (e.g., the instance counter and/or the temporal counter), and proceeds back to block **352** to start another grasp attempt.

[0055] In some implementations, the method **300** of FIG. **3** may be implemented on each of a plurality of robots, optionally operating in parallel during one or more (e.g., all) of their respective iterations of method **300**. This may enable more grasp attempts to be achieved in a given time period than if only one robot was operating the method **300**. Moreover, in implementations where one or more of the plurality of robots includes an associated vision sensor with a pose relative to the robot that is unique from the pose of one or more vision sensors associated with other of the robots, training examples generated based on grasp attempts from the plurality of robots may provide robustness to vision sensor pose in a neural network trained based on those training examples. Moreover, in implementations where gripping end effectors and/or other hardware components of the plurality of robots vary and/or wear differently, and/or in which different robots (e.g., same make and/or model and/or different make(s) and/or model(s)) interact with different objects (e.g., objects of different sizes, different weights, different shapes, different translucencies, different materials) and/or in different environments (e.g., different surfaces, different lighting, different environmental obstacles), training examples generated based on grasp attempts from the plurality of robots may provide robustness to various robotic and/or environmental configurations.

[0056] In some implementations, the objects that are reachable by a given robot and on which grasp attempts may be made may be different during different iterations of the method **300**. For example, a human operator and/or another robot may add and/or remove objects to the workspace of a robot between one or more grasp attempts of the robot. Also, for example, the robot itself may drop one or more objects out of its workspace following successful grasps of those objects. This may increase the diversity of the training data. In some implementations, environmental factors such as lighting, surface(s), obstacles, etc. may additionally and/or alternatively be different during different iterations of the method **300**, which may also increase the diversity of the training data.

[0057] FIG. **4** is a flowchart illustrating an example method **400** of generating training examples based on data associated with grasp attempts of robots. For convenience, the operations of the flow chart are described with reference to a system that performs the operations. This system may include one or more components of a robot and/or another computer system, such as a processor and/or robot control system of robot **180**A, **1806**, **1220**, and/or a processor of training example generation system **110** and/or other system that may optionally be implemented separate from a robot. Moreover, while operations of method **400** are shown in a particular order, this is not meant to be limiting. One or more operations may be reordered, omitted or added.

[0058] At block **452**, the system starts training example generation. At block **454**, the system selects a grasp attempt. For example, the system may access a database that includes data associated with a plurality of stored grasp attempts, and select one of the stored grasp attempts. The selected grasp attempt may be, for example, a grasp attempt generated based on the method **300** of FIG. **3**.

[0059] At block **456**, the system determines a grasp success label for the selected grasp attempt based on stored data for the selected grasp attempt. For example, as described with respect to block **364** of method **300**, additional data may be stored for the grasp attempt to enable determination of a grasp success label for the grasp attempt. The stored data may include data from one or more sensors, where the data is generated during and/or after the grasp attempt.

[0060] As one example, the additional data may be a position reading, a torque reading, and/or other reading from the gripping end effector. In such an example, the system may determine a grasp success label based on the reading(s). For example, where the reading is a position reading, the system may determine a "successful grasp" label if the reading is greater than some threshold (e.g., 1 cm)—and may determine an "unsuccessful grasp" label if the reading is less than some threshold (e.g., 1 cm).

[0061] As another example, the additional data may be an image that captures the original grasping position after the end effector and the object (if any) is moved and an image that captures the original grasping position after the object (if any) has been dropped. To determine the grasp success label, the system may compare (1) the image after the end effector and the object (if any) is moved to (2) the image after the object (if any) has been dropped. For example, the system may compare pixels of the two images and, if more than a threshold number of pixels between the two images are different, then the system may determine a "successful grasp" label. Also, for example, the system may perform object detection in each of the two images, and determine a "successful grasp" label if an object is detected in the image captured after the object (if any) has been dropped, but is not detected in the image captured after the end effector and the object (if any) is moved.

[0062] As yet another example, the additional data may be an image that captures the original grasping position after the end effector and the object (if any) is moved. To determine the grasp success label, the system may compare (1) the image after the end effector and the object (if any) is moved to (2) an additional image of the environment taken before the grasp attempt began (e.g., an additional image that omits the end effector).

[0063] In some implementations, the grasp success label is a binary label, such as a "successful"/"not successful" label. In some implementations, the grasp success label may be selected from more than two options, such as 0, 1, and one or more values between 0 and 1. For example, in a pixel comparison approach, "0" may indicate a confirmed "not successful grasp" and may be selected by the system when less than a first threshold number of pixels is different between the two images; "0.25" may indicate a "most likely not successful grasp" and may be selected when the number of different pixels is from the first threshold to a greater second threshold, "0.75" may indicate a "most likely successful grasp" and may be selected when the number of different pixels is greater than the second threshold (or other threshold), but less than a third threshold; and "1" may indicate a "confirmed successful grasp", and may be selected when the number of different pixels is equal to or greater than the third threshold.

[0064] At block **458**, the system selects an instance for the grasp attempt. For example, the system may select data associated with the instance based on a timestamp and/or other demarcation associated with the data that differentiates it from other instances of the grasp attempt.

[0065] At block **460**, the system generates an end effector motion vector for the instance based on the pose of the end effector at the instance and the pose of the end effector at the final instance of the grasp attempt. For example, the system may determine a transformation between the current pose and the final pose of the grasp attempt and use the transformation as the end effector motion vector. The current pose and the end effector motion vector from the current pose to the final pose of the grasp attempt may be represented in task-space, in joint-space, or in another space. For example, the end effector motion vector may be represented by five values in task-space: three values defining the three-dimensional (3D) translation vector, and two values representing a sine-cosine encoding of the change in orientation of the end effector about an axis of the end effector.

[0066] At block **462**, the system generates a training example for the instance that includes: (1) the stored image for the instance, (2) the end effector motion vector generated for the instance at block **460**, and (3) the grasp success label determined at block **456**. In some implementations, the system generates a training example that also includes a stored additional image for the grasping attempt, such as one that at least partially omits the end effector and that was captured before the grasp attempt. In some of those implementations, the system concatenates the stored image for the instance and the stored additional image for the grasping attempt to generate a concatenated image for the training example. The concatenated image includes both the stored image for the instance and the stored additional image. For example, where both images include X by Y pixels and three channels (e.g., red, blue, green), the concatenated image may include X by Y pixels and six channels (three from each image). As described herein, the current image, the additional image, and the vector from the current pose to the final pose of the grasp attempt of the training examples may be utilized as training example input; and the grasp success label may be utilized as training example output.

[0067] In some implementations, at block **462** the system may optionally process the image(s). For example, the system may optionally resize the image to fit a defined size of an input layer of the CNN, remove one or more channels from the image, and/or normalize the values for depth channel(s) (in implementations where the images include a depth channel).

[0068] At block **464**, the system determines whether the selected instance is the final instance of the grasp attempt. If the system determines the selected instance is not the final instance of the grasp attempt, the system returns to block **458** and selects another instance.

[0069] If the system determines the selected instance is the final instance of the grasp attempt, the system proceeds to block **466** and determines whether there are additional grasp attempts to process. If the system determines there are additional grasp attempts to process, the system returns to block **454** and selects another grasp attempt. In some implementations, determining whether there are additional grasp attempts to process may include determining whether there are any remaining unprocessed grasp attempts. In some implementations, determining whether there are additional grasp attempts to process may additionally and/or alternatively include determining whether a threshold number of training examples has already been generated and/or other criteria has been satisfied.

[0070] If the system determines there are not additional grasp attempts to process, the system proceeds to block **466** and the method **400** ends. Another iteration of method **400** may be performed again. For example, the method **400** may be performed again in response to at least a threshold number of additional grasp attempts being performed.

[0071] Although method **300** and method **400** are illustrated in separate figures herein for the sake of clarity, it is understood that one or more blocks of method **400** may be performed by the same component(s) that perform one or more blocks of the method **300**. For example, one or more (e.g., all) of the blocks of method **300** and the method **400** may be performed by processor(s) of a robot. Also, it is understood that one or more blocks of method **400** may be performed in combination with, or preceding or following, one or more blocks of method **300**.

[0072] FIG. **5** is a flowchart illustrating an example method **500** of training a convolutional neural network based on training examples. For convenience, the operations of the flow chart are described with reference to a system that performs the operations. This system may include one or more components of a computer system, such as a processor (e.g., a GPU) of training engine **120** and/or other computer system operating over the convolutional neural network (e.g., CNN **125**). Moreover, while operations of method **500** are shown in a particular order, this is not meant to be limiting. One or more operations may be reordered, omitted or added.

[0073] At block **552**, the system starts training. At block **554**, the system selects a training example. For example, the system may select a training example generated based on the method **400** of FIG. **4**.

[0074] At block **556**, the system applies an image for the instance of the training example and an additional image of the selected training example to an initial layer of a CNN. For example, the system may apply the images to an initial convolutional layer of the CNN. As described herein, the additional image may at least partially omit the end effector. In some implementations, the system concatenates the image and the additional image and applies the concatenated image to the initial layer. In some other implementations, the image and the additional image are already concatenated in the training example.

[0075] At block **558**, the system applies the end effector motion vector of the selected training example to an additional layer of the CNN. For example, the system may apply the end effector motion vector to an additional layer of the CNN that is downstream of the initial layer to which the images are applied at block **556**. In some implementations, to apply the end effector motion vector to the additional layer, the system passes the end effector motion vector through a fully connected layer to generate end effector motion vector output, and concatenates the end effector motion vector output with output from an immediately upstream layer of the CNN. The immediately upstream layer is immediately upstream of the additional layer to which the end effector motion vector is applied and may optionally be one or more layers downstream from the initial layer to which the images are applied at block **556**. In some implementations, the initial layer is a convolutional layer, the immediately upstream layer is a pooling layer, and the additional layer is a convolutional layer.

[0076] At block **560**, the system performs backpropogation on the CNN based on the grasp success label of the training example. At block **562**, the system determines whether there are additional training examples. If the system determines there are additional training examples, the system returns to block **554** and selects another training example. In some implementations, determining whether there are additional training examples may include determining whether there are any remaining training examples that have not been utilized to train the CNN. In some implementations, determining whether there are additional training examples may additionally and/or alternatively include determining whether a threshold number of training examples have been utilized and/or other criteria has been satisfied.

[0077] If the system determines there are not additional training examples and/or that some other criteria has been met, the system proceeds to block **564** or block **566**.

[0078] At block **564**, the training of the CNN may end. The trained CNN may then be provided for use by one or more robots in servoing a grasping end effector to achieve a successful grasp of an object by the grasping end effector. For example, a robot may utilize the trained CNN in performing the method **700** of FIG. **7**A.

[0079] At block **566**, the system may additionally and/or alternatively provide the trained CNN to generate additional training examples based on the trained CNN. For example, one or more robots may utilize the trained CNN in performing grasp attempts and data from those grasp attempts utilized to generate additional training examples. For instance, one or more robots may utilize the trained CNN in performing grasp attempts based on the method **700** of FIG. **7**A and data from those grasp attempts utilized to generate additional training examples based on the method **400** of FIG. **4**. The robots whose data is utilized to generate additional training examples may be robots in a laboratory/training set up and/or robots in actual use by one or more consumers.

[0080] At block **568**, the system may update the CNN based on the additional training examples generated in response to providing the trained CNN at block **566**. For example, the system may update the CNN by performing additional iterations of blocks **554**, **556**, **558**, and **560** based on additional training examples.

[0081] As indicated by the arrow extending between blocks **566** and **568**, the updated CNN may be provided again at block **566** to generate further training examples and those training examples utilized at block **568** to further update the CNN. In some implementations, grasp attempts performed in association with future iterations of block **566** may be temporally longer grasp attempts than those performed in future iterations and/or those performed without utilization of a trained CNN. For example, implementations of method **300** of FIG. **3** that are performed without utilization of a trained CNN may have the temporally shortest grasp attempts, those performed with an initially trained CNN may have temporally longer grasp attempts, those performed with the next iteration of a trained CNN yet temporally longer grasp attempts, etc. This may optionally be implemented via the optional instance counter and/or temporal counter of method **300**.

[0082] FIGS. **6**A and **6**B illustrate an example architecture of a CNN **600** of various implementations. The CNN **600** of FIGS. **6**A and **6**B is an example of a CNN that may be trained based on the method **500** of FIG. **5**. The CNN **600** of FIGS. **6**A and **6**B is further an example of a CNN that, once trained, may be utilized in servoing a grasping end effector based on the method **700** of FIG. **7**A. Generally, a convolutional neural network is a multilayer learning framework that includes an input layer, one or more convolutional layers, optional weight and/or other layers, and an output layer. During training, a convolutional neural network is trained to learn a hierarchy of feature representations. Convolutional layers of the network are convolved with filters and optionally down-sampled by pooling layers. Generally, the pooling layers aggregate values in a smaller region by one or more downsampling functions such as max, min, and/or normalization sampling.

[0083] The CNN **600** includes an initial input layer **663** that is a convolutional layer. In some implementations, the initial input layer **663** is a 6×6 convolutional layer with stride **2**, and 64 filters. Image with an end effector **661**A and image without an end effector **661**B are also illustrated in FIG. **6**A. The images **661**A and **661**B are further illustrated being concatenated (represented by the merging lines extending from each) and the concatenated image being fed to the initial input layer **663**. In some implementations, the images **661**A and **661**B may each be 472 pixels, by 472 pixels, by 3 channels (e.g., the 3 channels may be selected from depth channel, first color channel, second color channel, third color channel). Accordingly, the concatenated image may be 472 pixels, by 472 pixels, by **6** channels. Other sizes may be used such as different pixel sizes or more or fewer channels. The images **661**A and **661**B are convolved to the initial input layer **663**. The weights of the features of the initial input layer and other layers of CNN **600** are learned during training of the CNN **600** based on multiple training examples.

[0084] The initial input layer **663** is followed by a max-pooling layer **664**. In some implementations, the max-pooling layer **664** is a 3×3 max pooling layer with 64 filters. The max-pooling layer **664** is followed by six convolutional layers, two of which are represented in FIG. **6**A by **665** and **666**. In some implementations, the six convolutional layers are each 5×5 convolutional layers with 64 filters. The convolutional layer **666** is followed by a max pool layer **667**. In some implementations, the max-pooling layer **667** is a 3×3 max pooling layer with 64 filters.

[0085] An end effector motion vector **662** is also illustrated in FIG. **6**A. The end effector motion vector **662** is concatenated with the output of max-pooling layer **667** (as indicated by the "+" of FIG. **6**A) and the concatenated output applied to a convolutional layer **670** (FIG. **6**B). In some implementations, concatenating the end effector motion vector **662** with the output of max-pooling layer **667** includes processing the end effector motion vector **662** by a fully connected layer **668**, whose output is then pointwise added to each point in the response map of max-pooling layer **667** by tiling the output over the spatial dimensions via a tiled vector **669**. In other words, end effector motion vector **662** is passed through fully connected layer **668** and replicated, via tiled vector **669**, over the spatial dimensions of the response map of max-pooling layer **667**.

[0086] Turning now to FIG. **6**B, the concatenation of end effector motion vector **662** and the output of max-pooling layer **667** is provided to convolutional layer **670**, which is followed by five more convolutional layers (the last convolutional layer **671** of those five is illustrated in FIG. **6**B, but the intervening four are not). In some implementations, the convolutional layers **670** and **671**, and the four intervening convolutional layers are each 3×3 convolutional layers with 64 filters.

[0087] The convolutional layer **671** is followed by a max-pooling layer **672**. In some implementations, the max-pooling layer **672** is a 2×2 max pooling layer with 64 filters. The max-pooling layer **672** is followed by three convolutional layers, two of which are represented in FIG. **6**A by **673** and **674**.

[0088] The final convolutional layer **674** of the CNN **600** is fully connected to a first fully connected layer **675** which, in turn, is fully connected to a second fully connected layer **676**. The fully connected layers **675** and **676** may be vectors, such as vectors of size 64. The output of the second fully connected layer **676** is utilized to generate the measure **677** of a successful grasp. For example, a sigmoid may be utilized to generate and output the measure **677**. In some implementations of training the CNN **600**, various values for epochs, learning rate, weight decay, dropout probability, and/or other parameters may be utilized. In some implementations, one or more GPUs may be utilized for training and/or utilizing the CNN **600**. Although a particular convolutional neural network **600** is illustrated in FIG. **6**, variations are possible. For example, more or fewer convolutional layers may be provided, one or more layers may be different sizes than those provided as examples, etc.

[0089] Once CNN **600** or other neural network is trained according to techniques described herein, it may be utilized to servo a grasping end effector. With reference to FIG. **7**A, a flowchart illustrating an example method **700** of utilizing a trained convolutional neural network to servo a grasping end effector is illustrated. For convenience, the operations of the flow chart are described with reference to a system that performs the operations. This system may include one or more components of a robot, such as a processor (e.g., CPU and/or GPU) and/or robot control system of robot **180**A, **180**B, **840**, and/or other robot. In implementing one or more blocks of method **700**, the system may operate over a trained CNN which may, for example, be stored locally at a robot and/or may be stored remote from the robot. Moreover, while operations of method **700** are shown in a particular order, this is not meant to be limiting. One or more operations may be reordered, omitted or added.

[0090] At block **752**, the system generates a candidate end effector motion vector. The candidate end effector motion vector may be defined in task-space, joint-space, or other space, depending on the input parameters of the trained CNN to be utilized in further blocks.

[0091] In some implementations, the system generates a candidate end effector motion vector that is random within a given space, such as the work-space reachable by the end effector, a restricted space within which the end effector is confined for the grasp attempts, and/or a space defined by position and/or torque limits of actuator(s) that control the pose of the end effector.

[0092] In some implementations the system may utilize one or more techniques to sample a group of candidate end effector motion vectors and to select a subgroup from the sampled group. For example, the system may utilize an optimization technique, such as the cross-entropy method (CEM). CEM is a derivative-free optimization algorithm that samples a batch of N values at each iteration, fits a Gaussian distribution to $M<N$ of these samples, and then samples a new batch of N from this Gaussian. For instance, the system may utilize CEM and values of $M=64$ and $N=6$, and perform three iterations of CEM to determine a best available (according to the CEM) candidate end effector motion vector.

[0093] In some implementations, one or more constraints may be imposed on the candidate end effector motion vector that can be generated at block **752**. For example, the candidate end effector motions evaluated by CEM or other technique may be constrained based on the constraints. One example of constraints are human inputted constraints (e.g., via a user interface input device of a computer system) that imposes constraints on area(s) in which grasps may be attempted, constraints on particular object(s) and/or particular object classification(s) on which grasps may be attempted, etc. Another example of constraints are computer generated constraints that impose constraints on area(s) in which grasps may be attempted, constraints on particular object(s) and/or particular object classification(s) on which grasps may be attempted, etc. For example, an object classifier may classify one or more objects based on captured images and impose constraints that restrict grasps to objects of certain classifications. Yet other examples of constraints include, for example, constraints based on a workspace of the robot, joint limits of the robot, torque limits of the robot, constraints provided by a collision avoidance system and that restrict the movement of the robot to prevent collision with one or more objects, etc.

[0094] At block **754**, the system identifies a current image that captures the end effector and one or more environmental objects. In some implementations, the system also identifies an additional image that at least partially omits the end effector, such as an additional image of the environmental objects that was captured by a vision sensor when the end effector was at least partially out of view of the vision sensor. In some implementations, the system concatenates the image and the additional image to generate a concatenated image. In some implementations, the system optionally performs processing of the image(s) and/or concatenated image (e.g., to size to an input of the CNN).

[0095] At block **756**, the system applies the current image and the candidate end effector motion vector to a trained CNN. For example, the system may apply the concatenated image, that includes the current image and the additional image, to an initial layer of the trained CNN. The system may also apply the candidate end effector motion vector to an additional layer of the trained CNN that is downstream of the initial layer. In some implementations, in applying the candidate end effector motion vector to the additional layer, the system passes the end effector motion vector through a fully connected layer of the CNN to generate end effector motion vector output and concatenates the end effector motion vector output with upstream output of the CNN. The upstream output is from an immediately upstream layer of the CNN that is immediately upstream of the additional layer and that is downstream from the initial layer and from one or more intermediary layers of the CNN.

[0096] At block **758**, the system generates, over the trained CNN, a measure of a successful grasp based on the end effector motion vector. The measure is generated based on the applying of the current image (and optionally the additional image) and the candidate end effector motion vector to the trained CNN at block **756** and determining the measure based on the learned weights of the trained CNN.

[0097] At block **760**, the system generates an end effector command based on the measure of a successful grasp. For example, in some implementations, the system may generate one or more additional candidate end effector motion vectors at block **752**, and generate measures of successful grasps for those additional candidate end effector motion vectors at additional iterations of block **758** by applying those and the current image (and optionally the additional image) to the trained CNN at additional iterations of block **756**. The additional iterations of blocks **756** and **758** may optionally be performed in parallel by the system. In some of those implementations, the system may generate the end effector command based on the measure for the candidate end effector motion vector and the measures for the additional candidate end effector motion vectors. For example, the system may generate the end effector command to fully or substantially conform to the candidate end effector motion vector with the measure most indicative of a successful grasp. For example, a control system of a robot of the system may generate motion command(s) to actuate one or more actuators of the robot to move the end effector based on the end effector motion vector.

[0098] In some implementations, the system may also generate the end effector command based on a current measure of successful grasp if no candidate end effector motion vector is utilized to generate new motion commands (e.g., the current measure of successful grasp). For example, if one or more comparisons of the current measure to the measure of the candidate end effector motion vector that is most indicative of successful grasp fail to satisfy a threshold, then the end effector motion command may be a "grasp command" that causes the end effector to attempt a grasp (e.g., close digits of an impactive gripping end effector). For instance, if the result of the current measure divided by the measure of the candidate end effector motion vector that is most indicative of successful grasp is greater than or equal to a first threshold (e.g., 0.9), the grasp command may be generated (under the rationale of stopping the grasp early if closing the gripper is nearly as likely to produce a successful grasp as moving it). Also, for instance, if the result is less than or equal to a second threshold (e.g., 0.5), the end effector command may be a motion command to effectuate a trajectory correction (e.g., raise the gripping end effector "up" by at least X meters) (under the rationale that the gripping end effector is most likely not positioned in a good configuration and a relatively large motion is required). Also, for instance, if the result is between the first and second thresholds, a motion command may be generated that substantially or fully conforms to the candidate end effector motion vector with the measure that is most indicative of successful grasp. The end effector command generated by the system may be a single group of one or more commands, or a sequence of groups of one or more commands.

[0099] The measure of successful grasp if no candidate end effector motion vector is utilized to generate new motion commands may be based on the measure for the candidate end effector motion vector utilized in a previous iteration of the method **700** and/or based on applying a "null" motion vector and the current image (and optionally the additional image) to the trained CNN at an additional iteration of block **756**, and generating the measure based on an additional iteration of block **758**.

[0100] At block **762**, the system determines whether the end effector command is a grasp command. If the system determines at block **762** that the end effector command is a grasp command, the system proceeds to block **764** and implements the grasp command. In some implementations, the system may optionally determine whether the grasp command results in a successful grasp (e.g., using techniques described herein) and, if not successful, the system may optionally adjust the pose of the end effector and return to block **752**. Even where the grasp is successful, the system may return to block **752** at a later time to grasp another object.

[0101] If the system determines at block **762** that the end effector command is not a grasp command (e.g., it is a motion command), the system proceeds to block **766** and implements the end effector command, then returns to blocks **752**, where it generates another candidate end effector motion vector. For example, at block **766** the system may implement an end effector motion command that substantially or fully conforms to the candidate end effector motion vector with the measure that is most indicative of successful grasp.

[0102] In many implementations, blocks of method **700** may be performed at a relatively high frequency, thereby enabling iterative updating of end effector commands and enabling servoing of the end effector along a trajectory that is informed by the trained CNN to lead to a relatively high probability of successful grasp.

[0103] FIG. **7**B is a flowchart illustrating some implementations of certain blocks of the flowchart of FIG. **7**A. In particular, FIG. **7**B is a flowchart illustrating some implementations of blocks **758** and **760** of FIG. **7**A.

[0104] At block **758**A, the system generates, over the CNN, a measure of a successful grasp based on the candidate end effector motion vector of block **752**.

[0105] At block **758**B, the system determines the current measure of a successful grasp based on the current pose of the end effector. For example, the system may determine the current measure of successful grasp if no candidate end effector motion vector is utilized to generate new motion commands based on the measure for the candidate end effector motion vector utilized in an immediately previous iteration of the method **700**. Also, for example, the system may determine the current measure based on applying a "null" motion vector and the current image (and optionally the additional image) to the trained CNN at an additional iteration of block **756**, and generating the measure based on an additional iteration of block **758**.

[0106] At block **760**A, the system compares the measures of blocks **758**A and **758**B. For example, the system may compare them by dividing the measures, subtracting the measures, and/or applying the measures to one or more functions.

[0107] At block **760**B, the system generates an end effector command based on the comparison of block **760**A. For example, if the measure of block **758**B is divided by the measure of block **758**A and the quotient is greater than or equal to a threshold (e.g., 0.9), then the end effector motion command may be a "grasp command" that causes the end effector to attempt a grasp. Also, for instance, if the measure of block **758**B is divided by the measure of block **758**A and the quotient is less than or equal to a second threshold (e.g., 0.5), the end effector command may be a motion command to effectuate a trajectory correction. Also, for instance, if the measure of block **758**B is divided by the measure of block **758**A and the quotient is between the second threshold and the first threshold, a motion command may be generated that substantially or fully conforms to the candidate end effector motion vector.

[0108] Particular examples are given herein of training a CNN and/or utilizing a CNN to servo an end effector. However, some implementations may include additional and/or alternative features that vary from the particular examples. For example, in some implementations, a CNN may be trained to predict a measure indicating the probability that candidate motion data for an end effector of a robot will result in a successful grasp of one or more particular objects, such as objects of a particular classification (e.g., pencils, writing utensils, spatulas, kitchen utensils, objects having a generally rectangular configuration, soft objects, objects whose smallest bound is between X and Y, etc.).

[0109] For example, in some implementations objects of a particular classification may be included along with other objects for robots to grasp during various grasping attempts. Training examples may be generated where a "successful grasp" grasping label is only found if: (1) the grasp was successful and (2) the grasp was of an object that conforms to that particular classification. Determining if an object conforms to a particular classification may be determined, for example, based on the robot turning the grasping end effector to the vision sensor following a grasp attempt and using the vision sensor to capture an image of the object (if any) grasped by the grasping end effector. A human reviewer and/or an image classification neural network (or other image classification system) may then determine whether the object grasped by the end effector is of the particular classification—and that determination utilized to apply an appropriate grasping label. Such training examples may be utilized to train a CNN as described herein and, as a result of training by such training examples, the trained CNN may be utilized to servo a grasping end effector of a robot to achieve a successful grasp, by the grasping end effector, of an object that is of the particular classification.

[0110] FIG. **8** schematically depicts an example architecture of a robot **840**. The robot **840** includes a robot control system **860**, one or more operational components **840***a*-**840***n*, and one or more sensors **842***a*-**842***m*. The sensors **842***a*-**842***m* may include, for example, vision sensors, light sensors, pressure sensors, pressure wave sensors (e.g., microphones), proximity sensors, accelerometers, gyroscopes, thermometers, barometers, and so forth. While sensors **842***a-m* are depicted as being integral with robot **840**, this is not meant to be limiting. In some implementations, sensors **842***a-m* may be located external to robot **840**, e.g., as standalone units.

[0111] Operational components **840***a*-**840***n* may include, for example, one or more end effectors and/or one or more servo motors or other actuators to effectuate movement of one or more components of the robot. For example, the robot **840** may have multiple degrees of freedom and each of the actuators may control actuation of the robot **840** within one or more of the degrees of freedom responsive to the control commands. As used herein, the term actuator encompasses a mechanical or electrical device that creates motion (e.g., a motor), in addition to any driver(s) that may be associated with the actuator and that translate received control commands into one or more signals for driving the actuator. Accordingly, providing a control command to an actuator may comprise providing the control command to a driver that translates the control command into appropriate signals for driving an electrical or mechanical device to create desired motion.

[0112] The robot control system **860** may be implemented in one or more processors, such as a CPU, GPU, and/or other controller(s) of the robot **840**. In some implementations, the robot **840** may comprise a "brain box" that may include all or aspects of the control system **860**. For example, the brain box may provide real time bursts of data to the operational components **840***a-n*, with each of the real time bursts comprising a set of one or more control commands that dictate, inter alia, the parameters of motion (if any) for each of one or more of the operational components **840***a-n*. In some implementations, the robot control system **860** may perform one or more aspects of methods **300**, **400**, **500**, and/or **700** described herein.

[0113] As described herein, in some implementations all or aspects of the control commands generated by control system **860** in positioning an end effector to grasp an object may be based on end effector commands generated based on utilization of a trained neural network, such as a trained CNN. For example, a vision sensor of the sensors **842***a-m* may capture a current image and an additional image, and the robot control system **860** may generate a candidate motion vector. The robot control system **860** may provide the current image, the additional image, and the candidate motion vector to a trained CNN and utilize a measure generated based on the applying to generate one or more end effector control commands for controlling the movement and/or grasping of an end effector of the robot. Although control system **860** is illustrated in FIG. **8** as an integral part of the robot **840**, in some implementations, all or aspects of the control system **860** may be implemented in a component that is separate from, but in communication with, robot **840**. For example, all or aspects of control system **860** may be implemented on one or more computing devices that are in wired and/or wireless communication with the robot **840**, such as computing device **910**.

[0114] FIG. **9** is a block diagram of an example computing device **910** that may optionally be utilized to perform one or more aspects of techniques described herein. Computing device **910** typically includes at least one processor **914** which communicates with a number of peripheral devices via bus subsystem **912**. These peripheral devices may include a storage subsystem **924**, including, for example, a memory subsystem **925** and a file storage subsystem **926**, user interface output devices **920**, user interface input devices **922**, and a network interface subsystem **916**. The input and output devices allow user interaction with computing device **910**. Network interface subsystem **916** provides an interface to outside networks and is coupled to corresponding interface devices in other computing devices.

[0115] User interface input devices **922** may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a touchscreen incorporated into the display, audio input devices such as voice recognition systems, microphones, and/or other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and ways to input information into computing device **910** or onto a communication network.

[0116] User interface output devices **920** may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem may include a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), a projection device, or some other mechanism for creating a visible image. The display subsystem may also provide non-visual display such as via audio output devices. In general, use of the term "output device" is intended to include all possible types of devices and ways to output information from computing device **910** to the user or to another machine or computing device.

[0117] Storage subsystem **924** stores programming and data constructs that provide the functionality of some or all of the modules described herein. For example, the storage subsystem **924** may include the logic to perform selected aspects of the method of FIGS. **3**,**4**, **5**, and/or **7**A and **7**B.

[0118] These software modules are generally executed by processor **914** alone or in combination with other processors. Memory **925** used in the storage subsystem **924** can include a number of memories including a main random access memory (RAM) **930** for storage of instructions and data during program execution and a read only memory (ROM) **932** in which fixed instructions are stored. A file storage subsystem **926** can provide persistent storage for program and data files, and may include a hard disk drive, a floppy disk drive along with associated removable media, a CD-ROM drive, an optical drive, or removable media cartridges. The modules implementing the functionality of certain implementations may be stored by file storage subsystem **926** in the storage subsystem **924**, or in other machines accessible by the processor(s) **914**.

[0119] Bus subsystem **912** provides a mechanism for letting the various components and subsystems of computing device **910** communicate with each other as intended. Although bus subsystem **912** is shown schematically as a single bus, alternative implementations of the bus subsystem may use multiple busses.

[0120] Computing device **910** can be of varying types including a workstation, server, computing cluster, blade server, server farm, or any other data processing system or computing device. Due to the ever-changing nature of computers and networks, the description of computing device **910** depicted in FIG. **9** is intended only as a specific example for purposes of illustrating some implementations. Many other configurations of computing device **910** are possible having more or fewer components than the computing device depicted in FIG. **9**.

[0121] While several implementations have been described and illustrated herein, a variety of other means and/or structures for performing the function and/or obtaining the results and/or one or more of the advantages described herein may be utilized, and each of such variations and/or modifications is deemed to be within the scope of the implementations described herein. More generally, all parameters, dimensions, materials, and configurations described herein are meant to be exemplary and that the actual parameters, dimensions, materials, and/or configurations will depend upon the specific application or applications for which the teachings is/are used. Those skilled in the art will recognize, or be able to ascertain using no more than routine experimentation, many equivalents to the specific implementations described herein. It is, therefore, to be understood that the foregoing implementations are presented by way of example only and that, within the scope of the appended claims and equivalents thereto, implementations may be practiced otherwise than as specifically described and claimed. Implementations of the present disclosure are directed to each individual feature, system, article, material, kit, and/or method described herein. In addition, any combination of two or more such features, systems, articles, materials, kits, and/or methods, if such features, systems, articles, materials, kits, and/or methods are not mutually inconsistent, is included within the scope of the present disclosure.

What is claimed is:

1. A method performed by one or more processors of a robot, the method comprising:

generating multiple candidate end effector motion vectors, each of the multiple candidate end effector motion vectors defining corresponding motion to move a grasping end effector of the robot from a current pose;

identifying a current image captured by a vision sensor of the robot, the current image capturing the grasping end effector and an object in an environment of the robot;

for each of the multiple candidate end effector motion vectors:

applying, as input to a trained deep neural network, the current image and a corresponding one of the multiple candidate end effector motion vectors;

generating, over the trained deep neural network, a corresponding grasp measure for the corresponding one of the multiple candidate end effector motion vectors, the corresponding grasp measure being generated based on the application of the current image and the corresponding one of the multiple candidate end effector motion vectors to the trained deep neural network;

selecting a given one of the multiple candidate end effector motion vectors, wherein selecting the given one is based on the generated corresponding grasp measure, for the given one, being most indicative of successful grasp from amongst the generated corresponding grasp measures;

in response to selecting the given one of the multiple candidate end effector motion vectors:

generating an end effector command that conforms to the given one of the multiple candidate end effector motion vectors; and

providing the end effector command to one or more actuators of the robot to cause movement of the end effector from the current pose to a new pose.

2. The method of claim **1**, wherein applying the current image and the corresponding one of the multiple candidate end effector motion vectors as input to the trained deep neural network comprises:

applying the current image as input to an initial layer of the trained deep neural network; and

applying the corresponding one of the multiple candidate end effector motion vectors to an additional layer of the trained deep neural network, the additional layer being downstream of the initial layer.

3. The method of claim **1**, further comprising:

identifying an additional image captured by the vision sensor, the additional image capturing the object and omitting the robotic end effector or including the robotic end effector in a different pose than that of the robotic end effector in the current image; and

applying the additional image as additional input to the trained deep neural network.

4. The method of claim **3**, wherein applying the current image and the additional image to the trained deep neural network comprises:

concatenating the current image and the additional image to generate a concatenated image; and

applying the concatenated image as input to an initial layer of the trained deep neural network.

5. The method of claim **3**, wherein the additional image omits the robot end effector.

6. The method of claim **1**, wherein generating the multiple candidate end effector motion vectors comprises:

sampling an initial group of candidate end effector motion vectors utilizing an optimization technique; and

selecting the multiple candidate end effector motion vectors from the group based on the sampling.

7. The method of claim **6**, wherein the optimization technique is the cross-entropy method.

8. The method of claim **1**, further comprising, after providing the end effector command to one or more actuators of the robot to cause movement of the end effector from the current pose to a new pose:

generating a new candidate end effector motion vector;

identifying a new current image captured by a vision sensor of the robot;

applying, as input to the trained deep neural network, the new current image and the new candidate end effector motion vector;

generating, over the trained deep neural network, a new grasp measure for the new end effector motion vector, the new grasp measure being generated based on the application of the new current image and the new candidate end effector motion vector to the trained deep neural network;

determining, based on the new grasp measure, to provide a grasp command to cause the end effector to attempt a grasp of the object.

9. A robot, comprising:

an end effector;

actuators controlling movement of the end effector;

a vision sensor viewing an environment;

a trained deep neural network stored in one or more non-transitory computer readable media;

at least one processor configured to:

generate multiple candidate end effector motion vectors, each of the multiple candidate end effector motion vectors defining corresponding motion to move the end effector from a current pose;

identifying a current image captured by the vision sensor;

for each of the multiple candidate end effector motion vectors:

apply, as input to the trained deep neural network, the current image and a corresponding one of the multiple candidate end effector motion vectors;

generate, over the trained deep neural network, a corresponding measure for the corresponding one of the multiple candidate end effector motion vectors, the corresponding measure being generated based on the application of the current image and the corresponding one of the multiple candidate end effector motion vectors to the trained deep neural network;

select a given one of the multiple candidate end effector motion vectors, wherein selecting the given one is based on the generated corresponding measure, for the given one, being most indicative of success from amongst the generated corresponding grasp measures;

in response to selecting the given one of the multiple candidate end effector motion vectors:

generate an end effector command that conforms to the given one of the multiple candidate end effector motion vectors; and

provide the end effector command to one or more actuators of the robot to cause movement of the end effector from the current pose to a new pose.

10. The robot of claim **9**, wherein applying the current image and the corresponding one of the multiple candidate end effector motion vectors as input to the trained deep neural network causes the at least one processor to:

apply the current image as input to an initial layer of the trained deep neural network; and

apply the corresponding one of the multiple candidate end effector motion vectors to an additional layer of the trained deep neural network, the additional layer being downstream of the initial layer.

11. The robot of claim **9**, wherein the at least one processor is further configured to:

identify an additional image captured by the vision sensor, the additional image capturing the object and omitting the robotic end effector or including the robotic end effector in a different pose than that of the robotic end effector in the current image; and

apply the additional image as additional input to the trained deep neural network.

12. The robot of claim **11**, wherein applying the current image and the additional image to the trained deep neural network causes the at least one processor to:

concatenate the current image and the additional image to generate a concatenated image; and

apply the concatenated image as input to an initial layer of the trained deep neural network.

13. The robot of claim **9**, wherein generating the multiple candidate end effector motion vectors causes the at least one processor to:

sample an initial group of candidate end effector motion vectors utilizing an optimization technique; and

select the multiple candidate end effector motion vectors from the group based on the sampling.

14. The robot of claim **13**, wherein the optimization technique is the cross-entropy method.

15. The robot of claim **9**, wherein the at least one processor is further configured to, after providing the end effector command to one or more actuators of the robot to cause movement of the end effector from the current pose to a new pose:

generate a new candidate end effector motion vector;

identify a new current image captured by a vision sensor of the robot;

apply, as input to the trained deep neural network, the new current image and the new candidate end effector motion vector;

generate, over the trained deep neural network, a new grasp measure for the new end effector motion vector, the new grasp measure being generated based on the application of the new current image and the new candidate end effector motion vector to the trained deep neural network;

determine, based on the new grasp measure, to provide a grasp command to cause the end effector to attempt a grasp of the object.

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

generating a training example based on sensor output generated by a robot during a grasp attempt by the robot, generating the training example comprising:

defining training example input, for the training example, that includes:

an image for a corresponding instance of time of the grasp attempt, the image capturing a robotic end effector of the robot and one or more environmental objects at the corresponding instance of time, and

an end effector motion vector defining motion of the end effector to move from an instance of time pose of the end effector at the corresponding instance of time to a final pose of the end effector for the corresponding grasp attempt,

determining a positive grasp success label, for the grasp attempt, responsive to determining both:

that the grasp attempt resulted in successful grasp, by the end effector, of an object, and

that the grasped object is of a particular classification; and

defining training example output, for the training example, that includes the determined positive grasp success label; and

training a deep neural network based on the training example.

17. The method of claim **16**, wherein the training example input further comprises:

an additional image for the grasp attempt, the additional image capturing the one or more environmental objects and omitting the robotic end effector or including the robotic end effector in a different pose than that of the robotic end effector in the image.

18. The method of claim **16**, wherein determining that the grasped object is of the particular classification is based on a further image, captured following the grasp attempt and captured after the robot turned the grasping end effector to the vision sensor while grasping the object.

19. The method of claim **18**, wherein determining that the grasped object is of the particular classification based on the further image comprises performing image classification on the further image to determine whether the grasped object is of the particular classification.

20. The method of claim **16**, wherein training the deep neural network comprises:

generating a predicted grasp measure based on applying:

the image of the training example input as input to an initial layer of the deep neural network, and

the end effector motion vector of the training example as input to an additional layer of the deep neural network, the additional layer being downstream of the initial layer; and

updating the deep neural network based on comparing predicted grasp measure to the grasp success label.

* * * * *