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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
06 March 2025 (06.03.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/046003 A1

(51) International Patent Classification:
B25J 9/16 (2006.01) *G06V 10/82* (2022.01)

(21) International Application Number:
PCT/EP2024/074176

(22) International Filing Date:
29 August 2024 (29.08.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/535,568 30 August 2023 (30.08.2023) US

(71) Applicant: **DEEPMIND TECHNOLOGIES LIMITED** [GB/GB]; 5 New Street Square, London EC4A 3TW (GB).

(72) Inventors: **VECERIK, Mel**; 6 Pancras Square, London N1C 4AG (GB). **DOERSCH, Carl**; 6 Pancras Square, London N1C 4AG (GB). **SCHOLZ, Jonathan Karl**; 6 Pancras Square, London N1C 4AG (GB).

(74) Agent: **FISH & RICHARDSON P.C.**; HighLight Business Towers Mies-van-der-Rohe-Str. 8, 80807 München (DE).

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

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

(54) **Title:** CONTROLLING AGENTS BY TRACKING POINTS IN IMAGES

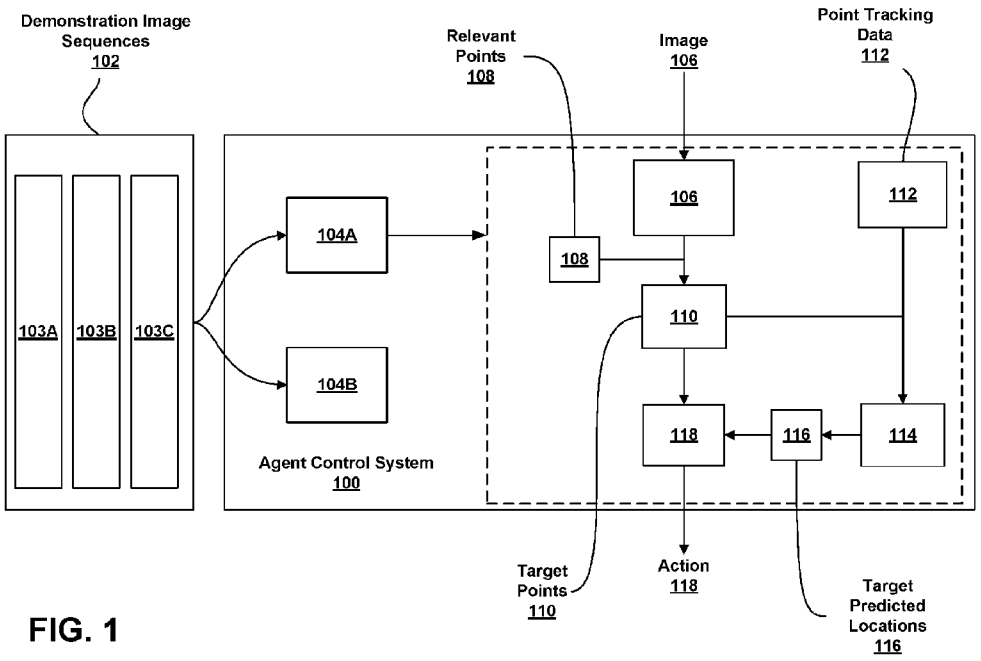


FIG. 1

(57) **Abstract:** Systems and methods for controlling agents using tracked points in images For example, controlling a mechanical agent that is interacting in a real-world environment by selecting action to be performed by the agent to perform instances of a task using images captured while the agent performs the instance of the task.

[Continued on next page]

WO 2025/046003 A1

Published:

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

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

CONTROLLING AGENTS BY TRACKING POINTS IN IMAGES

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority to U.S. Provisional Application No. 63/535,568, filed on August 30, 2023. The disclosure of the prior application is considered part of and is incorporated by reference in the disclosure of this application.

BACKGROUND

This specification relates to controlling agents using neural networks.

Neural networks are machine learning models that employ one or more layers of nonlinear units to predict an output for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer. Each layer of the network generates an output from a received input in accordance with current value inputs of a respective set of parameters.

SUMMARY

This specification describes a system implemented as computer programs on one or more computers in one or more locations that controls an agent, e.g., a robot, that is interacting in an environment by selecting actions to be performed by the agent and then causing the agent to perform the actions.

In particular, the system controls the agent to perform instances of a task using images captured while the agent performs the instance of the task.

The subject matter described in this specification can be implemented in particular embodiments so as to realize one or more of the following advantages.

Demonstration learning, i.e., learning how to perform a task from a set of demonstrations of the task being performed, enables agents to autonomously perform new instances of a task after learning from demonstrations of the task. That is, rather than manually programming the ability to perform a task, the agent learns from expert demonstrations, e.g., an agent controlled by a human, a fixed policy for the task, or an already-trained learned policy for the task.

Current approaches for demonstration learning often require task-specific engineering or require an excessive amount of demonstration data, preventing demonstration learning from being performed in an amount of time that enables practical use. For example, imitation learning, i.e., training a first system to mimic actions

demonstrated by a second different system, e.g., behavior cloning, and, inverse reinforcement learning, for image guided robot agents, are powerful but data and time intensive ways of training a robot agent to perform a task because they can take hundreds to thousands of demonstrations of a task across various environments to teach an agent to process images to perform a task robustly.

One reason for the large data and time requirements is that the inputs for demonstration learning are often raw images associated with performing the task. Because each demonstration provides a wide range of environments and scenarios, it may take a large number of demonstrations (and therefore large amounts of data and training time) for the agent to learn the appropriate internal representations necessary to generalize task performance.

This specification on the other hand, describes tracking points in images as inputs to allow faster and more general learning from demonstrations. By using points in an image as the input for demonstration learning as described in this specification, the number of demonstrations necessary to teach an agent to perform a task is reduced by orders of magnitude (and therefore the amount of data and training time necessary are reduced by orders of magnitude) while still enabling the generalization of task performance by the agent.

By tracking points in images during a demonstration task, the system can automatically extract the individual motions, the relevant points for each motion, goal locations for those points, and generate a plan that can be executed by the agent for new instances of the task, all while not requiring action-supervision, task specific training, or neural network fine tuning.

The details of one or more embodiments of the subject matter of this specification are set forth in the accompanying drawings and the description below.

Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an example agent control system.

FIG. 2 shows an example agent control system.

FIG. 3 is a flow diagram of an example process for determining relevant points for a task segment.

FIG. 4 shows an example sequence of images depicting an example process for determining relevant points for a task segment involving a robot with a camera mounted gripper.
FIG. 5 is a flow diagram of an example process for controlling an agent using an agent control system.
FIG. 6 is a flow diagram of an example process for performing a new instance of the task.
FIG. 7 shows an example of tasks performed by a robot with a camera mounted gripper using the described techniques.
Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

FIG. 1 shows an example agent control system 100. The agent control system 100 is an example of a system implemented as computer programs on one or more computers in one or more locations, in which the systems, components, and techniques described below can be implemented.

The agent control system 100 is a system that controls an agent, e.g., a robot, that is interacting in an environment (e.g. a real-world environment) by selecting actions to be performed by the agent and then causing the agent to perform the actions. In particular, the system 100 controls an agent to perform instances of a task using images captured while the agent performs the instance of the task. That is, the system 100 receives an image 106 and generates an action 118 for an agent to perform a task.

More specifically, the system 100 processes a plurality of demonstration image sequences 102 to generate data representing a plurality of task segments of the task, e.g., first task segment 104A, and second task segment 104B, and, for each task segment, a plurality of relevant points for the task segment, e.g., relevant points 108 for the first task segment 104A. Then, the system 100 processes the image 106 along with the data generated from the demonstrations 102 to generate the action 118 to be performed by the agent.

The images that make up the received image 106 and the plurality of demonstration image sequences 102 can be captured by a camera sensor of the robot or by a camera sensor located in the environment. The robot may be a mechanical robot operating in a real-world environment. The camera sensor may capture images of the robot as it performs a task in the environment (e.g. in the real-world environment). As a particular example, the robot can be a robot that includes a gripper for gripping and moving objects in the environment and the camera sensor can be positioned on the gripper (or mounted at a fixed position and orientation relative to the gripper), i.e., so that the gripper and objects gripped in the gripper

do not move significantly relative to the camera sensor unless the gripper is opened or closed.

Each demonstration image sequence of the demonstration image sequences 103A-C is a sequence of images of an agent performing a respective instance of the task, e.g., while the agent is controlled by a human, a fixed policy for the task, or an already-trained learned policy for the task. Different instances of the tasks can have, e.g., different configurations of objects in the environment but the same goal, e.g., to move a similar object to the same location, to arrange similar objects in a particular configuration, and so on.

While only three demonstration image sequences, demonstration image sequences 103A-C, are shown in FIG. 1, in practice any number of demonstration image sequences can be processed by the system 100.

The system 100 operates in two modes: an extraction mode and an action mode.

During extraction mode, the system 100 processes the plurality of demonstration image sequences 102 to generate data representing a plurality of task segments of the task, and, for each task segment, a plurality of relevant points for the task segment.

Each of the task segments corresponds to a respective portion of each of a plurality of demonstration image sequences 102.

Thus, each task segment has a corresponding portion in each of the demonstration image sequences 102, where a "portion" of a demonstration image sequence includes only a proper subset of the images in the image sequence.

In some implementations, the portions of each of the plurality of demonstration image sequences 102 associated with a task segment can contain varying numbers of images.

The system 100 can determine task segments using any of a variety of methods.

For example, the system 100 can use a time-based set of rules, e.g., predefined number of images, to generate task segments.

As another example, the system 100 can use events or actions to create task segments, such as segmenting according to robot pose information, e.g., gripper actions and forces. That is, the system 100 can divide each demonstration image sequence into portions based on positions of specified components of the robot, based on forces applied to the robot, or both. That is, the system can divide each demonstration image sequence into portions based on one or both of: positions of specified components of the robot; and forces applied to the robot.

As a specific example, for the case of a robot arm agent moving a gripped block to perform a stenciling task, the system 100 can extract gripper actuation events using the gripper openness positions and noting points where the position crosses a selected threshold. These time points are beginnings or ends of grasps and can be used to determine start and end points of task segments.

As another specific example, for the case of a robot arm agent moving a gripped block to perform a stenciling task, the system 100 can extract the beginning or the end of a force phase. To extract these, system 100 tracks the vertical force measured by a force torque sensor, smoothens the signal, and converts it to normalized force signal. Then the system 100 uses a selected threshold to determine the occurrence of a force event and uses these points to determine task segments.

While only two task segments, task segments 104A and 104B are shown in FIG. 1, in practice any number of task segments can be generated by the system 100.

As used in this specification, a "point" is a point in a corresponding image, i.e., that specifies a respective spatial position, i.e., a respective pixel, in the corresponding image. Each pixel may have one or more associated values or attributes (e.g. intensity values). For instance, each pixel may comprise one or more intensity values, each representing an intensity of a corresponding color (e.g. RGB values). The values for the pixels in an image may therefore represent features of the image.

The system 100 uses a point tracker to track a set of randomly selected points across all demonstrations 102 and generate tracking data. Based on this tracking data, the system 100 selects a subset of the points that are relevant for each task segment as the one or more relevant points for the task segment. A point may be tracked across a sequence of images by determining across the sequence, corresponding locations (points) within the images that each relate to the same features (e.g. the same section or portion of the scene or environment shown in the images). For instance, each point tracked across a sequence may represent the same position on a surface of an object within the environment. As the relative position of the camera and object move, the position of the point within the images may change.

For example, for a task involving manipulation of an object, the system 100 can select relevant points to be those on the relevant object being manipulated.

As another example, the system 100 can select points according to a set of rules such as relevant points must have a certain degree of motion or relevant points must have common locations across demonstrations.

Further details of determining relevant points for a task segment are described below with reference to FIG. 3 and FIG. 4.

The system 100 also maintains respective point tracking data for each of the relevant points for each of the task segments that identifies a respective spatial location of the point in at least some of the images corresponding to the task segment, e.g., point tracking data 112 for relevant points 108 of the first task segment 104A.

Thus, the point tracking data identifies spatial locations that represent the same point, but in different images. The point tracking data can also include occlusion scores that represent occlusion likelihoods and, further optionally, uncertainty scores that represent uncertainties in the predicted spatial locations. If the spatial locations for a point are different across different images, then the point has moved relative to the camera between the different images.

During action mode, the system 100 uses the maintained data to perform a new instance of the task, i.e., by using (i) images captured while the agent performs the new instance of the task and (ii) the relevant points for the task segments.

More specifically, the system 100 can perform the following operations at each of a plurality of time steps during the performance of the new instance of the task.

At each of the plurality of time steps, the system 100 obtains an image of the agent, e.g., the robot, at the time step, e.g., as captured by a camera sensor of the robot at the time step. As a specific example, the obtained image 106 belongs to time step t.

The system 100 identifies a current task segment for the time step.

For example, as described below, the task segments can be determined based on positions of components of the robot in the demonstration sequences or forces applied to the robot or both, i.e., so that each task segment starts when a corresponding component is in a first position or a particular force has been applied to the robot (or for the first task segment, when the instance of the task begins) and continues until a corresponding component is in a second position or a particular force has been applied to the robot. When the agent is a robot with a gripper, the segments can be based on forces applied to the gripper or positions of the gripper, i.e., openness and closedness positions that represent how open or closed the gripper is.

The system 100 can then identify the current task segment by identifying whether a criterion has been satisfied for terminating the task segment for the preceding time step. If the corresponding criterion has not been satisfied, the system 100 sets the current task segment to be the task segment for the preceding time step and, if the corresponding

criterion has been satisfied, the system 100 sets the current task segment to be the next task segment after the task segment for the preceding time step.

The system 100 determines one or more target points from the relevant points for the current task segment and determines, from the point tracking data for the task segment, a respective target predicted location of each of the target points in a future image.

As a particular example, for image 106 of time step t the system 100 determines one or more target points 110 from the relevant points 108 for the current task segment 104A and determines, from the point tracking data 112 for the task segment 104A, a respective target predicted location 116 of each of the target points 110 in a future image 114. Each target point 110 may be a location (e.g. pixel) within the image 106 that represents (e.g. shows) within the image 106 a feature (e.g. object or section of the image) corresponding to one of the relevant points 108 for the current task segment 104A.

The "future image" 114 is an image that identifies respective spatial locations of relevant points from one of the demonstration image sequences 102 stored in the point tracking data 112 that the system 100 aims to replicate.

The system 100 can determine the future image 114 as the image with corresponding relevant points most similar in position relative to the image frame to the target points 110 associated with image 106 among all the demonstration image sequences maintained and further determine the target predicted locations 116 to be the relevant points associated with the future image 114.

Generally, the target predicted locations 116 are "where" the system 100 aims to move the target points 110 to in order to replicate the "future image". That is, the target points 110 identify "what" points are relevant in the current image 106, the target predicted locations 116 determine "where" these points should be, and the generated action 118 will determine "how" to get target points 110 to target predicted locations 116.

The system 100 then causes the agent to perform an action 118 that is predicted to move the target points 110 to the target predicted locations 116.

For example, the system 100 can apply a controller, e.g., a visual servoing controller or other robotics controller, to process the one or more target points 110 and the respective target predicted location 116 of each of the target points 110 in a future image 114 to determine the action 118 that is predicted to move the target points 110 to the corresponding target predicted locations 116 and then cause the agent to perform the determined action 118, e.g., by applying a control input to one or more controllable elements, e.g., joints, actuators, and so on, of the agent.

Further details of updating the agent control system 100 and performing a new instance of a task are described below with reference to FIG. 5 and FIG. 6 respectively.

FIG. 2 shows an example agent control system 200. The agent control system 200 is an example of a system implemented as computer programs on one or more computers in one or more locations, in which the systems, components, and techniques described below can be implemented.

The agent control system 200 is a system that controls a robot that includes a gripper for gripping and moving objects in the environment and a camera sensor positioned on the gripper, i.e., so that the gripper and objects gripped in the gripper do not move significantly relative to the camera sensor unless the gripper is opened or closed. In particular, the system 200 controls the robot to perform instances of a task involving moving an object using images captured while the agent performs the instance of the task. That is, the system 200 receives an image 206 and generates an action 218 for the robot to perform the task of moving the object.

During extraction mode, the system 200 processes three demonstration image sequences 203A-C as a group of demonstration image sequences 202. The demonstrations are for the task of gripping the L-shaped block and placing it on the oval target, and FIG. 2 depicts the sequence of images in order of top to bottom for each demonstration. Whilst three demonstration image sequences are processed in the example of FIG. 2, it will be appreciated that the number of demonstration image sequences may be varied. Similarly, the number of demonstration images within each demonstration image sequence may be varied.

FIG. 2 also depicts the generation of a first task segment 204A and a second task segment 204B illustrating the respective portion of each of the plurality of demonstration image sequences 202 that make up the first task segment 204A and second task segment 204B. The system 200 divides each demonstration image sequence into portions based on forces applied to a gripper, and based on positions of a gripper, more specifically, for each of the demonstration image sequences, the first three images of the sequence make up the first task segment 204A, the fourth image corresponds to motor primitives, e.g., the gripper closing and moving up, then the next three images correspond to the second task segment 204B.

All the task segments, along with the motor primitives of the gripper, of the task are depicted in FIG. 2 under the heading “Motion Plan”.

The system 200 uses a point tracker called "tracking any point with per-frame initialization and temporal refinement" (TAPIR), a method for accurately tracking specific points across a sequence of images as described in ArXiv: 2306.08637, to generate point tracking data. The method employs two stages: 1) a matching stage, which independently locates a suitable candidate point match for each specific point on every other image, and 2) a refinement stage, which updates the trajectory based on local correlations across images. Although the system 200 uses TAPIR in the example of FIG. 2, more generally, the system 200 can use any other appropriate point tracker capable of generating the required outputs, such as BootsTap as described in ArXiv:2402.00847 and TAPNet as described in ArXiv:2211.03726. That is, any general purpose point tracking method may be used, to identify the relative motion of points across various images (frames) in a sequence. Point tracking may determine two pixels in two different images each represent the same section or portion of the scene or environment shown within the images (e.g. that are projections of the same point on the same physical surface within the environment). FIG. 2 illustrates the tracked points through connected lines across the images of the demonstrations for each task segment illustrated, e.g., the first task segment 204A illustrates tracked points associated with the relevant object for the task segment.

The system 200 then selects relevant points along with corresponding point tracking data for each task segment. Fig 2. depicts point tracking data 212, represented as three sets of continuous lines (a set for each demonstration) within a single image, for relevant points 208, represented as qt, for the first task segment 204A. In particular, the system 200 determines the relevant points across demonstrations for the first task segment 204A using object discovery and selecting the L-shaped block as the relevant object and the corresponding points on the relevant object as the relevant points.

During action mode, system 200 obtains an image 206 for time step t.

The system 200 determines the current task segment for image 206 and time step t is the first task segment 204A because the gripper has not begun to close for the first time and because the criterion of the L-shaped block being positioned underneath the gripper has not been met as can be determined upon inspecting the image underneath the heading 'current frame'. For image 206 of time step t the system 200 determines target points 210 from the relevant points 208 using a point tracker (e.g. an online version of TAPIR). Then the system 200 selects a future image 214 as the image with corresponding relevant points most similar in position to the target points 210 from the point tracking data 212 and defines the target predicted locations 216 to be the relevant points of the future image 214.

The system 200 then processes the target points 210 and target predicted locations 216 using a visual servoing controller to generate an action 218 for the robot agent.

A visual servoing controller generally refers to a control system that uses visual data, e.g., images from camera sensors, to control the actions of another system, e.g., a robot agent, in real-time by continuously processing visual data. For example, in the case of a robot agent, the visual servoing controller determines what velocity to move one or more components of the robot such that the target points 210 will move towards the target predicted locations 216. For example, in the case of a robot with a gripper, the visual servoing controller determines what velocity to move a gripper such that the target points 210 will move towards the target predicted locations 216.

The system 200 can generally use any appropriate visual servoing controller to select the action to be performed by the agent at any given time step. Examples of visual servoing are described in DOI: 10.1109/70.954764, Chen, Hanzhi, et al. "Texpose: Neural texture learning for self-supervised 6d object pose estimation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023. , and Hill, John. "Real time control of a robot with a mobile camera." Proc. 9th Int. Symp. on Industrial Robots. 1979.

FIG. 3 is a flow diagram of an example process 300 for determining relevant points for a task segment. For convenience, the process 300 will be described as being performed by a system of one or more computers located in one or more locations. For example, an agent control system, e.g., the agent control system 100 of FIG. 1, appropriately programmed in accordance with this specification, can perform the process 300.

The system selects, as initial relevant points, one or more points (step 302).

For example, the system can select the initial relevant points based at least on two criteria: (i) how close the one or more points are to one another at a last image in the task segment in each of the demonstration image sequences according to the point tracking data and (ii) how stationary the one or more points are during the task segment according to the point tracking data.

Criterion (i) refers to selecting points that end at common image locations across all demonstrations. That is, the system can calculate the location variance of the final point location across demonstrations and those points associated with a location variance below a threshold are selected. That is, points that have similar final point locations across each demonstration image sequence can be selected.

For example, for a task segment corresponding to stenciling, i.e., placing an object into its corresponding, appropriately shaped hole, a point associated with the object being

placed will most likely satisfy criteria (i) because across demonstrations the object to be placed can begin in a variety of environment locations but always ends in the appropriately shaped hole location at the end of the task segment. Therefore, points associated with the block will also begin at a variety of environment locations but end near a common location at the end of the task.

In criterion (ii), stationary refers to tracked points whose overall motion during the task segment is less than a threshold. That is, a metric of point movement relative to the frame of the image, rather than movement relative to a third-person perspective of the scene, is defined, and points with associated motion metric values above a threshold value across demonstrations are selected for.

For example, when a camera is mounted on a robot end effector for a stenciling task, points associated with an end effector within the images will not be selected for because the end effector does not move within the frame of the image.

As a particular example of determining relevant points from the point tracking data based on criteria (i) and (ii), the system can sequentially select points according to a sequence of parameterized rules that correspond to the criteria. For example, the system can first select points that satisfy a first rule such as “select points whose variance of position during the task segment across all demonstrations is greater than or equal to a particular parameter value” that corresponds to criterion (ii). Then, only evaluating the points that satisfy criteria (ii), the system can select points that satisfy a second rule such as “select points whose variance of position within the final frame across all demonstrations is less than a parameter value” that corresponds to criterion (i).

In some implementations, the system also considers other criteria when selecting the initial points.

For example, in addition to (i) and (ii) above, the system can also select the one or more points based on (iii) whether the one or more points are visible at the last image in the task segment in each of the demonstration image sequences according to the point tracking data.

Criterion (iii) refers to points that are not guaranteed to be visible at the end of the task segment, either due to occlusion, failures by sensors, or inconsistencies due to imprecise demonstrations.

For example, for a task segment corresponding to stenciling, a square block to be moved can have unique marking on a single face with corresponding points that are particularly easy to track (assuming the block face is always visible). But because the

uniquely marked block face may not always be visible, due to the marked block face being face down or facing away relative to the camera, the corresponding points to the unique marking will not be selected for using criteria (iii).

As a particular example of determining relevant points from the point tracking data based on criteria (i), (ii), and (iii), the system can sequentially select points according to a sequence of parameterized rules that correspond to the criteria. For example, after selecting points sequentially that satisfy criteria (i) and (ii) according to parameterized rules as described above, the system can then select points from the remaining set that satisfy a third rule such as "select points whose average visibility, i.e., average occlusion score, across the entire task segment is above a parameter value" that corresponds to criterion (iii).

In some implementations, the system can then generate the relevant points for the task segment using the initial relevant points.

For example, the system can select the initial relevant points previously selected as satisfying criteria (i), (ii), and (iii) as the relevant points.

The system clusters, using the point tracking data, the plurality of points to determine a plurality of clusters (step 304). The plurality of points refers to all tracked points, not just those designated as initial relevant points from step 302.

The system can cluster the plurality of points using any of a variety of methods.

For example, the system can use a "3D-motion-estimation-and-reprojection-based point clustering" method to cluster. That is, the system assumes that all points belong to one of several approximately rigid objects in the scene and that their motion can be explained by a set of 3D motions followed by reprojection, i.e., projection of 3D location of a point to a 2D location; parameterizes the 3D location of the points and the 3D transformation of those points for every image in every demonstration; and minimizes a reprojection error function to determine how many and to which clusters the points belong to as those that minimize the error function. The reprojection error function for a gripper performing a stenciling task can be, for example:

$$L(A, P) = \arg\min_{A,P} \min_{k} \Sigma_{i,t} v_{i,t} \parallel R(A_{t,k} P_{i,k}) - p_{i,t} \parallel^2$$

Where $p_{i,t}$ is the predicted location for point *i* at time *t* in the demonstrations (for simplicity, *t* indexes both time and demonstrations) and $v_{i,t}$ is a thresholded version of the occlusion probability o_t, *k* refers to the number of rigid objects in the scene, $P_{i,k}$ is the 3D location for the i'th point in the k'th object, $A_{t,k}$is a rigid 3D transformation for each object at each time, and R(x) is the reprojection function R(x) = [x[0]/x[2], x[1]/x[2]] that projects a 3D point

onto a 2D plane for which x[0], x[1], x[2] are the x, y, z coordinates in 3D and x[0]/x[2], x[1]/x[2] are the normalized x,y coordinates in the 2D plane.

For the previous example, both $P_{i,k}$ and $A_{t,k}$ can be parameterized using neural networks, which aim to capture the inductive biases that points nearby in 2D space, and also frames nearby in time, should have similar 3D configurations. Specifically, $P_{i,k} = \mathbf{P}(p_i, o_i \mid \theta_1)_k$, where θ_1 parameterizes the neural network P which outputs a matrix, and $A_{t,k} = \mathbf{A}(\phi_t \mid \theta_2)_k$, where ϕ_t is a temporally-smooth learned descriptor for image t, θ_2 is a neural network parameter for neural network A which outputs a tensor representing rigid transforms.

Also for the previous example, the optimal number of rigid objects k can be determined through a 'recursive split' method. To accomplish this, note that it can be the case that only the final linear projection layers of the two neural networks $P_{i,k}$ and $A_{t,k}$ depends on the number of clusters k: the parameters for these layers can be written as a matrix $w \in \mathbb{R}^{k \times c}$ for some number of channels c. For each such weight matrix, the system creates two new weight matrices $w' \in \mathbb{R}^{k \times c}$ and $w'' \in \mathbb{R}^{k \times c}$, where the k'th row parameterizes a new clustering where the k'th row of w has been split into two different clusters, termed 'forks' of the original weight matrix. The system computes the loss under every possible split, and optimizes for the split with the minimum loss. Mathematically, $w^K \in \mathbb{R}^{(k+1) \times c}$ defines a new matrix where the K-th row of w has been removed, and the K-th rows of both w' and w'' have been appended. The system can use w^K to compute two new 3D locations and 3D transformations A^K and P^K. Then minimize the following loss:

$$\arg\min_{\theta} \min_{K} L(A^K(\theta), P^K(\theta))$$

Here, $L(A^K(\theta), P^K(\theta))$ is the previous example reprojection error function, θ parameterizes the neural networks that output A and P, and includes w and both of the 'fork' variables w' and w''. After a multiple, e.g., a fifty, one hundred, five hundred or, more generally, a few hundred, optimization steps, the system replaces w with w^K, and creates new 'forks' of this matrix (initializing the forks with small perturbations of w^K). The system begins with $k = 1$ and repeats the recursive forking process until the desired number of objects is reached or the loss is minimized.

The system selects the relevant points using the clusters and the initial relevant points (step 306).

For example, given a clustering, every initial relevant point can cast (e.g. assign) a vote for a cluster and clusters with the largest number of votes are merged, repeating until

one or more criteria are met, such as number of clusters, number of points in clusters, and so on. Then the relevant points can be selected as belonging to those clusters that satisfy one or more criteria, such as the clusters with the most number of initial points, or the clusters whose initial points experience the most motion, and so on.

As another example, the system can use the motion-based cluster data generated to perform motion-based object segmentation, select a relevant object from among the segmentation, and select the initial relevant points on the selected object as the relevant points. That is, the system can assume that k objects are present and parameterize step 304 such that it results in k clusters corresponding to the k objects. More specifically, the constraint of producing k clusters corresponding to k objects can be enforced when minimizing the reprojection error, by fixing the number of clusters to be k, or when using initial points to vote and merge clusters, by merging or splitting clusters until k clusters result.

The system can select the relevant object from among the k clusters using one or more criteria, such as the cluster with the most points that also satisfy the above criteria (i), (ii), or (iii), or any combination of these.

The system can select, as the plurality of relevant points, points on the selected relevant object that is being manipulated, e.g., the relevant points are the points on the relevant object that satisfy criteria (i), (ii), (iii) and have occlusion scores that exceed a particular threshold.

FIG. 4 shows an example 400 sequence of images depicting an example process for determining relevant points for a task segment involving a robot with a camera mounted gripper. More specifically, each image in the example 400 sequence of images illustrates tracked points overlaid onto the last image of the task segment of moving the gripper over the cylinder shaped block.

The labeled ‘input’ image 402 illustrates all tracked points throughout the task segment. While the labeled ‘low cross-demo variance’ image 404 illustrates tracked points that end at common image locations across all demonstrations, the labeled ‘non-stationary’ image 406 illustrates tracked points whose overall motion during the task segment is greater than a threshold, and the labeled ‘motion clusters’ image 408 illustrates tracked points clustered into groups according to the objects in the image, i.e., tracked points according to the “3D-motion-estimation-and-reprojection-based point clustering” method described earlier for K objects. Tracked points present in images 404-408 are those tracked points present in image 402 that satisfy respective criteria for images 404-408. The labeled

'output' image 410 illustrates the determined relevant points as the intersection of tracked points present in images 404-408.

FIG. 5 is a flow diagram of an example process 500 for controlling an agent using an agent control system. For convenience, the process 500 will be described as being performed by a system of one or more computers located in one or more locations. For example, an agent control system, e.g., the agent control system 100 of FIG. 1, appropriately programmed in accordance with this specification, can perform the process 500.

The system obtains a plurality of demonstration image sequences, each demonstration image sequence being a sequence of images of an agent performing a respective instance of a task (step 502). For example, the system can obtain demonstration image sequences from a camera mounted directly on an agent, e.g., a robot, providing a first-person view of the task, or if the mounted camera is a 360-degree camera, providing a complete view of surrounds of the agent during the task.

As another example, the system can obtain demonstration image sequences from a camera mounted on a robot agent's end effector, e.g., a gripper, providing an end effector view of the task.

In some cases, the system can obtain demonstration image sequences from a camera associated with the environment instead of the agent. For example, the system can obtain images from an overhead camera, providing a bird's eye view of a task.

As another example, the system can obtain demonstration image sequences providing a third-person view of a task. For instance, the system can obtain demonstration image sequences from an environmentally mounted (e.g. wall mounted or tripod mounted) camera, providing a specific third-person view of a task.

The system generates data that divides the task into a plurality of task segments, each task segment including a respective portion of each of the demonstration image sequences (step 504).

The system can generate task segments using any of a variety of methods. Generally, these methods involve aligning the plurality of demonstration image sequences, i.e., synchronizing the demonstration image sequences so that corresponding events or actions across the sequences occur at matching or near matching images (e.g. at matching or near matching times or frame numbers), then segmenting each of the demonstration image sequences into equal number of task segments.

As an example of aligning demonstrations, the system can use a neural network trained using a self-supervised representation learning method to align the demonstrations. An example of such a method is described in ArXiv: 1904.07846.

As another example, the system can align the demonstrations using events or actions, such as aligning according to robot pose information, e.g., gripper actions and forces.

As an example of segmenting demonstrations once the system has aligned them, the system can create task segments according to a fixed number of images. That is, each of the demonstrations can be divided into task segments such that each task segment contains equal number of images.

As another example, the system can use scene information contained in the images to segment demonstrations. That is, one or more neural networks that learn embedding representations of demonstration image sequences can be used to process the demonstrations and generate task segments according to significant changes in scenes.

As another example, the system can use events or actions to create task segments, such as segmenting according to robot pose information, e.g., gripper actions and forces. That is, the system can divide each demonstration image sequence into portions based on positions of specified components of the robot, based on forces applied to the robot, or both. That is, the system can divide each demonstration image sequence into portions based on one or both of: positions of specified components of the robot, and forces applied to the robot.

The system, for each task segment, applies a point tracker to each of a plurality of points in the images that are in the respective portions of the image sequence to generate point tracking data for the task segment (step 506).

The system can use any of a variety of point tracking methods as the point tracker. For example, the system can use keypoint-based methods, i.e., methods that involve defining a small set of distinctive “keypoints” for an object class and identifying the keypoints in each image, for tracking random points that coincidently are keypoints. Examples of such a method are described in ArXiv: 2112.04910 and ArXiv: 1806.08756.

As another example, the system can use optical flow methods, i.e., methods that involve tracking points based on the changes in pixel intensities between two consecutive images, to track points.

As another example, the point tracker can be a neural-network based point tracker that, for each of a plurality of points, can extract query features for the point, generate

respective initial point tracking predictions for the point in each of the images in the task segment using the query features for the point and respective visual features for each of a plurality of spatial locations in the images in the task segment, and refine the respective initial point tracking predictions using a temporal refinement subnetwork to generate the point tracking data.

Examples of neural-network based point trackers are TAPNet as described in ArXiv:2211.03726, TAPIR as described in ArXiv: 2306.08637, and BootsTap as described in ArXiv:2402.00847.

Using any appropriate point tracker, the system can use the point tracker to generate point tracking data for each task segment that includes, for each tracked point and for each image in the task segment, (i) a predicted location of the point in the image and (ii) a predicted occlusion score for the point that indicates a likelihood that the point is occluded in the image.

The system, determines, for each task segment and using the point tracking data for the task segment, a plurality of relevant points for the task segment (step 508). For example, the system can determine relevant points as described with reference to FIG. 3.

The system, receives a request to perform a new instance of the task (step 510). For example, the system can receive a request from a user, or from an external system.

The system controls the agent to perform the new instance of the task using (i) images captured while the agent performs the new instance of the task and (ii) the relevant points for the task segments (step 512).

For example, at each time during execution of the new instance of the task, the system can receive an image captured at the time step; identify a current task segment corresponding to the image; determine target points from the relevant points for the current task segment; determine target predicted locations associated with completing the task; cause the agent to perform an action that is predicted to move the target points to the target predicted locations; and then, after performing the action, receive and process a new current image to repeat the previous steps until the task is complete.

Further details of performing a new instance of a task are described below with reference to FIG. 6.

FIG. 6 is a flow diagram of an example process 600 for performing a new instance of a task by controlling an agent at each of a sequence of time steps during the task. For convenience, the process 600 will be described as being performed by a system of one or more computers located in one or more locations. For example, an agent control system,

e.g., the agent control system 100 of FIG. 1, appropriately programmed in accordance with this specification, can perform the process 600.

The system can perform an iteration of the process 600 at each time step during the performance of the new instance of the task. That is, the system can continue performing iterations of the process 600 until the new instance of the task is completed.

The system obtains an image of the agent at the time step (step 602). For example, the image can come from the same source as the demonstration image sequences, i.e., a camera associated with the agent performing the task, as described earlier with reference to FIG. 5.

The system identifies a current task segment for the time step (step 604).

The system can then identify the current task segment by identifying whether a criterion has been satisfied for terminating the task segment for the preceding time step. If the corresponding criterion has not been satisfied, the system can set the current task segment to be the task segment for the preceding time step and, if the corresponding criterion has been satisfied, the system can set the current task segment to be the next task segment after the task segment for the preceding time step.

For example, the system can identify a current task segment by identifying whether a criterion, e.g., occurrence of event or action, e.g., robot pose information, e.g., gripper actions and forces, has been satisfied for the task segment and the time step value.

For the first time step, the current task segment can be set to a default task segment, such as the first task segment the system created during “extraction mode”.

In other cases, for the first time step, the system determines the current task segment by processing the first received image and determining which task segment the image is most likely to be associated with.

The system determines one or more target points from the relevant points for the current task segment (step 606). In order to determine target points from the relevant points, the system can use the previous point tracker in an online fashion, i.e., identifying target points as tracked points that are relevant points.

For example, to use the neural-network based point tracker in online fashion the neural-network based point tracker is modified to be causal. That is, the neural-network based point tracker is modified to determine the target points from the relevant points and only the images associated with the current time step and all previous timesteps for the task segment.

As a particular example, the TAPIR point tracker can be used in an online fashion by modifying its temporal refinement subnetwork to apply causal convolutions instead of temporal convolutions, as is described in ArXiv: 2308.15975. More specifically, the temporal point refinement of the TAPIR model uses a depthwise convolutional module, where the query point features, the x and y positions, occlusion and uncertainty estimates, and score maps for each image are all concatenated into a single sequence, and the convolutional model outputs an update for the position and occlusion. The depthwise convolutional model replaces every depthwise layer in the original model with a causal depthwise convolution; therefore, the resulting model has the same number of parameters as the original TAPIR model, with all hidden layers having the same shape.

The system determines, from the point tracking data for the current task segment, a respective target predicted location of each of the target points in a future image at a future time step (step 608).

For example, the system can measure the Euclidean distance between the target points and the relevant points of every image of every demonstration image sequence to select the image having the lowest distance (e.g. lowest average or net distance) as the future image. In which case, the target predicted location of each of the target points are the relevant points of the future image.

In some cases, for the previous example, there is a threshold distance value such that the future image is selected to be the following image in the corresponding demonstration image sequence to the image with the lowest Euclidean distance.

Also in some cases, for the same previous example, the target predicted location of each of the target points are determined to be the average relevant points associated with future images across demonstrations for a future time step.

The system causes the agent to perform an action that is predicted to move the one or more target points to the corresponding target predicted locations (step 610). The system can use any of a variety of appropriate controllers, i.e., algorithms that processes images and points to generate actions for an agent such that the target points become more aligned with target predicted locations.

Generally, any position-based visual servoing technique, i.e., techniques that include converting image data into real-world 3D pose data to determine actions, or image-based visual servoing, i.e., techniques that include converting image data into image features to determine actions, can be used to cause an agent perform an action that is predicted to move the one or more target points to the corresponding target predicted locations.

For example, for the case of a robot arm agent moving a gripped block to perform a stenciling task, the action corresponding to movement velocity of the arm can be determined by a controller that computes the Jacobian, i.e., an estimate of how the position of target points changes with respect to movement velocity of the robot arm, and uses the Jacobian to compute the action that corresponds to a movement velocity that minimizes the squared error between the target points and the target predicted locations under the linear approximation.

As a more specific example, for the case of a robot arm agent moving a gripped block to perform a stenciling task, given a set of target predicted locations g_t and corresponding target points p_t, the controller computes an action that minimizes the error, using a linear approximation of the function mapping actions to changes in p_t. Then the system causes the agent to perform the computed action. This process can be summarized as:

$$v_{vs} = \arg\min_{v_t} \| J_{p_t} v_t - (g_t - p_t) \|^2$$

Where t denotes the timestep, v_t is the gripper velocity, i.e., the action, and $J_{p_t}$is the image Jacobian.

FIG. 7 shows an example 700 of tasks performed by a robot with a camera mounted gripper using the described techniques.

In particular, the example 700 shows that the described techniques can be used to control a robot to perform any of a variety of tasks, even when relatively few demonstrations of the task being successfully performed are available.

For example, the example 700 shows that the system can perform “four object stack” with only four demonstrations, a large improvement over using tens to hundreds of demonstrations that other methods can require.

Some examples of environments in which the agent can be interacting and agents will now follow.

In some implementations, the environment is a real-world environment and the agent is a mechanical agent interacting with the real-world environment. For example, the agent may be a robot interacting with the environment to accomplish a goal, e.g., to locate an object of interest in the environment, to move an object of interest to a specified location in the environment, to physically manipulate an object of interest in the environment in a specified way, or to navigate to a specified destination in the environment; or the agent may

be an autonomous or semi-autonomous land, air, or sea vehicle navigating through the environment to a specified destination in the environment.

The actions may be control inputs to control a robot, e.g., torques for the joints of the robot or higher-level control commands, or the autonomous or semi-autonomous land or air or sea vehicle, e.g., torques to the control surface or other control elements of the vehicle or higher-level control commands.

In other words, the actions can include for example, position, velocity, or force/torque/acceleration data for one or more joints of a robot or parts of another mechanical agent. Actions may additionally or alternatively include electronic control data such as motor control data, or more generally data for controlling one or more electronic devices within the environment the control of which has an effect on the observed state of the environment. For example in the case of an autonomous or semi-autonomous land, air, or sea vehicle the actions may include actions to control navigation, e.g., steering, and movement e.g., braking and/or acceleration of the vehicle.

In some implementations the environment is a simulated environment and the agent is implemented as one or more computer programs interacting with the simulated environment. For example, the environment can be a computer simulation of a real-world environment and the agent can be a simulated mechanical agent navigating through the computer simulation.

For example, the simulated environment may be a motion simulation environment, e.g., a driving simulation or a flight simulation, and the agent may be a simulated vehicle navigating through the motion simulation. In these implementations, the actions may be control inputs to control the simulated user or simulated vehicle. As another example, the simulated environment may be a computer simulation of a real-world environment and the agent may be a simulated robot interacting with the computer simulation.

Generally, when the environment is a simulated environment, the actions may include simulated versions of one or more of the previously described actions or types of actions.

While this specification generally describes that the inputs are images, in some cases the inputs can include additional data in addition to or instead of image data, e.g., proprioceptive data and/or force data characterizing the agent or other data captured by other sensor of the agent.

This specification uses the term “configured” in connection with systems and computer program components. For a system of one or more computers to be configured to

perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions.

Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory storage medium for execution by, or to control the operation of, data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus.

The term “data processing apparatus” refers to data processing hardware and encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can also be, or further include, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can optionally include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand alone program or as a module, component, subroutine, or other unit

suitable for use in a computing environment. A program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a data communication network.

In this specification, the term "database" is used broadly to refer to any collection of data: the data does not need to be structured in any particular way, or structured at all, and it can be stored on storage devices in one or more locations. Thus, for example, the index database can include multiple collections of data, each of which may be organized and accessed differently.

Similarly, in this specification the term "engine" is used broadly to refer to a software-based system, subsystem, or process that is programmed to perform one or more specific functions. Generally, an engine will be implemented as one or more software modules or components, installed on one or more computers in one or more locations. In some cases, one or more computers will be dedicated to a particular engine; in other cases, multiple engines can be installed and running on the same computer or computers.

The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by special purpose logic circuitry, e.g., an FPGA or an ASIC, or by a combination of special purpose logic circuitry and one or more programmed computers.

Computers suitable for the execution of a computer program can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special purpose logic circuitry. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or

optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

Computer readable media suitable for storing computer program instructions and data include all forms of non volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks.

To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's device in response to requests received from the web browser. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone that is running a messaging application, and receiving responsive messages from the user in return.

Data processing apparatus for implementing machine learning models can also include, for example, special-purpose hardware accelerator units for processing common and compute-intensive parts of machine learning training or production, i.e., inference, workloads.

Machine learning models can be implemented and deployed using a machine learning framework, e.g., a TensorFlow framework or a Jax framework.

Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface, a web

browser, or an app through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network (LAN) and a wide area network (WAN), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other. In some embodiments, a server transmits data, e.g., an HTML page, to a user device, e.g., for purposes of displaying data to and receiving user input from a user interacting with the device, which acts as a client. Data generated at the user device, e.g., a result of the user interaction, can be received at the server from the device.

While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or on the scope of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described

program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

CLAIMS

1. A method performed by one or more computers, the method comprising:

obtaining a plurality of demonstration image sequences, each demonstration image sequence being a sequence of images of an agent performing a respective instance of a task;

generating data that divides the task into a plurality of task segments, each task segment including a respective portion of each of the demonstration image sequences;

for each task segment, applying a point tracker to each of a plurality of points in the images that are in the respective portions of the image sequence to generate point tracking data for the task segment;

determining, for each task segment and using the point tracking data for the task segment, a plurality of relevant points for the task segment;

receiving a request to perform a new instance of the task; and

controlling the agent to perform the new instance of the task using (i) images captured while the agent performs the new instance of the task and (ii) the relevant points for the task segments.

2. The method of claim 1, wherein the agent is a robot.

3. The method of claim 2, wherein each image in each of the demonstration image sequences is captured by a camera of a robot.

4. The method of claim 2 or claim 3, wherein generating data that divides the task into a plurality of task segments comprises:

dividing each demonstration image sequence into portions based on positions of specified components of the robot, based on forces applied to the robot, or both.

5. The method of claim 4, wherein the robot has a gripper and wherein dividing each demonstration image sequence into portions comprises:

dividing each demonstration image sequence into portions based on forces applied to the gripper, based on positions of the gripper, or both.

6. The method of claim 5, when dependent on claim 3, wherein the camera is positioned on the gripper of the robot.

7. The method of any preceding claim, wherein the point tracking data for each task segment comprises, for each of the plurality of points and for each image in the task segment:

(i) a predicted location of the point in the image; and

(ii) a predicted occlusion score for the point that indicates a likelihood that the point is occluded in the image.

8. The method of any preceding claim, wherein controlling the agent to perform the new instance of the task using (i) images captured while the agent performs the new instance of the task and (ii) the relevant points for the task segments comprises, at each of a plurality of time steps:

obtaining an image of the agent at the time step;

identifying a current task segment for the time step;

determining one or more target points from the relevant points for the current task segment;

determining, from the point tracking data for the current task segment, a respective target predicted location of each of the target points in a future image at a future time step; and

causing the agent to perform an action that is predicted to move the one or more target points to the corresponding target predicted locations.

9. The method of claim 8, wherein determining, from the point tracking data, a respective target predicted location of each of the target points in a future image at a future time step comprises, at a first time step:

applying the point tracker to the image of the agent at the first time step to generate predicted locations of the one or more target points in the image;

selecting, based on the predicted locations of the one or more target points in the image, a target image from a particular portion of a particular one of the demonstration image sequences in the current task segment;

selecting, as the corresponding target locations for the one or more target points, the predicted locations according to the point tracking data of the one or more target points in a

subsequent image that is subsequent to the target image in the particular demonstration image sequence.

10. The method of claim 9, wherein determining, from the point tracking data, a respective target predicted location of each of the target points in a future image at a future time step comprises, at a second time step:

applying the point tracker to the image of the agent at the second time step to generate predicted locations of the one or more target points in the image; and

determining, based at least on distances between the predicted locations of the one or more target points in the image and the predicted locations according to the point tracking data of the one or more target points in the subsequent image, whether to update the target image to be the subsequent image and to select new target predicted locations for each of the target points.

11. The method of any preceding claim, wherein determining, for each task segment and using the point tracking data for the task segment, a plurality of relevant points for the task segment comprises:

determining, using the point tracking data, a relevant object that is being manipulated during the task segment; and

selecting, as the plurality of relevant points, points on the relevant object that is being manipulated.

12. The method of any preceding claim, wherein determining, for each task segment and using the point tracking data for the task segment, a plurality of relevant points for the task segment comprises:

selecting, as initial relevant points, one or more points based at least on (i) how close the one or more points are to one another at a last image in the task segment in each of the demonstration image sequences according to the point tracking data and (ii) how stationary the one or more points are during the task segment according to the point tracking data; and

generating the relevant points using the initial relevant points.

13. The method of claim 12, wherein selecting, as initial relevant points, one or more points comprises selecting the one or more points based at least on (i) how close the one or more points are to one another at a last image in the task segment in each of the

demonstration image sequences according to the point tracking data (ii) how stationary the one or more points are during the task segment according to the point tracking data and (iii) whether the one or more points are visible at the last image in the task segment in each of the demonstration image sequences according to the point tracking data.

14. The method of claim 12 or 13, wherein generating the relevant points using the initial relevant points comprises:

clustering, using the point tracking data, the plurality of points to determine a plurality of clusters; and

selecting the relevant points using the clusters and the initial relevant points.

15. The method of any preceding claim, wherein the point tracker is a neural-network based point tracker that, for each of the plurality of points, is configured to:

extract query features for the point;

generate respective initial point tracking predictions for the point in each of the images in the task segment using the query features for the point and respective visual features for each of a plurality of spatial locations in the images in the task segment; and

refine the respective initial point tracking predictions using a temporal refinement subnetwork to generate the point tracking data.

16. The method of claim 15, wherein the temporal refinement subnetwork applies causal temporal convolutions to refine the respective initial point tracking predictions for the point.

17. A system comprising one or more computers and one or more storage devices storing instructions that when executed by the one or more computers cause the one more computers to perform the operations of the respective method of any one of claims 1-16.

18. One or more computer storage media storing instructions that when executed by one or more computers cause the one more computers to perform the operations of the respective method of any one of claims 1-16.

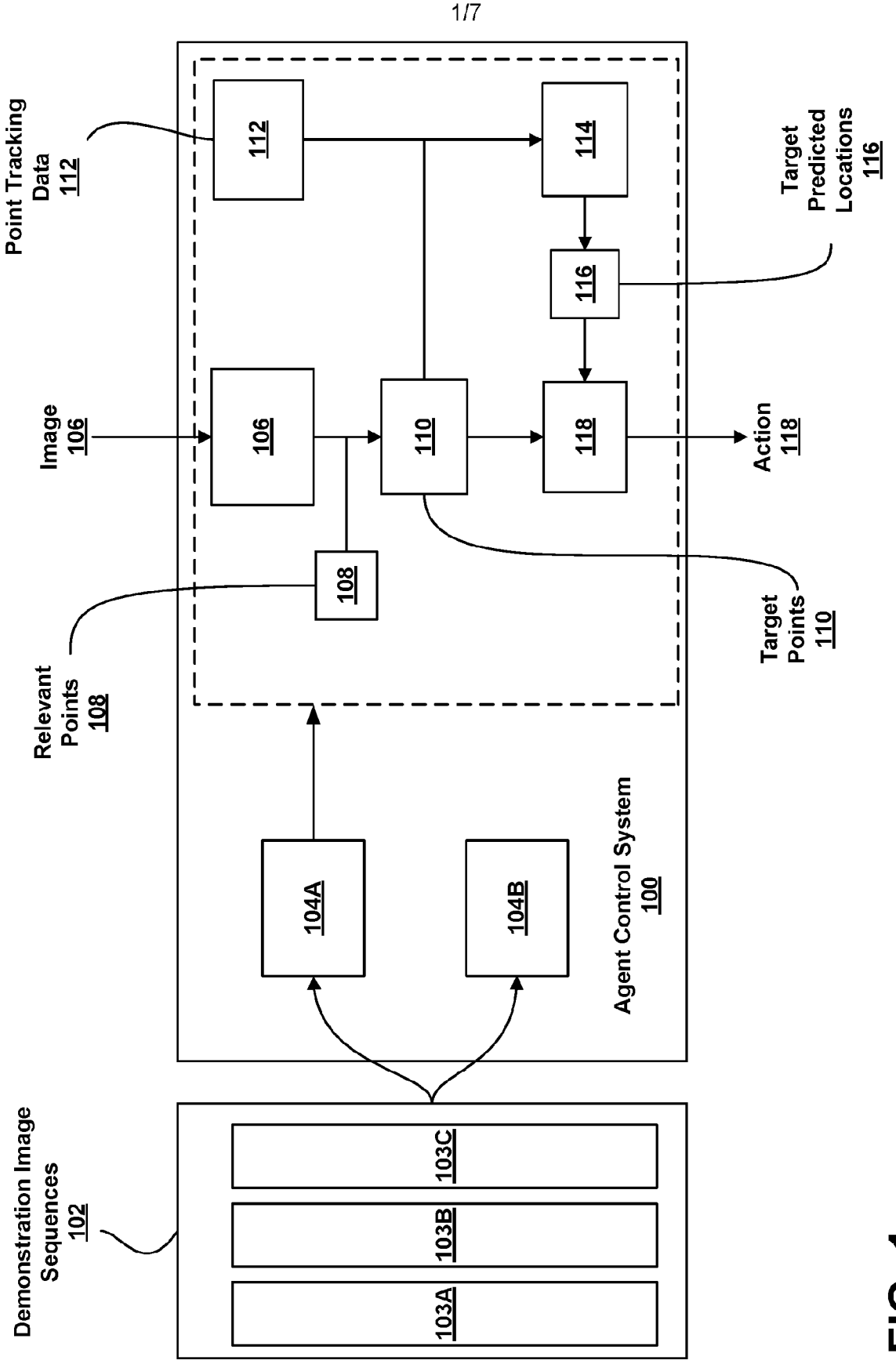

Point Tracking Data 112
Image 106
Relevant Points 108
Demonstration Image Sequences 102
103A
103B
103C
104A
104B
Agent Control System 100
106
108
110
112
114
116
118
Target Predicted Locations 116
Action 118
Target Points 110

FIG. 1

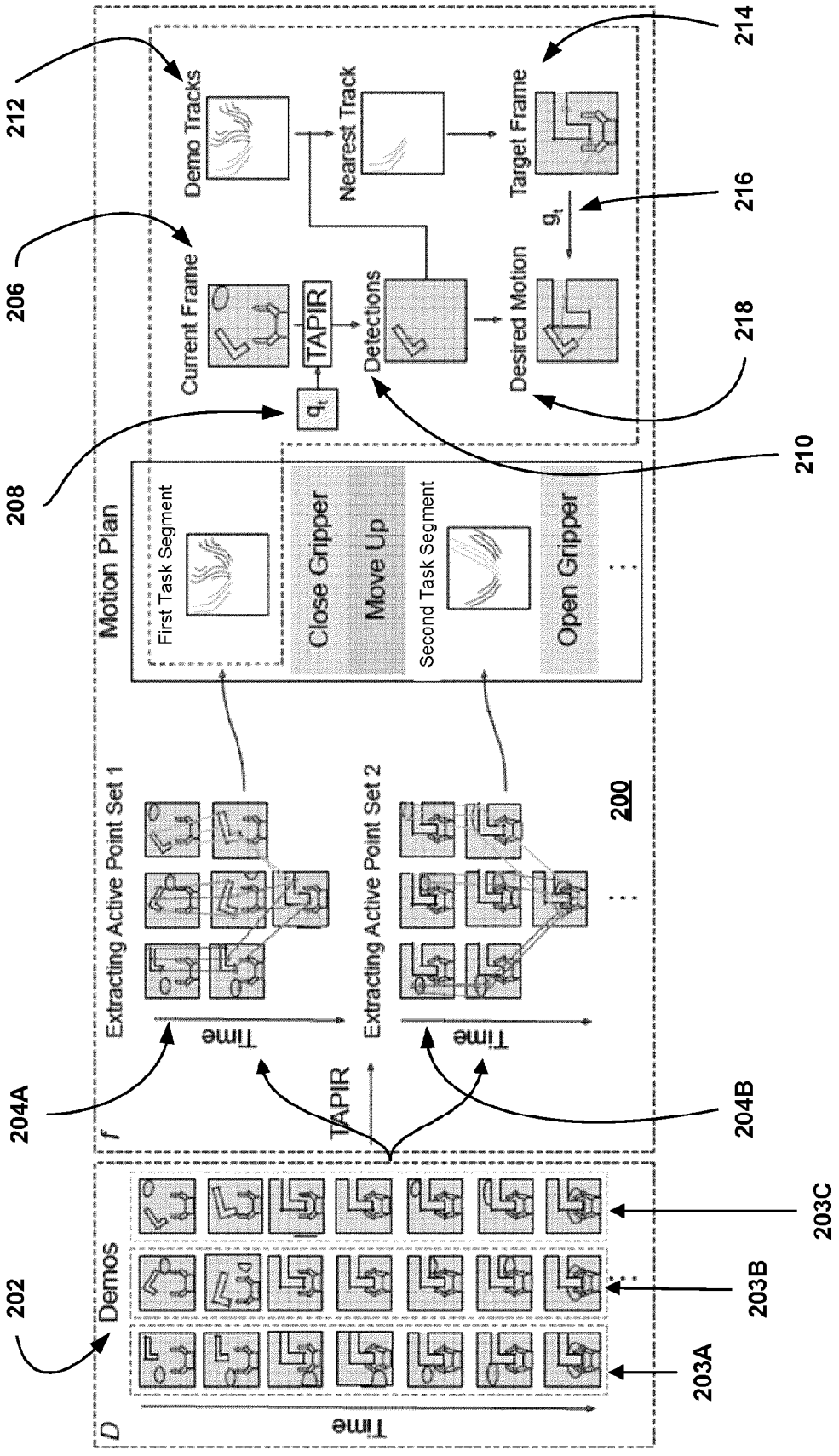

D
Demos
Time
202
203A
203B
203C
f
Extracting Active Point Set 1
Extracting Active Point Set 2
TAPIR
Time
200
204A
204B
Motion Plan
First Task Segment
Close Gripper
Move Up
Second Task Segment
Open Gripper
208
210
Current Frame
TAPIR
Detections
Demo Tracks
Nearest Track
Target Frame
Desired Motion
206
212
214
216
218

FIG. 2

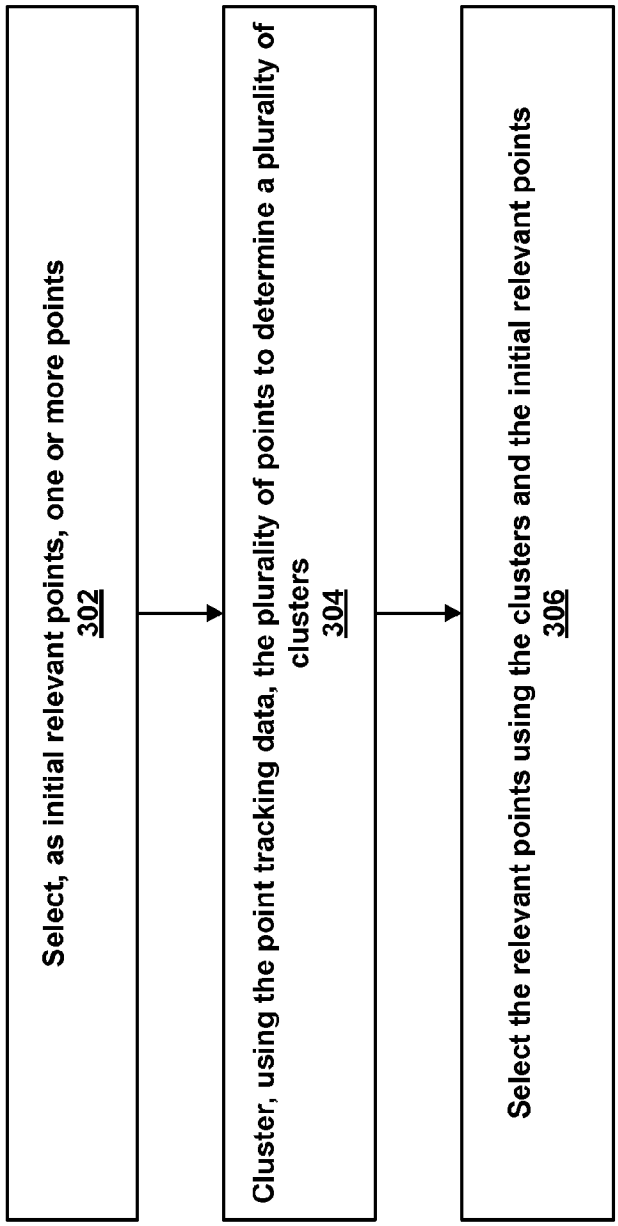

Select, as initial relevant points, one or more points
302
Cluster, using the point tracking data, the plurality of points to determine a plurality of clusters
304
Select the relevant points using the clusters and the initial relevant points
306
300

FIG. 3

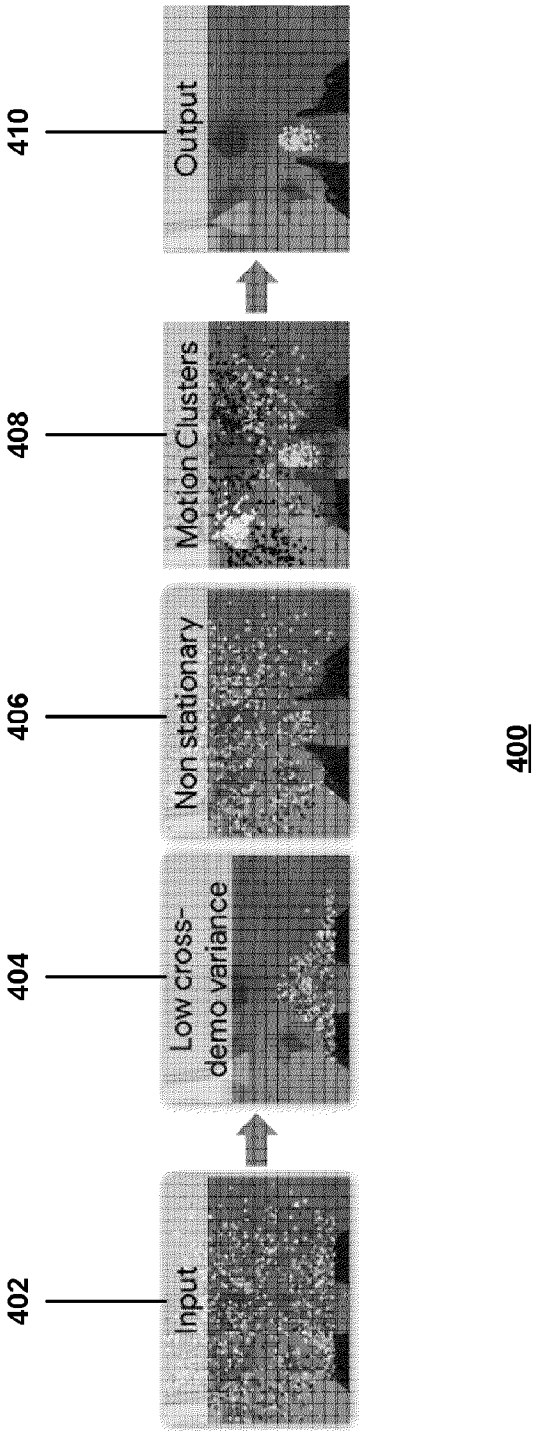

410
Output
408
Motion Clusters
406
Non stationary
404
Low cross-demo variance
402
Input
400

FIG. 4

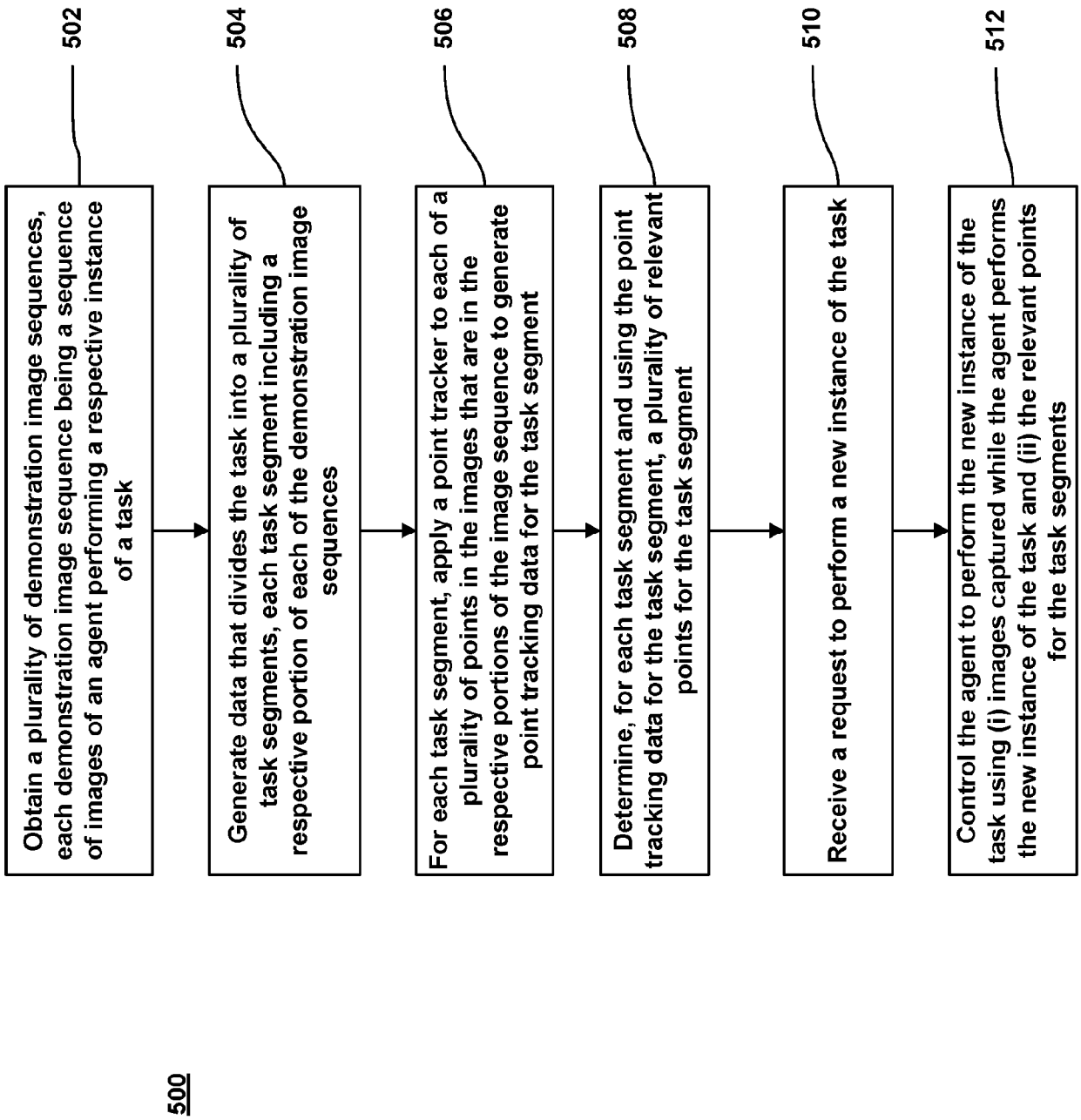

500
502
Obtain a plurality of demonstration image sequences, each demonstration image sequence being a sequence of images of an agent performing a respective instance of a task
504
Generate data that divides the task into a plurality of task segments, each task segment including a respective portion of each of the demonstration image sequences
506
For each task segment, apply a point tracker to each of a plurality of points in the images that are in the respective portions of the image sequence to generate point tracking data for the task segment
508
Determine, for each task segment and using the point tracking data for the task segment, a plurality of relevant points for the task segment
510
Receive a request to perform a new instance of the task
512
Control the agent to perform the new instance of the task using (i) images captured while the agent performs the new instance of the task and (ii) the relevant points for the task segments

FIG. 5

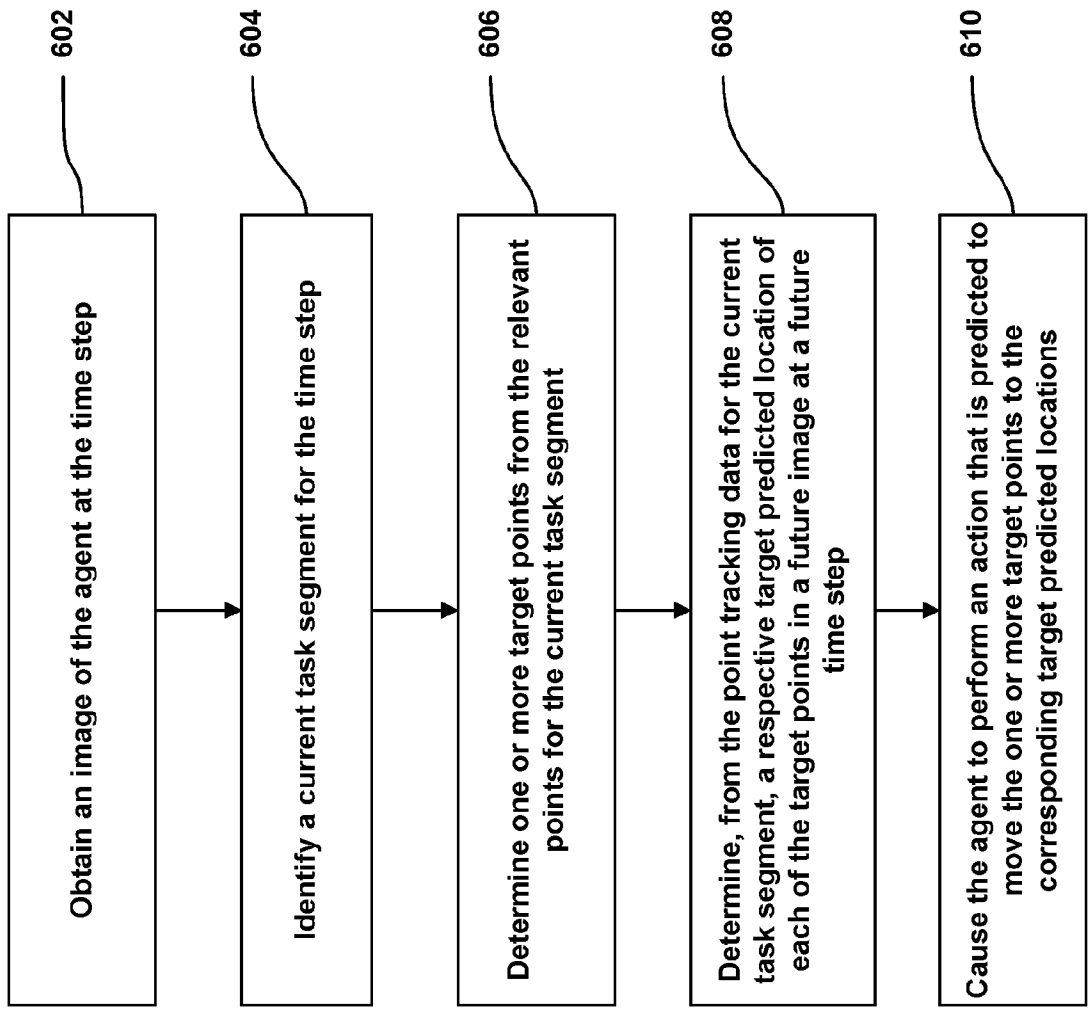

600
602
Obtain an image of the agent at the time step
604
Identify a current task segment for the time step
606
Determine one or more target points from the relevant points for the current task segment
608
Determine, from the point tracking data for the current task segment, a respective target predicted location of each of the target points in a future image at a future time step
610
Cause the agent to perform an action that is predicted to move the one or more target points to the corresponding target predicted locations

FIG. 6

Task name	Description	# Demos
Apple on jello	Pick up the apple and place it on jello.	5
Juggling ball stack	Pick up a green juggling ball and place it on a blue ball.	3
Four object puzzle	Place four wooden objects into their cutouts on a puzzle board.	6
Lego Stack	Put orange LEGO brick on top of a blue brick and push them together.	4
Tapir to robot	Pick up the plush tapir toy and place it next to the wooden robot.	4
Gluing	Glue the 2 wooden block together and place them beside the white gear.	5
Gear on grid	Pick up the white gear and place it on the grid.	6
Four object stack	Stack 4 wooden objects on top of each other.	4
Pass the butter	Pick up a butter and place it in human's hand.	5

700

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2024/074176

A. CLASSIFICATION OF SUBJECT MATTER
INV. B25J9/16 G06V10/82
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
B25J G06V G06N G05B

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y A	US 2016/243704 A1 (VAKANSKI ALEKSANDAR [US] ET AL) 25 August 2016 (2016-08-25) paragraph [0006] - paragraph [0010] paragraph [0034] - paragraph [0040] paragraph [0078] paragraph [0084] paragraph [0088] - paragraph [0089] paragraph [0095] paragraph [0104] - paragraph [0105] figures 1-4	1-6,8, 11,17,18 7,15,16 9,10, 12-14

-/--

[x] Further documents are listed in the continuation of Box C.

[x] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 December 2024	19/12/2024

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

1

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

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2024/074176

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	CARL DOERSCH ET AL: "TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 14 June 2023 (2023-06-14), XP091538130, cited in the application	7,15,16
A	abstract Sections 1 to 3 Section 5 figures 1-3	1,2,17, 18

A	XIANGHAI WU ET AL: "Human-inspired robot task learning from human teaching", 2008 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. THE HALF-DAY WORKSHOP ON: TOWARDS AUTONOMOUS AGRICULTURE OF TOMORROW; PASADENA, CA, USA, MAY 19-23, 2008, IEEE, PISCATAWAY, NJ, USA, 19 May 2008 (2008-05-19), pages 3334-3339, XP031273273, ISBN: 978-1-4244-1646-2 abstract Sections I, III and IV	1,2,4,5, 17,18

A	SING BING KANG ET AL: "Toward Automatic Robot Instruction from Perception-Mapping Human Grasps to Manipulator Grasps", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, IEEE INC, NEW YORK, US, vol. 13, no. 1, 1 February 1997 (1997-02-01), XP011053174, ISSN: 1042-296X abstract Section II Sections VI.B to VI.D	1,2,4,5, 17,18

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2024/074176

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016243704 A1	25-08-2016	CA 2928645 A1	30-04-2015
		US 2016243704 A1	25-08-2016
		WO 2015058297 A1	30-04-2015