HHS Public Access

Author manuscript

Nat Protoc. Author manuscript; available in PMC 2020 October 26.

Published in final edited form as:

Nat Protoc. 2020 March ; 15(3): 1237–1254. doi:10.1038/s41596-019-0286-8.

Skilled reaching tasks for head-fixed mice using a robotic manipulandum

Mark J. Wagner[1,*], **Joan Savall**[1], **Tony Hyun Kim**[1,2], **Mark J. Schnitzer**[1,3,*], **Liqun Luo**[1,*]

[1]Department of Biology and Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA.

[2]Department of Electrical Engineering, Stanford University, Stanford, CA, USA.

[3]Department of Applied Physics, Stanford University, Stanford, CA, USA.

Abstract

Skilled forelimb behaviors are among the most important for studying motor learning in multiple species including humans. This protocol describes learned forelimb tasks for mice using a two-axis robotic manipulandum. Our device provides a highly compact adaptation of actuated planar two-axis arms that is simple and inexpensive to construct. This paradigm has been dominant for decades in primate motor neuroscience. Our device can generate arbitrary virtual movement tracks, arbitrary time-varying forces or arbitrary position- or velocity-dependent force patterns. We describe several example tasks permitted by our device, including linear movements, movement sequences and aiming movements. We provide the mechanical drawings and source code needed to assemble and control the device, and detail the procedure to train mice to use the device. Our software can be simply extended to allow users to program various customized movement assays. The device can be assembled in a few days, and the time to train mice on the tasks that we describe ranges from a few days to several weeks. Furthermore, the device is compatible with various neurophysiological techniques that require head fixation.

Reporting Summary

[*]**Correspondence and requests for materials** should be addressed to M.J.W. or M.J.S. or L.L. mjwagner@stanford.edu; mschnitz@stanford.edu; lluo@stanford.edu.

Author contributions

M.J.W. and J.S. designed the manipulandum. M.J.W. designed and implemented the electronics and software code for robot control and designed and implemented behavioral tasks and training strategies. T.H.K. designed the mouse-restraining tube and contributed to robot assembly design. M.J.S. and L.L. supervised the project. All authors contributed to the manuscript writing.

Data availability

An example raw data file is provided in Supplementary Data 1. Additional data are available upon reasonable request.

Code availability

LabVIEW code used for operating the robotic manipulandum is available from GitHub: https://GitHub.com/mjwagner/haptic-for-mice.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41596-019-0286-8.

Peer review information *Nature Protocols* thanks Silvestro Micera and the other anonymous reviewer(s) for their contribution to the peer review of this work.

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Introduction

Learned reaching behaviors using robotic manipulanda have long been a mainstay of motor neuroscience in human and nonhuman primate studies[1–4], and are also used in nonmotor and cognitive neuroscience studies that require the measurement of motor output[5]. More recently, various rodent forelimb tasks adapting features of the reaching paradigms in primates have become more widely used as well[6–11]. Forelimb tasks in rodents allow convenient access to numerous models of neurological disorders. Such studies can also be made amenable to various genetic tools for neural perturbations, for in vivo brain imaging and for electrophysiology. Nevertheless, forelimb adaptations of primate reaching devices have not been widely available and have typically required extensive engineering expertise to fabricate, assemble and program[12–14]. In our research, we have developed a direct rodent adaptation of the planar two-axis robotic manipulanda used in primate research that can be easily and inexpensively constructed from a small number of simple parts with little engineering expertise needed for implementation. It is also compact enough to be compatible with complex microscopy and electrophysiology methodologies[15,16].

Development of the approach

Our device recapitulates the capabilities of primate planar manipulanda, including a large workspace, arbitrary position- or velocity-dependent force patterns or constrained movements along arbitrary virtual tracks in the *x*-*y* plane. We also designed our device to be highly compact (73 mm high × 69 mm wide × 83 mm deep) so that in vivo brain microscopy and electrophysiology equipment can gain unobstructed access to the space around the animal. The design is also streamlined, requiring only five custom 3D-printed linkages and a single custom machined mount that can be commercially fabricated, and can thus be assembled with little engineering expertise.

Our design permits a wide variety of learned rodent forelimb movement assays ranging from simple linear reaching movements[16], to movement sequences[15], to a variety of forelimb aiming tasks, a subset of which we describe in this protocol. The key elements of the design are lightweight 3D-printed plastic linkages that, when assembled, allow motion with low inertia and friction, and little anisotropy over the planar workspace. These features are important for mice to learn and execute complex movement trajectories ~200 trials per session, due to the small size of mice and the large effects of relatively small forces on their paw motion. We employ the two-degree-of-freedom selective compliance (‘double SCARA’) design used in devices constructed for larger organisms[17]. This permits a large planar workspace (7.75 cm^2), which we have found to be useful for developing tasks in which animals alternate between executing multiple distinct movement types (e.g., trajectories aimed in different directions in distinct spatial locations)[15]. This, in turn, allows the neural underpinnings of different types of forelimb movements to be distinguished in single sessions (e.g., by comparing the effects of neural perturbation on each movement type or comparing physiological recordings during each movement type)[15].

In this protocol, we first describe the manufacture and assembly of our device and provide the relevant mechanical drawings. We then detail the electronics used to control the device and the programmatic strategy for operating these electronics during behavioral experiments, including example code that experimenters can easily extend to program arbitrary custom movement assays. Finally, we summarize the procedure for training animals to learn and execute several example tasks.

Applications and limitations of the method

The procedures that we describe here apply to studies in which head-fixed mice execute skilled forelimb movements for water reward. There is no fundamental reason this device could not be operated by a freely moving animal, although we have not attempted such experiments. In addition, it is likely that the device could be used by other small animals such as rats. To date, however, we have not attempted rat studies. It is likely, for example, that because rats are larger than mice, a larger device would be needed (i.e., one capable of producing and withstanding larger forces). In this scenario, we anticipate that it would be straightforward to replace the motor/encoder units that we recommend with larger-diameter models. This, in turn, would require scaling up the size of the custom computer-aided design (CAD) parts that we provide by a corresponding amount, updating the linkage lengths in the LabVIEW code to ensure correct geometric transformations and updating the mechanical parameters used by the device controllers.

A limitation of the current design is that the force applied by the animal to the handle is not directly measured by a force transducer. In addition, our device does not include a touch sensor to sense skin contact from the animal with the handle. In this design, we sought to produce a device with minimal inertia and friction and highly isotropic motion across movement directions (achieved by the simple construction from lightweight 3D-printed plastics and low-inertia DC motors). We found that the addition of electronic components to the handle, with their associated wiring, made it very difficult to maintain the lightweight, low-friction, isotropic mechanical properties of the minimal design presented here. Nevertheless, such additional components could in principle be added to revised versions of our design.

Comparison with other methods

Several alternative devices for rodent forelimb assays have been described. For example, a common behavioral assay uses a one-dimensional lever-type device[7], which is only appropriate for studying linear pushing or pulling movements. Alternatively, a two-degree-of-freedom joystick-type device has also been described[6]. This device has been used in conjunction with a magnet to provide a fixed-magnitude transverse magnetic field triggered on the handle position. However, the lack of a true controlled force actuator output prevents design and application of arbitrary two-dimensional time-, position- or velocity-dependent forces, or arbitrary constrained trajectories.

A three-degree-of-freedom actuated manipulandum has also been described, primarily for assaying pulling motions in freely moving rats (ETH Pattus[12–14]). This device shares general capabilities with our device, with a few notable differences. First, the ETH device possesses

additional mechanisms for actuating a third degree of freedom for wrist rotation, whereas our device is designed solely for planar motion. Second, our device was designed for use in head-fixed mice to be compatible with in vivo brain microscopy and electrophysiology methodologies. Thus, compared to the ETH device, our design is much more compact (73 × 69 × 83 mm versus 212 × 150 × 160 mm[13]) and occupies only the space ventral to the animal, to allow unobstructed access to the animal from above for microscopy and physiology. Finally, because we aimed to produce a device that could be assembled in neuroscience laboratories with limited mechanical engineering expertise, the construction is much simpler, requiring only five custom 3D-printed linkages and a single custom aluminum mount.

Compared to two-degree-of-freedom planar manipulanda of similar design that are used in human and non-human primate studies[1,4,18,19], our device recapitulates the large planar workspace and real-time control of arbitrary time-varying and kinematic-dependent force patterns. One difference is that, while most primate devices also have force transducers on their handles, as described above, the extremely low inertia of our device makes such an addition difficult, to also maintain our device's isotropic mechanical properties.

Level of expertise needed to implement the protocol

Basic engineering and programming experience are sufficient to assemble the device. Surgical skills are needed for head-fixation preparations, and basic animal handling experience is needed to train mice to perform the tasks described. For extending the capabilities of our example code, or for customizing the electronics, a moderate skill in programming in the National Instruments LabVIEW environment is needed.

Experimental design

Device mechanical assembly—The device consists of four 3D-printed plastic mechanical linkages: two 'upper arm' and two 'lower arm' linkages (Fig. 1a). The two lower linkages are connected to each other at a 'wrist' joint that consists of ball bearings and an extended shaft on which is mounted a 3D-printed plastic handle for the animal to grasp with its forepaw (Fig. 2a,b). Each upper and lower linkage pair is connected by an 'elbow' joint consisting of a 2-mm shaft in 2-mm ball bearings (Fig. 2c,d). Each upper arm linkage is also connected to one of the two motor shafts (Fig. 2e–h). The two motor-encoder units are affixed to a custom mounting bracket (Fig. 1b), and each motor unit has an integrated backshaft-mounted optical rotary encoder (Fig. 1c; Maxon Motors). We also place the animal in a custom 3D-printed transparent plastic tube (Fig. 1d).

The forward and inverse kinematics to move between handle cartesian coordinates and the shoulder rotational coordinates, as well as the Jacobian to convert from cartesian forces to joint torques, follow standard five-bar linkage configurations (e.g., as in ref. [17]), with link lengths of 37 mm for the upper linkages, 31 mm for the lower linkages and 22 mm for the separation between the motor shafts. These transformations are all executed in the provided code that we describe below.

The range of motion of the arms of the device is constrained by three M3 screws mounted at indexed locations on the mounting bracket (Fig. 2g). The motion limits set by these screws are chosen to avoid reaching the mechanical singularities associated with this design (beyond which there is no longer a 1:1 mapping between shoulder rotational coordinates and handle cartesian position). In addition, these screws define indexed locations that the robot can sense to locate its 'home' position, so that the software can determine the absolute rotational positions of the two shoulder joints. The fully assembled device (Fig. 1e,f; model available at https://grabcad.com/library/full-protocol-2) is compact enough to fit in typical in vivo physiology and microscopy rigs (Fig. 1g).

Control electronics—The control electronics for this device are based on the National Instruments (NI) cRIO (compact reconfigurable input-output) environment. This consists of a 'chassis' that contains an FPGA (field-programmable gate array) chip (Fig. 3a and Supplementary Fig. 1a), a personal computer (PC) running a real-time operating system (OS) (RT-PC; Fig. 3b and Supplementary Fig. 1b) and several modules that interface with the FPGA and provide the device with customizable input-output capabilities. Finally, the cRIO system communicates with a standard PC via universal serial bus (USB) (Fig. 3c and Supplementary Fig. 1c). All three devices (FPGA with input/output (I/O) modules, RT-PC and standard PC) are programmed and communicate through the LabVIEW programming environment. The system we describe here uses three modules: two motor control modules that monitor rotations of the optical encoder, drive the motors and read analog motor winding current (NI 9505); and one module that provides 32 digital input and output channels (NI 9403) for reading or controlling any other devices needed for in vivo behavior and physiology studies (Fig. 3d,e; our code implements communication with a water delivery solenoid, lick sensor and two microscope frame clocks).

These electronics are used to operate the device in one of two modes. In the first mode, the device can be sent to and held at arbitrary positions in the workspace. This is used to 'lock' the handle position (e.g., at the home position before trial start, or at the trial end position), as well as to return the handle from arbitrary trial end positions back to the home position. In the second mode, the device allows the animal to execute a forelimb movement. In this case, the robot applies force vectors to the handle depending on the animal's instantaneous position and velocity and the trial type—i.e., for a constrained trajectory (Fig. 4b,c), the force needed to constrain motion to the desired track; for an aiming movement (Fig. 4f,i), no force or simple simulated friction. Any other arbitrary force patterns can be encoded through LabVIEW, as decribed below.

Control software—The electronics are programmed and operated in a hierarchy of controllers. Input/output operations through the modules are executed in FPGA, which operates on a 40 MHz clock (Fig. 3a). The FPGA also contains low-level device controllers (Supplementary Fig. 1a). At all times, the FPGA controls motor current (which is related by a scale factor to motor torque) at 10 kHz by comparing the current in the motor windings to the 'desired' current and correcting errors by varying the PWM (pulse-width-modulated) voltage delivered to the motor (Supplementary Fig. 1a; 'Torque controllers'). This and the other controllers are 'PID' (proportional-integral-derivative).

When the robot is instructed to hold a fixed specific position, an additional FGPA process is activated to control each motor's rotational angle (Supplementary Fig. 1a; 'Position controllers'). This position controller compares the present rotational angle of each motor to the desired angle (at 1 kHz), by adjusting the 'desired current' used in the 'Torque controller' described previously.

Above the FPGA controller is the RT-PC controller, which executes at 1 kHz (Fig. 3b). This software converts between the rotational positions of the encoders and the cartesian *x*-*y* position of the handle via the forward dynamics (joints to cartesian) and inverse dynamics (cartesian to joints) kinematic transformations (Supplementary Fig. 1b). It uses this information to operate the general 'state machine' of the device (Supplementary Fig. 1b). In our example code, device states include: 'homing' the device to find the positional origin, sending the device to a desired location via a controlled trajectory, holding position and allowing the animal to execute a trial. The code executed during a behavioral trial depends on trial type, which the experimenter sets programmatically (Supplementary Fig. 1b). The trial execution code consists of the following: (i) releasing the robot from the prior position-hold; (ii) monitoring the handle position and velocity at each point in time; (iii) computing the desired force vector to apply to the handle, depending on the present position and velocity and trial type; and (iv) terminating the trial upon completion or termination criterion by initiating the hold-position state. At each point in time, the device converts the desired cartesian force into desired torques in each motor winding, via the transpose of the velocity Jacobian matrix[12] (matrix of first-order partial derivatives of the forward kinematics transform). The desired torques are then relayed to the torque controllers in the FPGA (Supplementary Fig. 1a). In a separate process, the RT-PC also samples data from the FPGA, logs it to a memory buffer and periodically empties the buffer into a USB stream, to be read and emptied periodically by the standard PC (not shown).

At the top of the hierarchy, the standard PC executes a high-level control loop (Fig. 3c). This is the software with which users interact during behavioral experiments (Supplementary Fig. 1c). It consists of a front panel to monitor robot state and animal behavioral performance and to adjust various robot and behavioral parameters (Fig. 5d). A glossary of controls in the user interface is listed in Box 1. It also reads 'trial lists' written by the experimenter, which specify the parameters that define each trial. Trial lists contain in order the following: trial type, desired movement angle (rads), movement length (mm), turning movement length (mm), turning movement angle (rads), maximum error, reward magnitude scale factor, movement timeout (ms) and reward type. These are defined in 'TrialListElements.ctl'. The standard PC also operates a state machine that loads trials, updates robot parameters, executes trials through the RT-PC state machine and delivers water reward to the animal. Finally, the standard PC periodically empties the USB stream from the RT-PC into a memory buffer and allows the experimenter to write the data samples in the memory buffer to a hard disk. The physical connections between the devices that implement these processes are described below and in Fig. 3d,e.

Trial types and real-time state-dependent force output

The central capability of our device is that every millisecond (i.e., at 1 kHz) it generates a two-dimensional *x* and *y* force that can be programmed as arbitrary mathematical functions of instantaneous present and past position and velocity. In particular, users can specify at every point in time (at 1 kHz) the following relationships:

$$\overrightarrow{F} = \mathrm{A} \cdot \overrightarrow{x} + \mathrm{B} \cdot \overrightarrow{y} + \mathrm{C} \cdot \overrightarrow{v}_x + \mathrm{D} \cdot \overrightarrow{v}_y$$

Where

$$\overrightarrow{x} = \begin{bmatrix} x_t \\ x_t - 1 \\ \vdots \\ x_t - n \end{bmatrix}, \overrightarrow{y} = \begin{bmatrix} y_t \\ y_t - 1 \\ \vdots \\ y_t - n \end{bmatrix}, \overrightarrow{v}_x = \begin{bmatrix} v_x^t \\ v_x^t - 1 \\ \vdots \\ v_x^t - n \end{bmatrix}, \overrightarrow{v}_y = \begin{bmatrix} v_y^t \\ v_y^t - 1 \\ \vdots \\ v_y^t - n \end{bmatrix}, \overrightarrow{F} = \begin{bmatrix} F_x^t \\ F_y^t \end{bmatrix}$$

x_t and y_t are the position, v_x^t and v_y^t are the velocity, and F_x^t and F_y^t are the force, in the *x* and *y* dimensions, respectively, at time point *t*. A, B, C and D are matrices of constants that can take on any user-defined values. (In our code, n, the number of past position and velocity measurements kept in memory for subsequent force computations, is set to 8, but this could in principle be set to larger buffers.)

Or

$$\begin{aligned} F_x^t = {} & \mathrm{a}_1^x \cdot x_t + \mathrm{b}_1^x \cdot y_t + \mathrm{c}_1^x \cdot v_x^t + \mathrm{d}_1^x \cdot v_y^t \\ & + \mathrm{a}_2^x \cdot x_t - 1 + \mathrm{b}_2^x \cdot y_t - 1 + \mathrm{c}_2^x \cdot v_x^t - 1 + \mathrm{d}_2^x \cdot v_y^t - 1 + \ldots \\ F_y^t = {} & \mathrm{a}_1^y \cdot x_t + \mathrm{b}_1^y \cdot y_t + \mathrm{c}_1^y \cdot v_x^t + \mathrm{d}_1^y \cdot v_y^t \\ & + \mathrm{a}_2^y \cdot x_t - 1 + \mathrm{b}_2^y \cdot y_t - 1 + \mathrm{c}_2^y \cdot v_x^t - 1 + \mathrm{d}_2^y \cdot v_y^t - 1 + \ldots \end{aligned}$$

The generation of such arbitrary position- and velocity-dependent forces is demonstrated in practice in both of the tasks illustrated in Fig. 4b,c.

Figure 4b uses the following position- and velocity- dependent force generation function to constrain motion to a forward trajectory:

$$F_x^t = \mathrm{a} \cdot x_t + \mathrm{b} \cdot v_x^t;\ F_y^t = 0$$

Where a and b are negative numbers, such that this produces an instantaneous *x* force that is proportionate in magnitude and opposite in direction to the mouse's instantaneous distance from the *y* axis and to the *x* velocity, while allowing unimpeded motion along the *y* axis.

Figure 4c uses the following position- and velocity-dependent force generation function to constrain motion to a forward trajectory for 6 mm, followed by a lateral trajectory of 6 mm:

$$F_x^t = \begin{cases} a \cdot x_t + b \cdot v_x^t, & y < 6mm \\ 0, & y \geq 6mm \end{cases}$$
$$F_y^t = \begin{cases} 0, & y < 6mm \\ a \cdot (y_t - 6) + b \cdot v_y^t, & y \geq 6mm \end{cases}$$

Where a and b are negative numbers, such that for the first 6 mm of forward motion, this produces an instantaneous x force that is proportionate in magnitude and opposite in direction to the mouse's instantaneous distance from the y axis and to the x velocity. When the mouse's forward position reaches 6 mm, the code switches to a different function in which the robot produces an instantaneous y force that is proportionate in magnitude and opposite in direction to the mouse's instantaneous distance from the horizontal line located at y = 6 mm and to the y velocity, while allowing unimpeded motion along that horizontal line located at y = 6 mm.

Thus, these data explicitly demonstrate two examples of position- and velocity-dependent force functions used to produce constrained motion trajectories, updated by the device at 1 kHz. Users can program any other arbitrary position- and velocity-dependent force by the mathematical relations described above.

The user interface for controlling the pattern of applied forces involves specifying different 'trial types' (Supplementary Fig. 1). Each trial type leads to the execution of a different code block in the 'MathScript' node 'computeTorqueAndForce.vi'. These are simple MATLAB-syntax mathematical statements that specify the instantaneous cartesian force output as arbitrary functions position and velocity as described above. The example code that we provide implements three trial types: linear pushing movements (in a virtual linear track), two-part turning motions (in a virtual L-shaped or T-shaped track) in which a forward motion is followed by a lateral motion to the left or right and aiming movements, in which the mouse is able to produce arbitrary two-dimensional trajectories, but is only rewarded for those terminating near a target region of the user's choosing (Supplementary Video 1). Users who desire to implement additional assays can do so by adding additional trial types and writing the equations that specify the force patterns on those trial types in the force computation MathScript node.

Data collection and synchronization

The instantaneous values of all state variables are maintained in memory on the FPGA. The value of each variable is 'sampled' by the RT-PC at 200 Hz, and the resulting data are streamed over USB to a memory buffer on the standard PC. The standard PC software collects all data streams into a single T×N matrix, where T is the total number of time points, and N is the number of data streams (N = 11 in our example code). At either experiment termination or when instructed by the user (via the 'write data' operation described in Fig. 5/Step 39), the standard PC software takes this data matrix and writes it to a plain-text comma-separated value (CSV) file, which can be read by a variety of software, including by directly 'dragging and dropping' the file into MATLAB. Data from an example file can be found in Supplementary Data 1 and are described in Anticipated results.

In our example code, the data matrix contains the following 11 data stream variables in order: *x* and *y* position (mm), commanded current to motors 1 and 2, trial type, desired movement angle (rads), frame counters 1 and 2, lick sensor, reward solenoid and robot state. These are defined in 'DataRecord.ctl' (in the 'typedef controls' folder in Fig. 5b). The frame counters count digital TTL ('transistor-transistor logic') pulses, and if either counter is not wired to an input, that count will simply remain at zero. Thus, when users analyze the resulting data file, cross-referencing a frame counter with the simultaneously acquired behavioral data allows synchronization of different data sources such as microscopes or electrophysiology systems.

Animal training

During behavioral training, animals are water restricted and generally obtain all water during training sessions. Our studies have employed head-fixed behavior. Thus, mice undergo a surgical procedure during which we implant a steel plate over the cranium to permit fixation to external grounding bars[15,16]. During behavioral sessions, animals' bodies are confined to a tube (Fig. 1d), with an opening under the right forelimb (for our studies of right forelimb movement). The robotic manipulandum handle is placed ~2 cm ventral to, and at roughly the same anterior-posterior plane as, the head (Fig. 1e). Placement is a key factor in the early training of animals, and heavier animals (e.g., older males) typically require placement farther from the body than lighter animals (e.g., younger females). In our tasks, successful trial completion is followed by a 1-s delay before reward delivery, followed by another ~3.5-s delay before the robotic handle automatically returns itself to the home position beneath the mouse (Fig. 4a). This basic task structure, and the associated delays, can be customized through the state machine code (Supplementary Fig. 1c).

For most tasks, we first employ an initial training phase during which animals can only execute forward-directed linear movements in a virtual track (Fig. 4b). In our experience, this simple task facilitates the learning of the basic task structure, timing and reward association. We typically continue with this initial phase until mice can execute ~150–200 trials in a single ~25-min session (typically ~3–5 d).

Depending on the task under study, subsequent training can proceed in different ways. For the alternating left-right movement sequence task (Fig. 4c), in early training, mice experience two different L-shaped virtual tracks ('Max error' of 0 mm in Fig. 5d). This helps animals to gain experience in the kinematic patterns of the two movement types. In later training (after ~1 week), we transition to T-shaped tracks ('Max error' of, e.g., 3 mm). In this case, animals have the option of initiating either of the two movement trajectories. If they move in the wrong direction, however, no reward is delivered. In our experience, this helps to drive learning to distinguish the two trial types for the animal (a process that takes an additional 1–2 weeks). Alternatively, we have studied aiming movements (Fig. 4f,i). In this case, early in training, movements that terminate relatively far from the target still yield reward (large 'Max error' in Fig. 5d). As animals begin to better target their movements, we progressively tighten the reward contingency (smaller 'Max error'). This encourages further performance improvements, with a total training time of 1–2 weeks.

Materials

Mice

We have trained a variety of wild-type (C57/bl6, CD1) and transgenic mice (e.g., *GAD2-Cre/lox-stop-lox-ChR2* and *Math1-Cre/Rbp4-Cre/CAG-lox-stop-lox-tTA/TRE-lox-stop-lox-GCaMP6f*) on these tasks without finding clear differences in their ability to learn these behaviors. Mice of both sexes perform equally well, and all age ranges tested (~6 weeks to ~8 months) can learn these tasks **!CAUTION** Any experiments involving live mice must conform to relevant institutional and national regulations. All procedures followed animal care and biosafety guidelines approved by Stanford University's Administrative Panel on Laboratory Animal Care and Administrative Panel on Biosafety in accordance with NIH guidelines.

Example dataset

We have provided the raw data in Supplementary Data 1 for the data set that gave rise to Fig. 4f and the rightmost panel of Fig. 4g. This contains the time-varying x and y position, the solenoid open state, the time-varying robot state and the lick sensor value for this experiment.

Equipment

Manipulandum

- Plastic linkages and handle (3D printed from CAD files provided in Supplementary Data 2)
- 2-mm diameter stainless steel shaft (McMaster #1174K19), cut to 12-mm length for each of the two elbow joints and 22-mm length (or as desired) for the wrist joint/handle
- 2-mm diameter bearings × 5 mm outer × 2 mm height (quantity: six) (McMaster 7804K119)
- M2 × 0.4 mm, 6-mm length button head screws (quantity: two) (McMaster 91306A653)
- M2 × 0.4-mm low-profile nuts (quantity: two) (McMaster 90710A020)
- Flat M2 screws (quantity: six) for affixing the motors to the bracket (McMaster 91294A004)
- M3 screws (quantity: three) for motion limiters (McMaster 91292A109)
- Integrated motor + encoder units (quantity: two) (Maxon B7A1F24007CF, containing DCX22S EB KL

  24V motor with ENX 16 RIO 65536IMP encoder)

Electronics

- NI cRIO chassis and RT-PC (cRIO-9053, #786424–01)

- NI 9505 (#779126–01) DC brushed servo drive and encoder reader (quantity: two)
- NI 9403 (#779787–01) 32-channel digital I/O, with NI 9923 (781503–01) terminal block
- Power supplies for each of the two motor drivers (24 V, >2.5 A) and for the NI cRIO chassis (NI PS-15 or 24 V/5 A) **!CAUTION** Care should always be exercised when handling power supplies that deliver high currents. Always disconnect all power before modifying wiring connections.

Software

- National Instruments LabVIEW programing environment (tested on LabVIEW 2019 32-bit); packages: Mathscript/Mathscript Real-Time, LabVIEW FPGA and compactRIO
- National Instruments Linux real-time OS; packages: Network Streams
- Custom software: RobotNP LabVIEW project, including the Windows software user interface (UI) Main.vi, the RTOS software RTMain.vi and the FGPA software FPGA Main.vi (https://GitHub.com/mjwagner/haptic-for-mice)

Equipment setup

Minimum equipment requirements—For this device, the primary cRIO performance considerations are the available FPGA space and the RT-PC processing speed. At the time of writing, the cRIO-9053 is the most inexpensive cRIO system. On this minimal system, our code uses only a minority of the FPGA space, and the RT-PC reliably executes at the specified 1 kHz. Other cRIO systems, which are more expensive and feature higher specifications, should therefore also perform adequately. In addition, we have successfully tested similar versions of our code on the now-discontinued cRIO-9114, which has roughly one-fourth of the processing power of the cRIO-9053.

At the time of writing, there are no alternative cRIO motor control modules to the 9505. There are alternative modules that provide functionality similar to the 9403 module that we use. For our example code, all that is required is a module providing at least five bidirectional digital I/O channels, but researchers with differing digital I/O needs may consider other modules. In addition, some cRIO modules provide analog I/O, which may be useful for some studies.

We recommend using the specific DC motors described in this protocol. In principle, other customizations of the Maxon DCX22/ENX16RIO motor/encoders that have identical physical casing and form factor, and the same encoder resolution, as the part numbers that we recommend would be physically compatible with our design. At the time of writing, these motors provide the maximal torque in this form factor. If an experimenter deems such torques inadequate (e.g., for larger animals), a different form factor and therefore a redesign of the device would be necessary. Alternatively, if users desire lower peak torque motors,

they would need to ensure that any alternative is sufficient to counter the forces that mice typically produce during experiments, which we have not tested.

The standard PC code does not execute demanding computations or highly time-sensitive operations, and thus the requirements of the computer are minimal. We have tested our standard PC code on a range of Windows PCs, including an ~10-year old machine based on Pentium IV technology, without encountering limitations. LabVIEW is platform independent and is available for MacOS; however, we have not explicitly tested device control using MacOS, and users would need to confirm proper functionality.

Finally, while we have not extensively tested 3D-printing options, we have produced devices that operate for years using either the relatively higher-end Protolabs 3D-printing service recommended in this protocol or the inexpensive entry-level Stratasys OBJET30 standalone printer. Thus, we expect a wide range of printing materials and dimension tolerances to produce a properly functioning device.

Procedure

Manipulandum assembly

1 Use a 3D printer or 3D-printing service to produce the five plastic pieces from the provided CAD files (Supplementary Data 2): two upper arm linkages, two lower arm linkages and a handle. Optionally, also print the mouse body-restraining tube. We print the linkages and handle using ProtoLabs material ‘RenShape SL 7820 High-Resolution Stereolithography build in 0.002’ and ‘High-resolution’. The restraining tube can be printed from the same stock, or in cases where a transparent tube is needed to visualize the animal’s body, we use the ‘Watershed XC 11122’ material, specifying ‘transparent finish’ in the notes section.

2 Use an 8–32 screw with a nut to affix the restraint tube from the side to Thorlabs aluminum post hardware.

3 Machine the mounting bracket in the provided CAD file (Supplementary Data 2) from aluminum stock, placing M3 taps in the three corresponding holes in the bracket. We use Protolabs CNC milling service from aluminum, specifying the three M3 holes to be tapped.

4 From the 2-mm stainless steel shaft stock, cut three pieces: two 12-mm pieces for the elbow joints and one 22-mm piece (or the length desired for the handle that extends to the animal) for the wrist joint and handle shaft. We typically use a Dremel for this step. Lightly grinding the corner can help with insertion.

5 Insert two of the 2 mm ball bearings into the pockets in the end of the female lower arm linkage (Fig. 2a). Insert the remaining lower arm linkage between the bearings (Fig. 2b), forming the wrist joint.

6 Insert two of the 2-mm ball bearings into the pockets in the end of each upper arm linkage (Fig. 2c). Insert the loose ends of the lower arm linkage assembly

between the bearings in each upper arm linkage (Fig. 2d), forming the two elbow joints.

7 Insert button head M2 screws into the loose end of the upper arm linkages and secure into thin hex nuts (Fig. 2e).

8 Insert 12-mm shafts into each elbow joint and the 22-mm shaft into the wrist joint and mount the plastic handle onto the wrist shaft (Fig. 2f).

9 Affix each motor to the mounting bracket via the three flat M2 screws (Fig. 2g).

10 Insert the M3 screws into the three tapped holes (Fig. 2g).

11 Insert the exposed motor shafts into the holes at the loose ends of the upper arm linkages, forming the shoulder joints. Secure the joints by tightening the M2 button head screws (Fig. 2h).

NI installation

12 Per the NI guides, install the cRIO, placing the two NI 9505 modules in Slots 1 and 2 and the NI 9403 module in Slot 4 (Fig. 3e). Connect the cRIO power leads (Fig. 3d,e).

13 Install the LabVIEW environment (including LabVIEW FPGA, LabVIEW Real-time, MathScript Real-time and compactRIO packages) on a standard PC and connect to the cRIO via USB (Fig. 3d,e).

14 After installation, follow the NI instructions for using NI device manager to confirm communication between the PC and the cRIO.

15 LabVIEW software is also required on the cRIO RT-PC, including NI packages: compactRIO, LabVIEW Real-time and Network Streams. Follow the NI instructions for installing these through NI-MAX.

Electronics wiring

16 Note the following convention: we term the two motor/encoder units ‘left’ and ‘right’ as shown in Fig. 5a. Connect the NI 9505 in Slot 1 to the left unit, and connect the 9505 in Slot 2 to the right motor/encoder unit. All software labels refer to the left unit as ‘1’ and the right unit as ‘2’.

17 Wire the encoders to the NI 9505 modules: following the pin-out of the Maxon encoder (https://www.maxongroup.com/medias/sys_master/8827190738974.pdf, p.12) and the pin-out of the NI 9505 encoder reader (http://www.ni.com/pdf/manuals/374211h.pdf, p.17), match the Maxon outputs to the 9505 inputs (e.g., solder the Maxon wires to a DB9 connector, to match the DB9 input on the NI 9505).

18 For each motor/NI 9505 module pair, connect the two leads from the motor to the two motor driver outputs on the NI 9505 (Fig. 3d,e). Follow the convention: red motor lead to the terminal labeled ‘−’.

19 Connect power leads to both 9505 modules (Fig. 3d,e).

!CAUTION Do not remove or insert motor lead wires when the power supplies to the 9505 modules are powered on.

20 Connect digital inputs and outputs to the NI 9403 (Fig. 3d,e). Our code provides inputs (Pin 9 and 10, Frame counters 1 and 2 (pulse counter); Pin 12, touch/lick sensor) and outputs (Pin 6, solenoid trigger for water reward delivery (timed digital pulse)).

FPGA software setup and first-time device calibration

21 Open the provided LabVIEW project 'Robot_NP.lvproj' (a project explorer will appear as in Fig. 5b).

22 Confirm that the LabVIEW project is communicating with the cRIO device by right-clicking the device in the project explorer (Fig. 5b) and clicking connect. The green circle should become bright.

▲**CRITICAL STEP** If the green circle does not become bright, exit LabVIEW and return to NI-MAX to follow the NI instructions for diagnosing communication issues.

23 Perform initial device calibration through the FPGA front panel by opening 'FPGA main.vi' (Fig. 5b), which will open a panel shown in Fig. 5c. Run this program (it may compile first; ~15 min).

24 Confirm that the encoders are read correctly. For this test, it is easier to remove the plastic linkages from the motors (undo Step 11). Manually turn the left motor shaft clockwise while observing the number in 'Encoder 1 Position' (Fig. 5c under 'Position Controllers'). The number should decrease if the channels have been correctly wired in Step 17. If Encoder 2 Position changes instead, check the left/right motor convention in Step 16. Repeat for the right motor shaft/'Encoder 2 Position'.

25 Next, confirm that the FPGA can drive the motors. First click 'Enable Drives' in the 'Drive Status' box on the FPGA panel (Fig. 5c). Next, in the 'PWM Generators' box on the FPGA panel (Fig. 5c), make sure that 0 is entered into both 'PWM 1' and 'PWM 2.' Next, switch on 'Manual PWM.' Now, while observing the left motor shaft, type '100' into the 'PWM 1' box and confirm that the left motor shaft is rotating counter-clockwise. If it rotates clockwise, check the motor lead convention in Step 18. Repeat for the right motor/'PWM 2'. If both are functioning, switch off 'Manual PWM'.

26 Reattach the manipulandum linkages to the motors (as described in Step 11).

27 Confirm that the torque controllers are able to control the motor current. First, ensure that the 'Desired Current 1' and 'Desired Current 2' are both set to 0. Next, ensure that both 'Position/Current mode 1' and 'Position/Current mode 2' are switched on and then toggle on both the 'Start new mode 1' and 'Start new mode 2'. Next enter '20' into 'Desired Current 1'. The handle will move.

Confirm that the ‘Actual Current 1’ reading is close to 20. Set ‘Desired Current 1’ back to 0 and repeat for ‘Desired Current 2’.

28 Perform the final FPGA calibration step, which is to confirm that the Position Controllers operate correctly. First, switch off ‘Position/Current mode 1’ and ‘Position/Current mode 2’. Then, toggle on ‘Start new mode 1’ and ‘Start new mode 2’. The handle should lock in place—press it lightly and check for resistance. Next, check for controller instability—lightly tap the handle. It should resist motion and remain near the home position. Finally, close ‘FPGA main.vi’. This will not typically be directly operated during normal device usage.

▲**CRITICAL STEP** In the unlikely event that the device oscillates unstably, press ‘Disable Drives’ under ‘Drive Status’. Return the motors to current mode with ‘Desired Current 0’ (Step 27). The controller needs to be calibrated.

? TROUBLESHOOTING

Behavioral software operation

29 Open ‘UI Main.vi’ and ‘RT Main.vi’ (Fig. 5b). RT Main produces an empty panel, while UI Main opens the primary user interface front panel (see Fig. 5d for a full description of all controls).

30 Press ‘enable drives’ (Fig. 5d) to activate the drive control in both NI 9505 modules.

31 Press ‘Find home’ (Fig. 5d) to instruct the robot to search for the M3 mechanical stops and to move to the home position, where it will remain locked until instructed otherwise.

32 Select the ‘Trial list file’ folder icon button (Fig. 5d) to choose a ‘trial list’, in which each line specifies the parameters for a single trial. These parameters are listed in order on the front panel in the ‘current trial parameters’ cluster at the bottom of the window.

33 When setting up for the first time, choose the provided file ‘2D_0deg.csv’. This executes aiming trials with a target 8 mm ahead (Fig. 4f).

34 Press ‘Start block’. A trial will be read in from the trial list, and the appropriate parameters will update.

▲**CRITICAL STEP** The robot will release control of the handle—but only when it has detected that no external force has been applied for the preceding 100 ms. At this point, the software waits until it senses handle movement, and then either waits for trial completion (either successfully or incorrectly) or terminates an incomplete trial after the ‘timeout’ period elapses.

35 Manually push the handle forward. When it has moved ≥8 mm radial distance, the robot will lock it in place to terminate the trial.

36 For movements sufficiently close to straight ahead (‘Max Error’ trial parameter), the robot will deliver reward. To confirm, in the ‘Movement criteria’ box, ‘Last trial rewarded’ should be highlighted after a straight movement or not highlighted after a mistargeted movement off to the side. Check that highlighting takes place after correctly targeted movements.

37 Confirm that after trial termination and a delay, the software successfully returns the handle to the start position and advances to the next trial in the list.

Constrained trajectory trials and first-time calibration

38 Repeat from Step 33, this time loading ‘linear_0deg.csv’. This will load linearly constrained trials (Fig. 4b). Trajectories are constrained in the RT Main software, as described above. Variations in device assembly could result in different mechanical properties that require different controller parameters. To confirm, manually execute movements. The handle should feel constrained to linear motion. This completes all first-time calibration procedures and confirms that the robot is functioning correctly.

▲**CRITICAL STEP** In the unlikely event that unstable oscillations are felt, press, ‘STOP MOTORS’. The controller must be tuned.

? TROUBLESHOOTING

39 Experiment termination: after an experiment, prevent loading and initiating additional trials by pressing the ‘Stop Block’ button. At this point, all data remain in memory. Use the ‘write data’ button to empty the memory buffer to a data file on a hard disk. Once ready, press the ‘EXIT’ button to exit the program and write any remaining data in memory to disk.

Animal training

40 Acclimate mice. We acclimate our mice to ~1 ml of water per day for several days prior to training. In our experience, after 2 d of restricted water access, mice will begin performing the task; thus, this is the fastest way to begin training. However, motivation levels and thus basic task-learning rates are often higher after ~4–5 d of restricted water access, in circumstances where additional time is available.

41 Prior to a behavioral session, prepare the robot by homing the device, loading the desired trial list and confirming water delivery, so that the training session can begin as soon as the animal is head-fixed, to minimize the time the animal spends waiting on the rig.

42 Head-fix the animal by holding it by the tail with its head and forepaws directly behind the tube, which mice will naturally walk into. Then, grasp the head-fixation bar on the cranium from the front to secure it to mounting bars.

? TROUBLESHOOTING

43 Place the water reward spout just in front of the mouth (~1 mm below and anterior to the mouth), without touching the animal's face.

? TROUBLESHOOTING

44 Place the manipulandum handle near the mouse. The software provides the ability to make small adjustments to the 'home position' of the handle (Fig. 5d). For larger adjustments, physically move the robot mounting base.

▲**CRITICAL STEP** In the earliest training sessions, it can be helpful to place the manipulandum handle significantly closer to the mouse's body, to maximize the odds that the animal contacts it with the right forelimb by chance. As animals become proficient, it is more helpful to give more space between the body and the handle to make it easier to move the limb.

? TROUBLESHOOTING

45 For all tasks, begin training using forward-constrained movement tasks (example file: 'linear_0deg. csv'). As described in Step 34, the only user step required to initiate a training session is to click 'Start Block', after which animals will be able to self-initiate and self-pace each trial in the sequence specified in the trial list that the user loaded, until the user terminates the block as described in Step 39. Additional ways to make the task easier early on include: using a lower 'Rewarded length', which we typically set to 7 mm in expert mice, but can be lowered to 6 mm initially; using a longer 'Timeout' period, which we typically set to 100 in expert mice (note: 'Timeout' of '0' corresponds to no timeout and automatically toggles on the 'wait for completion' switch).

? TROUBLESHOOTING

▲**CRITICAL STEP** In this and the next step, the experimenter must choose when to advance the animal from easier task variations to more challenging variations. In general, we find that there is a tradeoff between permitting the animal to achieve greater proficiency on the simpler tasks and the subsequent difficulty of training the animal to 'break' its earlier habits to learn another task variation. Our general recommendation is to look for 2–3 d of consistent performance of ~150 trials in ~25 min on an easier task before advancing to a more complex task version.

? TROUBLESHOOTING

46 If appropriate, apply additional training procedures. For the aiming task, early on, allow rewards for substantially mistargeted movements via a larger 'Max Error' (defines the maximum lateral distance from the target direction vector at 7-mm radial distance from the start position). Expert mice should be able to perform the task at 2-mm or 2.5-mm max error. This task should take ~1–2 weeks to train. For the two-part turning movement sequence task, early on, deliver L-shaped constrained trajectories by using 'Max Error' of 0 mm. After several days, provide T-shaped trajectories by using 'Max Error' of, e.g., 3 mm. This task should take ~2–3 weeks to train. In all tasks, expert animals should be

able to perform on average ~150–250 trials in an ~25-min behavioral session, and animals should be trained once a day until proficiency (depending on task).

? TROUBLESHOOTING

Troubleshooting

For Step 28: FPGA position controller oscillation: in the unlikely event that the position controllers are unstable or oscillate (Step 28), the proportional-integral-derivative (PID) controller parameters need to be tuned. With the motors in 'Current mode' with 'Desired current' set to 0 (Step 27), first enter '0' for the derivative constant, Dpos. Next, begin by setting the proportional constant, Ppos, to 1. Then, enable 'Position mode' (Step 28). The device should not oscillate. Next, begin slowly increasing Ppos (e.g., by 0.5 increments) to a level just below where instability appears. Now, begin incrementing Dpos (e.g., by increments of 5) to a level just below when instability appears. Repeat this procedure (incrementing Ppos and then incrementing Dpos) until the handle is stably held in place.

For Step 38: constrained trajectory controller oscillation: in the unlikely event that the constrained trajectories are unstable or oscillate (Step 38), the PID parameters must be tuned ('Robot parameters'; P, I and D). With the robot disabled after pressing 'Stop block', set P to 100 and D to 0. Click 'Start block' to initiate trials. Confirm that there is no oscillation. Repeat after incrementally increasing P to a level just below where instability appears. Then, increase D to a level just below where instability appears. Repeat these alternating adjustments of P and D until trajectories are constrained stably.

See Table 1 for troubleshooting guidance for behavioral issues.

Timing

The custom 3D-printed pieces as well as the custom-machined flange mount can be obtained from Protolabs in 2–3 business days typically, although it is also possible to specify faster production. Production time via, e.g., in-house CNC milling or 3D printing, will vary based on the specific devices used. The products from McMaster are typically available on the same day, while the National Instruments products generally have a 2–6-week lead time, and the motor-encoder units from Maxon typically have a 4–6-week lead time.

Robot assembly (Steps 1–11) should take several hours. Electronics assembly and wiring (Steps 12–20) should also take several hours. Software installation, setup and calibration (Steps 21–39) should take less than a day. Animal training (Steps 40–46) takes between 1 and 3 weeks, depending on the task.

Anticipated results

Example results from four different tasks are shown in Fig. 4b–i. We provide trajectories from the simplest task, in which the animal's motion is linearly constrained by a virtual track, in this case directed straight ahead, as in ref. [16]. We also provide example trajectories from a task where the animal's movement trajectory is constrained to a T-shaped track, such that it must push forward and subsequently push to the left or right, as in ref. [15]. For this

turning movement sequence task, we show trajectories in which the animal made the correct motion, and we summarize the improvement in performance across 10 mice over several weeks in Fig. 3d,e.

We also provide example data from the aiming task, in which the animal can produce arbitrary two-dimensional trajectories but is rewarded only for those sufficiently close to a user-specified target direction (Fig. 4f). To provide a sense of the across-trial variability, we provide dot plots of the mouse's maximal deviation in millimeters from the target movement axis during each trial, which we display for three mice (three left-most panels in Fig. 4g). To provide an illustration of across-day variability in an expert mouse, we provide performance dot plots for three consecutive days in one mouse (Fig. 4g, three right-most panels). The performance level across all trials from eight different mice is quantified in the histogram in Fig. 4h. The possibility of training animals to alternate between three different target movement directions in the aiming task is illustrated in Fig. 4i.

To give users a sense of the type of raw data to expect, we provide an example MATLAB file in Supplementary Data 1 that contains the x and y position, licking, solenoid opening and robot state data for the experiment displayed in Fig. 4f and the rightmost dot plot in Fig. 4g.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

M.J.W. is supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund. M.J.S. and L.L. are HHMI investigators. This work was supported by NIH and NSF grants.

References

1. Shadmehr R. & Mussa-Ivaldi FA Adaptive representation of dynamics during learning of a motor task. J. Neurosci. 14, 3208–3224 (1994). [PubMed: 8182467]
2. Shadmehr R, Smith MA & Krakauer JW Error correction, sensory prediction, and adaptation in motorcontrol. Annu. Rev. Neurosci. 33, 89–108 (2010). [PubMed: 20367317]
3. Smith MA & Shadmehr R. Intact ability to learn internal models of arm dynamics in Huntington's disease but not cerebellar degeneration. J. Neurophysiol. 93, 2809–2821 (2005). [PubMed: 15625094]
4. Pasalar S, Roitman AV, Durfee WK & Ebner TJ Force field effects on cerebellar Purkinje cell discharge with implications for internal models. Nat. Neurosci. 9, 1404–1411 (2006). [PubMed: 17028585]
5. Padoa-Schioppa C. Neurobiology of economic choice: a good-based model. Annu. Rev. Neurosci. 34, 333–359 (2011). [PubMed: 21456961]
6. Mathis MW, Mathis A. & Uchida N. Somatosensory cortex plays an essential role in forelimb motoradaptation in mice. Neuron 93, 1493–1503.e6 (2017). [PubMed: 28334611]
7. Komiyama T. et al. Learning-related fine-scale specificity imaged in motor cortex circuits of behaving mice. Nature 464, 1182–1186 (2010). [PubMed: 20376005]
8. Tennant KA et al. The organization of the forelimb representation of the C57BL/6 mouse motor cortex asdefined by intracortical microstimulation and cytoarchitecture. Cereb. Cortex 21, 865–876 (2011). [PubMed: 20739477]

9. Azim E, Jiang J, Alstermark B. & Jessell TM Skilled reaching relies on a V2a propriospinal internal copy circuit. Nature 508, 357–363 (2014). [PubMed: 24487617]
10. Guo JZ et al. Cortex commands the performance of skilled movement. Elife 4, e10774 (2015).
11. Chen CC, Gilmore A. & Zuo Y. Study motor skill learning by single-pellet reaching tasks in mice. J. Vis. Exp. 85, e51238 (2014).
12. Lambercy O. et al. Sub-processes of motor learning revealed by a robotic manipulandum for rodents. Behav. Brain Res. 278, 569–576 (2015). [PubMed: 25446755]
13. Vigaru B. et al. A small-scale robotic manipulandum for motor training in stroke rats. in 2011 IEEE International Conference on Rehabilitation Robotics 1–7 (IEEE, 2011). 10.1109/ICORR.2011.5975349
14. Vigaru BC et al. A robotic platform to assess, guide and perturb rat forelimb movements. IEEE Trans. Neural Syst. Rehabil. Eng. 21, 796–805 (2013). [PubMed: 23335672]
15. Wagner MJ et al. Shared cortex-cerebellum dynamics in the execution and learning of a motor task. Cell 177, 669–682 (2019). [PubMed: 30929904]
16. Wagner MJ, Kim TH, Savall J, Schnitzer MJ & Luo L. Cerebellar granule cells encode the expectation of reward. Nature 544, 96–100 (2017). [PubMed: 28321129]
17. Campion G. The pantograph MK-II: a haptic instrument in The Synthesis of Three Dimensional Haptic Textures: Geometry, Control, and Psychophysics 45–58 (Springer, 2011).
18. Wagner MJ & Smith MA Shared internal models for feedforward and feedback control. J. Neurosci. 28, 10663–10673 (2008). [PubMed: 18923042]
19. Flash T. & Hogan N. The coordination of arm movements: an experimentally confirmed mathematical model. J. Neurosci. 5, 1688–1703 (1985). [PubMed: 4020415]

Box 1 |

Glossary of controls in the user interface software

- ‘Start/Stop block’ – initiates trials from the current trial list file
- ‘Enable drives’ – activates drive control modules
- ‘Trial list file’ – allows selection of a list of trial parameters
- ‘Find home’ – instructs the robot to execute a series of motions to locate the home position
- ‘Go home’ – sends the handle from the present position to the home position
- ‘X home’/’Y home’ – this sets the coordinates of the home position (default 0, 57), and relative to the default, has an approximate range of ±1.5 mm in the y direction and ±3 mm in the x direction.
- ‘Only advance after reward’ – determines whether to repeat the same trial after a failed attempt
- ‘Wait for completion’ – if enabled, robot will not terminate trial (equivalent to timeout of ‘0’).
- ‘Hold position’ – locks robot at current position
- ‘Robot parameters’ – P, I and D specify the force parameters used for constrained trajectories.
- ‘Go to target’ – instructs the robot to move to the position specified in ‘X/Y target’
- ‘STOP MOTORS’ – disables robot
- ‘Rewarded length’ – sets the minimum movement length that yields reward (separate from and in addition to other potential reward criteria)
- ‘Viscosity’ (in Robot parameters box) – applies artificial friction to slow a mouse’s movements, if desired. A typical value is 0.4, and 0 disables this.
- ‘Reward’ – sets the duration of the TTL pulse, to control the duration of solenoid opening
- ‘Give reward’ – immediately delivers the solenoid TTL pulse
- ‘Data record file’ – specifies where to write data
- ‘Write data’ – immediately empties data in the memory buffer to disk
- ‘EXIT’ – first writes data in memory to disk and then terminates the program (preferred method to exit)

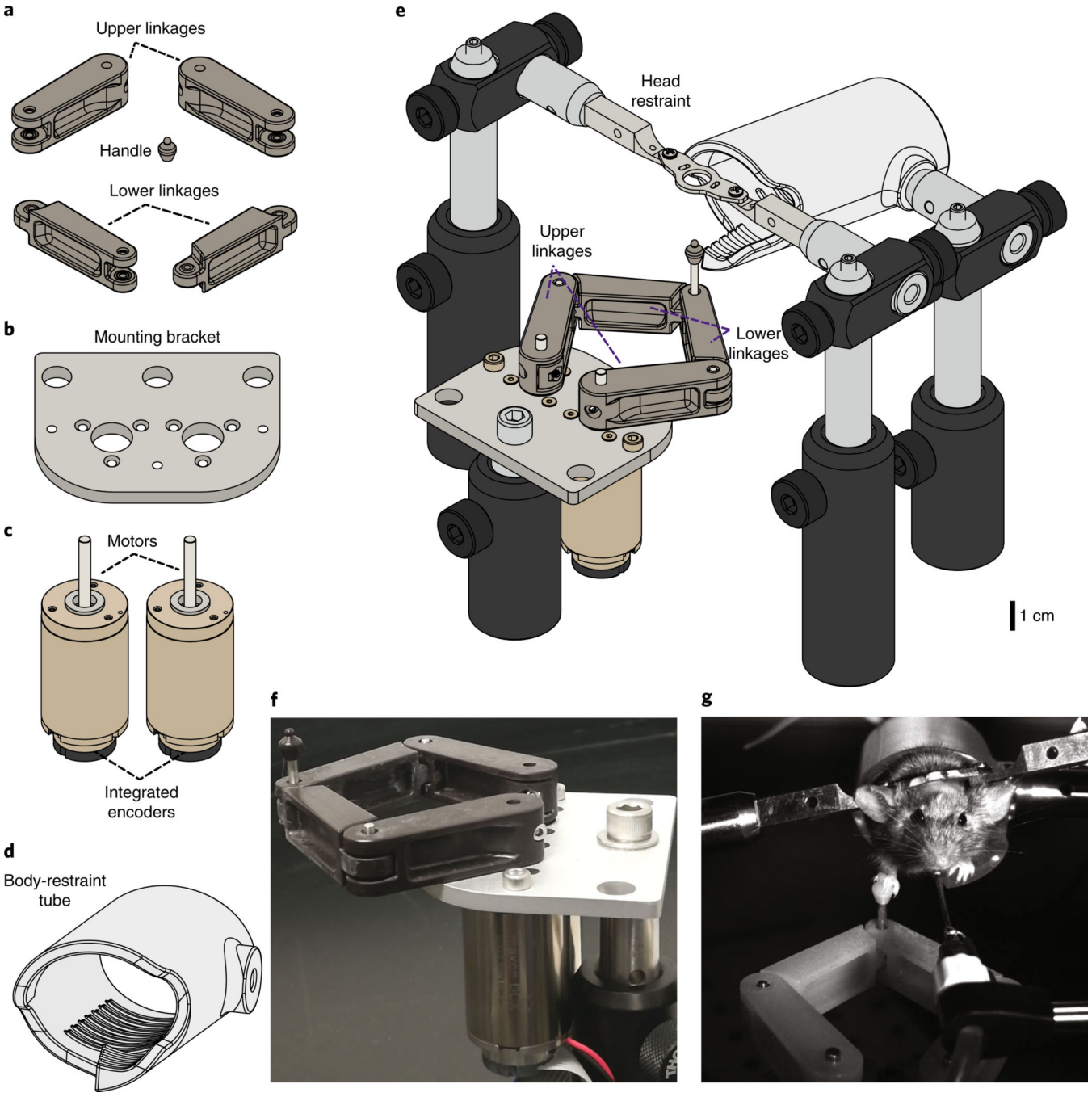

Fig. 1 |. Actuated two-axis rodent manipulandum design and assembly.

a-d, Mechanical drawings of apparatus parts. **a,** Custom 3D-printed plastic manipulandum linkages and handle for the mouse forepaw. **b,** Custom aluminum machined mounting bracket to mount the motors. **c,** DC Motors with integrated high-resolution optical rotary encoders (Maxon Motors). **d,** Custom 3D-printed transparent plastic restraining tube for the mouse to stand inside. **e,** Complete assembly of behavioral apparatus including head restraint bars and Thorlabs mounting hardware. See Fig. 2 for assembly details. Full 3D rendering of assembly is available at https://grabcad.com/library/full-protocol-2. **f,**

Photograph of assembled manipulandum. **g,** Still image from a video of a mouse executing a skilled forelimb task while head fixed and body restrained by grasping and moving the manipulandum handle for water reward. The procedure used on the mouse followed animal care and biosafety guidelines approved by Stanford University's Administrative Panel on Laboratory Animal Care and Administrative Panel on Biosafety in accordance with NIH guidelines.

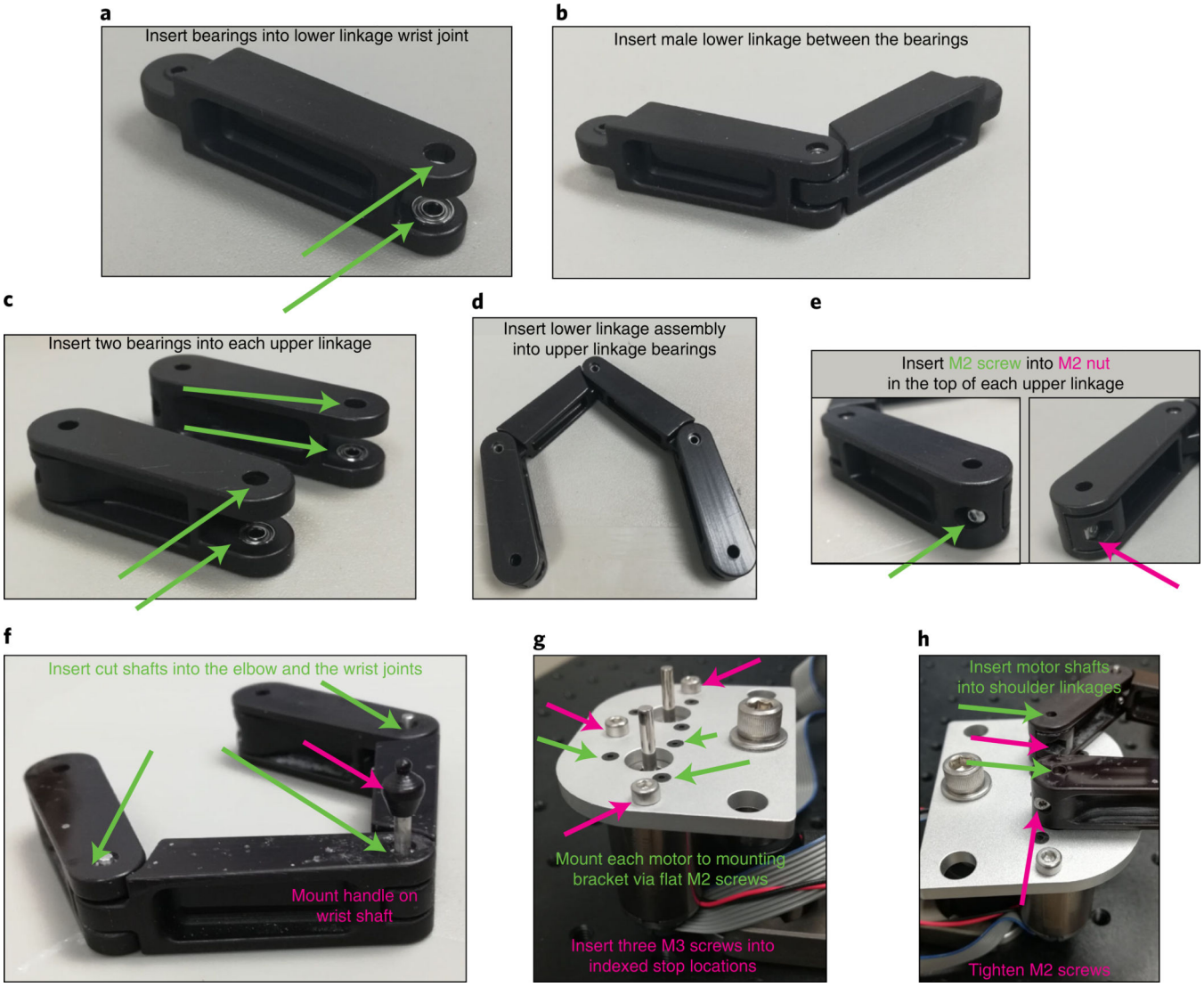

Fig. 2 |. Detailed manipulandum assembly.

a, For the lower linkage with pockets at one end, insert a ball bearing into each pocket, making sure to press firmly to insert fully. **b,** Insert the small end of the remaining lower linkage between the two bearings. When inserted correctly, the joint should rotate freely. **c,** For each (identical) upper linkage, insert a ball bearing into each pocket, making sure to insert fully. **d,** Take the free ends of the combined lower linkages and insert between the upper linkage ball bearings. **e,** Insert a button head M2 screw (from the outer side of linkage) into a thin M2 hex nut (pressed into the pocket in the inner side of the linkage) at the free end of each upper linkage. **f,** Insert one of the 12-mm cut shafts into each elbow joint and the 22-mm cut shaft into the wrist joint. Press the plastic handle onto the top of the wrist joint shaft. **g,** Affix each motor to the bottom of the mounting bracket by matching the tapped M2 holes on the motor to the M2 bores on the bracket and inserting a flat M2 screw into each. Also insert M3 screws into each tapped hole in the bracket to serve as the indexed mechanical stops for the manipulandum workspace. **h,** Press the holes of the free end of the

upper linkages onto the motor shafts. Secure each upper linkage by tightening the button head M2 screw.

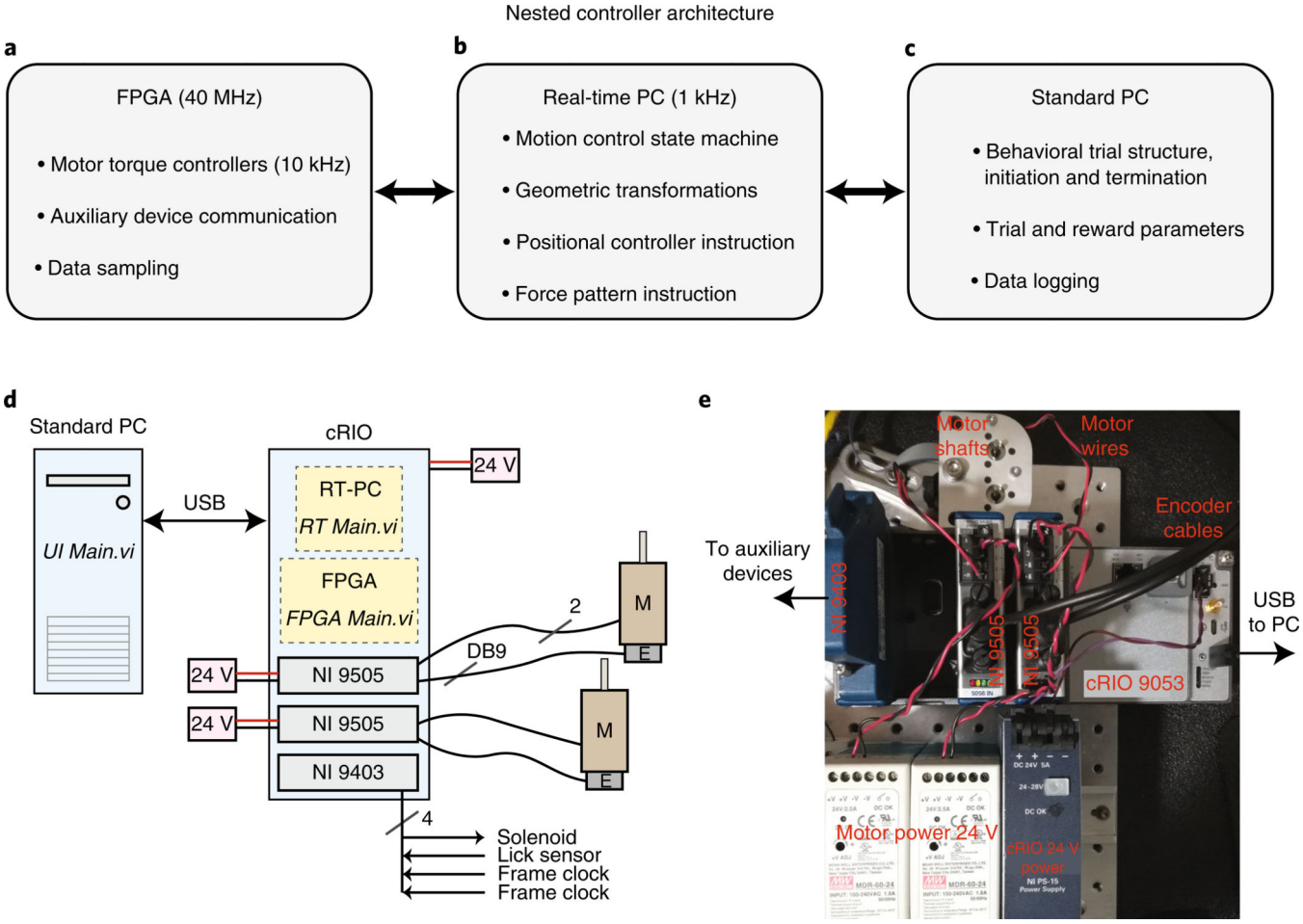

Fig. 3 |. Nested controllers used to operate the manipulandum behavioral experiments.

a, Low-level control and device communication are executed through the FPGA (see also Supplementary Fig. 1a). **b,** The RT-PC executes the bulk of the computationally intensive calculations at 1 kHz needed for real-time device control, including all custom force production and constrained trajectories (see also Supplementary Fig. 1b). **c,** The standard PC runs the user interface panel and executes high-level trial structure state transitions, such as loading trial parameters for each trial, instructing the RT-PC to execute the trial, delivering reward to the animal and logging data to the hard disk (see also Supplementary Fig. 1c). **d,** Wiring diagram. The standard PC, which runs the software 'UI Main.vi', communicates over USB to the cRIO. The cRIO requires power input (24 V, 5A; NI PS-15 shown in **e**). The cRIO houses a real-time PC, which runs the software 'RT Main.vi', and an FPGA that runs the software 'FPGA Main.vi'. It also houses the three modules. Each 9505 module needs power input (24 V, >2.5 A) and communicates with a motor ('M') via two wires and with the integrated encoder ('E') via the DB9 input. The NI 9403 sends digital output to a reward delivery solenoid and also receives and samples input from a lick sensor and two frame clocks. **e,** Photograph of actual device wiring following schematic in **d.**

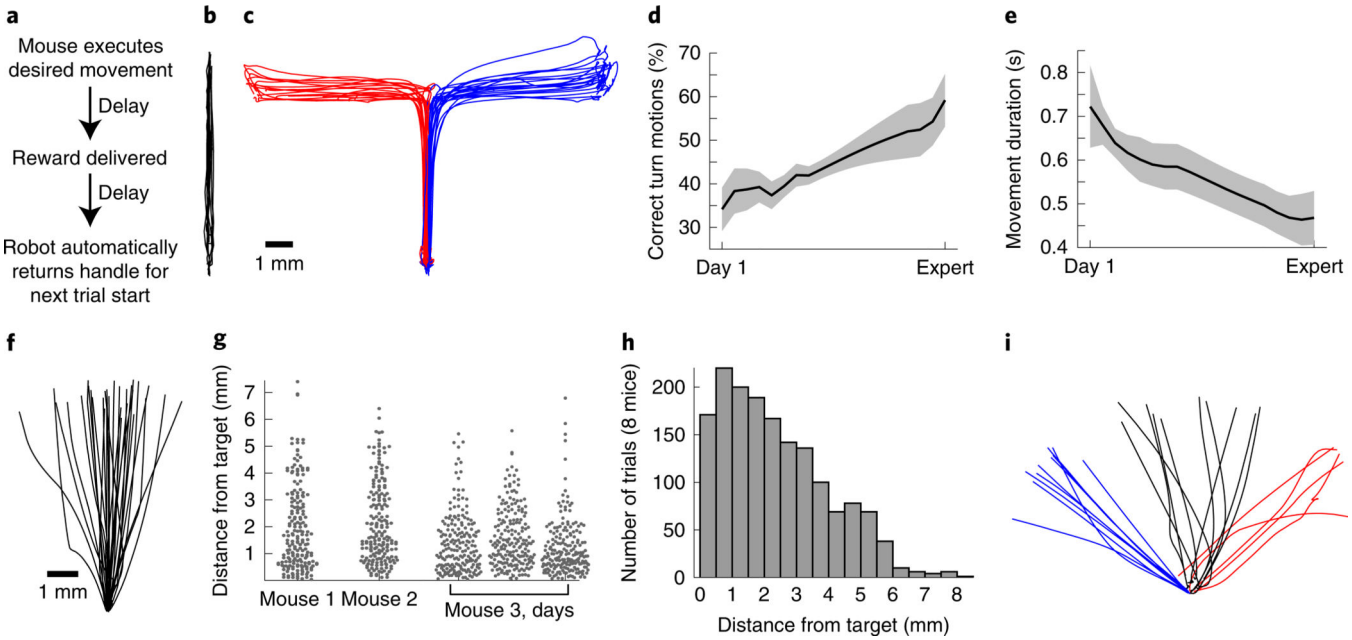

Fig. 4 |. Example behavioral tasks, performance and learning.

a, General trial structure in our behavioral tasks. **b,** Example movement trajectories from a mouse executing the forward pushing task in a virtual linear track. **c–e,** For the forward-turning movement sequence task, example trajectories (**c**) and learning over time (**d** and **e**). Mice nearly double the fraction of forward-turning movements that they execute correctly over learning (**d**; $n = 7$ mice, $p = 0.003$), and decrease the time taken to successfully complete each movement (**e**; $p < 10^{-6}$; 19.3 ± 1 d between Day 1 and Expert, mean ± s.e.m.; Wilcoxon rank-sum tests comparing Day 1 to all Expert days). Adapted with permission from ref. [15]. **f–h**, Aiming movement task, in which reward is delivered only for trajectories directed sufficiently close to a target region. **f,** Example trajectories from one mouse. **g,** Dot plots showing variability in performance (measured as the maximum lateral deviation from the axis connecting the home and target positions) across all trials for three single mice, and across three consecutive days for one mouse. **h,** Performance summarized across all trials from 8 mice on one day. **i,** Example trajectories for a multiple-target aiming task. Error bars denote mean ± S.E.M. All procedures used to obtain these results followed animal care and biosafety guidelines approved by Stanford University's Administrative Panel on Laboratory Animal Care and Administrative Panel on Biosafety in accordance with NIH guidelines. Mice were mixed background male and female animals aged 6–16 weeks.

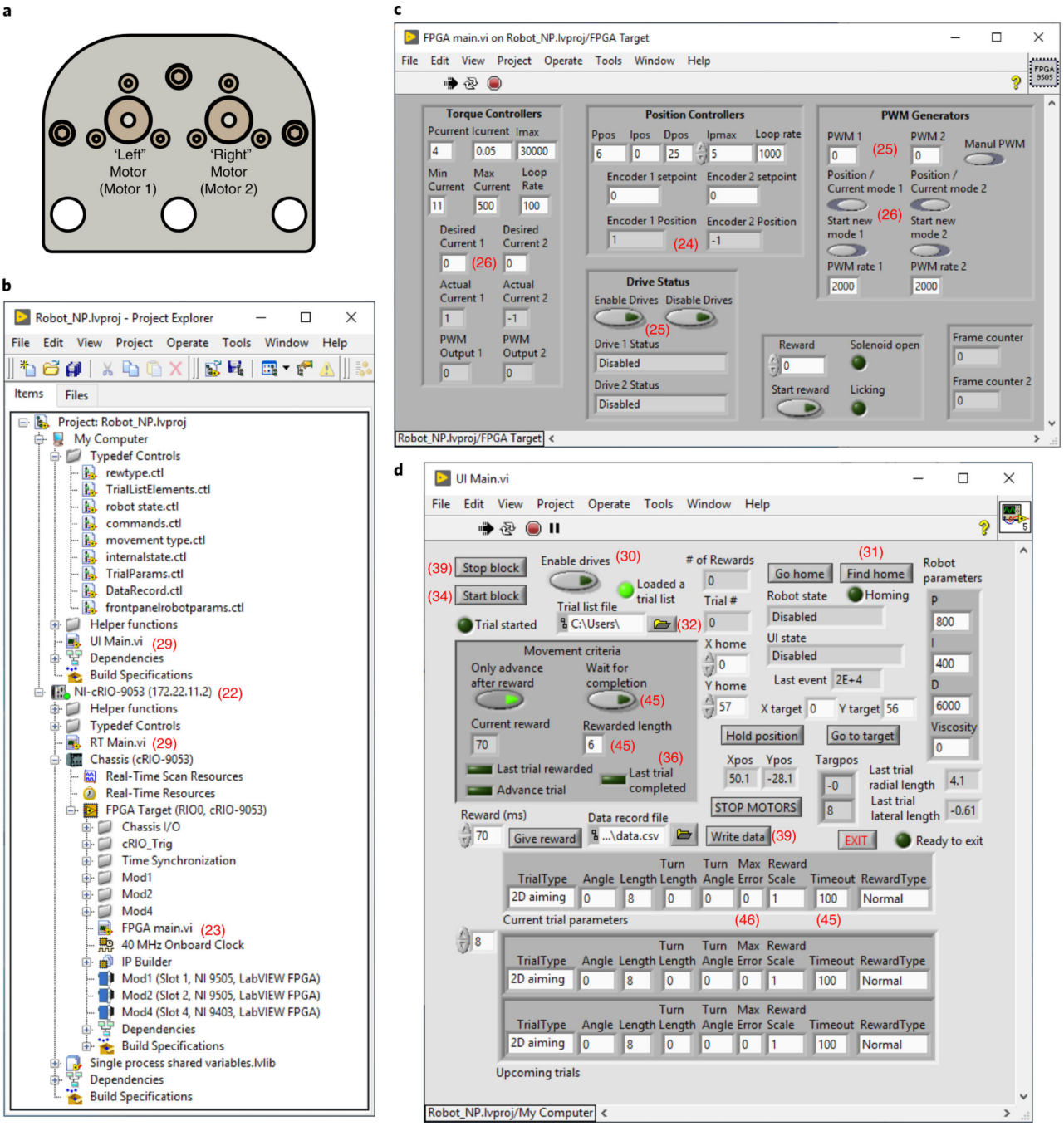

Fig. 5 |. Custom LabVIEW software and user interface.

a, Top view of the manipulandum with linkages removed, illustrating our motor designations 'left'/'right' ('1' and '2' in all software labeling) for consistency. **b,** The 'Project Explorer' hierarchy showing all project components. **c,** FGPA main.vi front panel user interface (Steps 23–28). **d,** UI Main.vi front panel user interface (Steps 29–39). A glossary of user interface controls is listed in Box 1.

Table 1 |

Troubleshooting table

Step	Problem	Possible reason	Solution
42	Animal walks so that body is out of restraining tube	Tube is too far posterior or at an uncomfortable height relative to head restraint	Adjust restraining tube anterior-posterior location or height
43	Animal grasps spout	Spout is too close to face	Move spout farther away
	Animal does not lick	Spout is too far from face	Move spout closer
44	Animal does not contact handle with paw	Handle is too far from natural range of paw positions	Move handle closer to body
	Animal grasps handle with wrong paw	Restraining tube left paw ‘hand rest’ is not positioned near natural left paw location. Device handle is too close to midline	Adjust position of the restraining tube and/or device handle
45	Animal terminates after a small number of trials	Insufficient prior water restriction or potentially administration of water droplets that are too large	Restrict water for more days or decrease reward duration/size
	Movement of the handle appears difficult or unnatural for the animal	If the handle appears to ‘crowd’ the animal’s forelimb movement, it may be too close	Move handle ventrally and/or more anterior
46	Animal has a constant bias for one movement direction that does not improve with substantial training	Handle position may not be ‘neutral’ for the animal (e.g., if too close to the body, it may be difficult to push straight ahead)	Often, moving the handle farther ventrally and sometimes also away from the directional bias can yield a more neutral natural motion direction