bio-protocol

Open Access

Published: Nov 05, 2024

A Real-Time Approach for Assessing Rodent Engagement in a Nose-Poking Go/No-Go Behavioral Task Using ArUco Markers

Thomas J. Smith[1, #], Trevor R. Smith[2, #], Fareeha Faruk[1], Mihai Bendea[1], Shreya Tirumala Kumara[3], Jeffrey R. Capadona[4, 5], Ana G. Hernandez-Reynoso[4] and Joseph J. Pancrazio[3, *]

[1]School of Behavioral and Brain Sciences, The University of Texas at Dallas, Richardson, TX, USA
[2]Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV, USA
[3]Department of Bioengineering, The University of Texas at Dallas, Richardson, TX, USA
[4]Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH, USA
[5]Advanced Platform Technology Center, Louis Stokes Cleveland Veterans Affairs Medical Center, Cleveland, OH, USA
*For correspondence: joseph.pancrazio@utdallas.edu
[#]Contributed equally to this work

Abstract

Behavioral neuroscience requires precise and unbiased methods for animal behavior assessment to elucidate complex brain–behavior interactions. Traditional manual scoring methods are often labor-intensive and can be prone to error, necessitating advances in automated techniques. Recent innovations in computer vision have led to both marker- and markerless-based tracking systems. In this protocol, we outline the procedures required for utilizing Augmented Reality University of Cordoba (ArUco) markers, a marker-based tracking approach, to automate the assessment and scoring of rodent engagement during an established intracortical microstimulation-based nose-poking go/no-go task. In short, this protocol involves detailed instructions for building a suitable behavioral chamber, installing and configuring all required software packages, constructing and attaching an ArUco marker pattern to a rat, running the behavioral software to track marker positions, and analyzing the engagement data for determining optimal task durations. These methods provide a robust framework for real-time behavioral analysis without the need for extensive training data or high-end computational resources. The main advantages of this protocol include its computational efficiency, ease of implementation, and adaptability to various experimental setups, making it an accessible tool for laboratories with diverse resources. Overall, this approach streamlines the process of behavioral scoring, enhancing both the scalability and reproducibility of behavioral neuroscience research. All resources, including software, 3D models, and example data, are freely available at https://github.com/tomcatsmith19/ArucoDetection.

Key features

- The ArUco marker mounting hardware is lightweight, compact, and detachable for minimizing interference with natural animal behavior.
- Requires minimal computational resources and commercially available equipment, ensuring ease of use for diverse laboratory settings.
- Instructions for extracting necessary code are included to enhance accessibility within custom environments.

Cite as: Smith, T.J. et al. (2024). A Real-Time Approach for Assessing Rodent Engagement in a Nose-Poking Go/No-Go Behavioral Task Using ArUco Markers. Bio-protocol 14(21): e5098. DOI: 10.21769/BioProtoc.5098.

bio-protocol

Open Access

Published: Nov 05, 2024

- Developed for real-time assessment and scoring of rodent engagement across a diverse array of pre-loaded behavioral tasks; instructions for adding custom tasks are included.
- Engagement analysis allows for the quantification of optimal task durations for consistent behavioral data collection without confirmation biases.

Keywords: ArUco markers, Automated scoring, Computer vision, Engagement analysis, Go/no-go, Real-time tracking

This protocol is used in: eNeuro (2024), DOI: 10.1523/ENEURO.0500-23.2024

Background

Behavioral neuroscience relies on an accurate assessment of animal behavior to understand complex brain–behavior relationships. Traditionally, such assessments have been performed using labor-intensive manual scoring methods, which are time-consuming, subject to human error, and can become impractical for large-scale studies [1,2]. Recent advancements in computer vision and tracking technologies have led to the development of automated behavioral scoring systems, offering more efficient and objective ways to analyze animal behavior [3–7]. Of these technologies, markerless-based approaches, which leverage machine-learning algorithms for position estimation, have been favored for their ability to capture complex animal movements by tracking an object's inherent features instead of an attached marker [5,8]. However, they often require high-end computational equipment and extensive training data for robust performance [3,6]. For instance, DeepLabCut, a leading markerless system, recommends a powerful GPU for training models, as relying solely on the CPU can significantly slow execution [3]. Alternatively, marker-based approaches utilize physical markers that can be attached to animals directly, tracking only the markers themselves as a robust estimation of an object's movement and orientation [4,7,9]. Among these marker-based tracking systems, Augmented Reality University of Cordoba (ArUco) markers have emerged as a promising tool for automated behavioral scoring procedures [10]. ArUco markers consist of black-and-white square grid patterns of various sizes, serving as fiducial target markers that are easily detected and tracked by cameras. Unlike markerless systems, ArUco markers do not rely on machine learning or extensive training data to be tracked, making them computationally lightweight and feasible for real-time analyses [10]. This simplicity enables ArUco markers to be tracked with minimal computational resources, such as low-end CPUs, ensuring accessibility and scalability for a wide range of laboratories. However, one limitation of this approach is the selectivity of the tracked animal behavior. In contrast to a markerless system's ability to track quick complex movements with predictions between unmarked frames, the ArUco approach is best when utilized for simple movement analyses like locomotion due to camera capture motion blur [10–12]. Nevertheless, ArUco markers offer flexibility in experimental setups where researchers can adapt their tracking to suit their specific experimental needs and study a wide range of behaviors in diverse animal models. Although simple by design, detailed protocols enabling researchers with different backgrounds to readily employ these methods for animal tracking are lacking. Here, we present detailed protocols for establishing a simple and consistent method for automated behavioral scoring of task engagement in an intracortical microstimulation-based nose-poke, go/no-go task using ArUco markers. Our step-by-step approach provides clear instructions, allowing researchers to implement ArUco marker-based tracking systems effectively, thereby advancing the accessibility and reproducibility of behavioral neuroscience research.

Materials and reagents

Biological materials

1. Male Sprague-Dawley rats (Charles River Laboratories Inc., catalog number: 001CD)

Laboratory supplies

1. ¼ in. diameter vinyl tubing (the length required depends on the setup)
2. ¼ in. diameter, 20 threads/inch, 1 in. long bolt with corresponding nut and (8×) washers
3. 3 mm diameter heat shrink tubing (NTE Electronics, catalog number: HS-ASST-5)
4. ¾ in. binder clip (DUIJINYU, catalog number: W003)
5. 4 in. cable ties (Grand Rapids Industrial Products, catalog number: 54157)
6. ⅝ in. binder clip (OWLKELA, catalog number: Clips-Black-0.6inch-120pcs)
7. 6 in. cable ties (Utilitech, catalog number: SGY-CT33)
8. (2×) 4 ft. Bayonet Neill–Concelman (BNC) cables
9. 6.4 mm diameter heat shrink tubing (Gardner Bender, catalog number: HST-101B)
10. Acoustic foam wedges (Ultimate Support, catalog number: UA-KIT-SBI)

11. Break away male header pins, straight (Leo Sales Ltd., catalog number: LS-00004)
12. Clear packing tape (Duck Brand, catalog number: 287206)
13. ClearWeld quick-set epoxy (J-B Weld, catalog number: 50112)
14. Double-sided foam tape
15. Dustless sugar precision reward pellets (Bio-Serv, catalog number: F0021)
16. Electrical tape
17. Hot glue gun with thermal-plastic adhesive (hot glue) sticks
18. M3 10 mm machine screws with corresponding nuts (6×) (K Kwokker, catalog number: TLEEP2324+FBADE-28)
19. M3 20 mm machine screws with corresponding nuts (12×) (K Kwokker, catalog number: TLEEP2324+FBADE-28)
20. M3 30 mm machine screw with corresponding nut (Everbilt, model: 837841)
21. Medical-grade air tank with regulator (Cuevas Distribution Inc., catalog number: AI USPxxx)
22. Polylactic acid (PLA) fused deposition modeling (FDM) 3D printer filament
23. Rubber bands (Advantage, catalog number: 2632A)
24. Solder wire (Shenzhen Joycefook Technology Co. Ltd, catalog number: JF850)
25. Stainless steel spring cable shielding (Protech International Inc., catalog number: 6Y000123101F)
26. Super glue (Loctite, catalog number: 234796-6)
27. Supplemental food pellets (LabDiet, catalog number: 5LL2, Prolab RMH 1800)
28. TIP120 negative-positive-negative (NPN) transistors (3×) (BOJACK, catalog number: BJ-TIP-120)
29. White stock printer paper

Equipment

1. 12V 1A DC power supplies (2×) (Mo-gu, catalog number: B09TXJ9RK6)
2. 150 mm T-Slot 2020 aluminum extrusion rails (4×) (MECCANIXITY, catalog number: mea231123ee000975)
3. 200 mm T-Slot 2020 aluminum extrusion rails (x8) (Tsnamay, catalog number: TSCP5035)
4. 3D-printed ArUco marker mounting assembly hardware (see Procedure)
5. 3D-printed camera mounts (see Procedure)
6. 3D-printed RGB LED strip mounting kit (see Procedure)
7. 3D-printed T-Slot rail connector pieces (see Procedure)
8. 4-pin 12V RGB LED strip
9. 5 mm wide ~4.5 in. long hex wrench
10. 57 in. × 42 in. × 56 in. sound-attenuating chamber (Maze Engineers, catalog number: 5831)
11. 5V activation relay module (Inland, catalog number: 509687)
12. ArUco tracking cameras (2×) (Logitech, catalog number: 960-001105)
13. ATmega328 UNO R3 microcontroller (Inland, model: UNO R3 BOARD)
14. Behavior camera (j5create, catalog number: JVCU100)
15. Cable tensioner bar (see Procedure)
16. Commutator (Moog Inc., catalog number: AC6023-18)
17. Disposable lighter
18. Electrical Stimulator (Plexon Inc., model: PlexStim)
19. Fused deposition modeling (FDM) 3D printer
20. Heat gun (Wagner, catalog number: FBA_503008-cr)
21. Luxmeter (Leaton, catalog number: HRD-PN-79081807)
22. Multimeter (Innova, model: 3320)
23. Omnetics tether adapter (Omnetics Connector Corporation, catalog number: A79021-001)
24. Operant conditioning chamber (Vulintus, model: OmniTrak)
25. Oscilloscope (Tektronix Inc., model: TBS 1052B)
26. Pneumatic solenoid (AOMAG, catalog number: SKUSKD1384729)
27. Soldering iron (Guangzhou Yihua Electronic Equipment Co. Ltd., model: YIHUA 926 III)

28. Stackable headers, female 10 pin (2×) (Leo Sales Ltd., catalog number: LS-00009)
29. Workstation computer with Windows 10–11 (Dell, model: precision 5860 workstation)

Software and datasets

All data and code have been deposited to GitHub: https://github.com/tomcatsmith19/ArucoDetection

1. Cura v4.13.0 (Ultimaker)
2. Microsoft Excel
3. MATLAB vR2023b (MathWorks; requires a student license or higher)
   a. Add-On: MATLAB Support Package for Arduino Hardware v23.1.0 (MathWorks)
   b. Add-On: MATLAB Support Package for USB Webcams v23.1.0 (MathWorks)
   c. Add-On: Instrument Control Toolbox v4.8 (MathWorks)
4. PlexStim v2.3 (Plexon Inc.)
5. Prism v10.0.2 (GraphPad; requires a student license or higher)
6. Python v3.10.11 (Python Software Foundation)
7. TekVisa v4.0.4 (Tektronix)
8. Visual Studio Code v1.85 (Microsoft)
   a. Extension: Pylance v2024.6.1 (Microsoft)
   b. Extension: Python v2024.8.0 (Microsoft)
   c. Extension: Python Debugger v2024.6.0 (Microsoft)
   d. Package: numpy v1.25.0
   e. Package: opencv-contrib-python v4.7.0.72 (Open Source Vision Foundation)
   f. Package: scipy v1.13.0 (SciPy)
9. Microsoft Word

Procedure

A. Constructing the behavioral chamber

1. Preparing the sound-attenuating chamber
   a. Adhere acoustic foam wedges around the inside walls of the sound-attenuating chamber (excluding the floor) using super glue or double-sided foam tape. See Figure 1A for an example of the intended outcome.

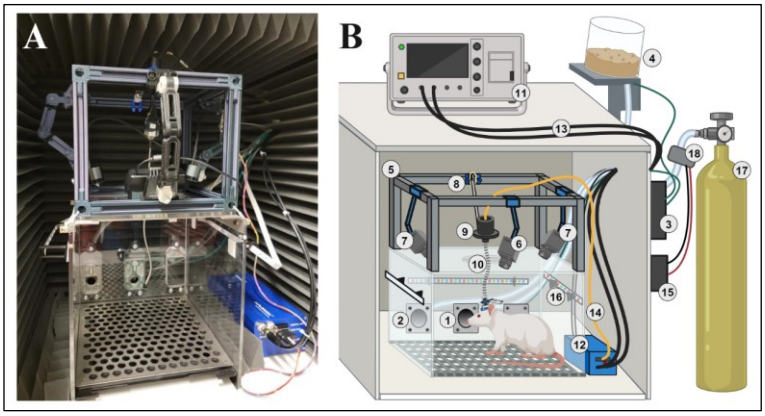


Figure 1. Behavioral chamber setup. A) Photographic image of the behavioral chamber configured inside the sound-attenuating chamber. B) Cartoon illustration of the behavioral chamber with numbered

components: 1) nose-poke module, 2) reward pellet module, 3) OmniTrak controller board, 4) sugar pellet dispenser, 5) T-Slot 2020 rail frame, 6) mounted behavior camera, 7) mounted ArUco marker tracking cameras, 8) cable tensioner bar, 9) commutator, 10) tether, 11) oscilloscope, 12) electrical stimulator, 13) BNC cables, 14) stimulation cable, 15) microcontroller, 16) RGB LED strip, 17) medical-grade air tank and regulator, and 18) pneumatic solenoid.

2. Setting up the OmniTrak operant conditioning chamber
   a. Place the assembled OmniTrak operant conditioning chamber with nose-poke (Figure 1B.1) and pellet dispenser (Figure 1B.2) modules inside of the sound-attenuating chamber.
      Caution: Read the installation instructions provided by Vulintus for the OmniTrak operant conditioning chamber first, then proceed with the protocol instructions listed below.
   b. Mount the matching OmniTrak controller board (Figure 1B.3) to the outside of the sound-attenuating chamber's right-side panel with double-sided foam tape.
   c. Connect one of the provided RJ45 (ethernet) cables from the OmniTrak nose-poke module to the controller board by feeding it through the upper-right corner hole of the sound-attenuating chamber.
   d. Place the OmniTrak reward pellet dispenser (Figure 1B.4) on the top right edge of the sound-attenuating chamber and connect it to the OmniTrak reward pellet module using the provided vinyl tubing, feeding it through the same hole.
   e. Insert the remaining RJ45 and provided 4-pin power connector cable into both the reward pellet dispenser and the controller board as instructed by Vulintus.
   f. Fill the dispenser reservoir halfway with dustless sugar precision reward pellets.
   g. Connect the OmniTrak controller board to the intended computer using the provided USB cable.
   h. Power on the OmniTrak system by inserting the controller board power cable into a nearby wall socket.
3. 3D printing behavioral chamber components
   a. Visit the highlighted GitHub page for this protocol (https://github.com/tomcatsmith19/ArucoDetection) and download each of the .stl files listed within the *3D Models* folder.
   b. Before printing, slice each of the .stl files in Ultimaker's Cura software to transform them into usable g-code files that are specific to your 3D printer.
      Note: In Cura, a layer height of 0.2 mm and cubic pattern infill density of 20% were set for each print, although many setting combinations would work.
   c. Using an FDM 3D printer with PLA filament loaded, print the "RGB LED strip mounting kits" (1×), the "T-Slot rail connector pieces" (8×), and the "camera mount" (3×) files, which include modified models from username: danid87 on Thingiverse (https://www.thingiverse.com/thing:2600507).
      Note: Any third-party slicer and FDM 3D printer will work without needing to modify the .stl files as long as the printer has an allocated print volume of at least 220 mm × 220 mm × 250 mm.
4. Assembling the T-Slot 2020 rail frame
   a. Connect the T-Slot 2020 aluminum extrusion rails with the newly printed T-Slot rail connectors to form an open cube frame. Use 200 mm rails for the top and bottom planes, and 150 mm rails for the vertical sides.
      Note: A soft rubber mallet may be used to insert the T-Slot rail connectors completely if they get stuck.
   b. Fix the T-Slot rail frame on top of the operant conditioning chamber (Figure 1B.5) using glue, velcro, double-sided foam tape, cable ties, or a combination of options depending on how permanent the user would prefer the setup to be.
5. Installing cameras to the T-Slot rails
   a. Assemble three of the 3D-printed camera mounts by referencing the steps outlined in Figure 2.
   b. Mount the behavior camera to the top T-Slot rail closest to the chamber doors (Figure 1B.6) and the two ArUco tracking cameras to the top left and right rails (Figure 1B.7).
   c. Plug in each of the camera's USB cables to the computer through the hole in the sound-attenuating chamber.
      Note: If the cables are too short to reach the computer, USB 3.0 extenders may be utilized.

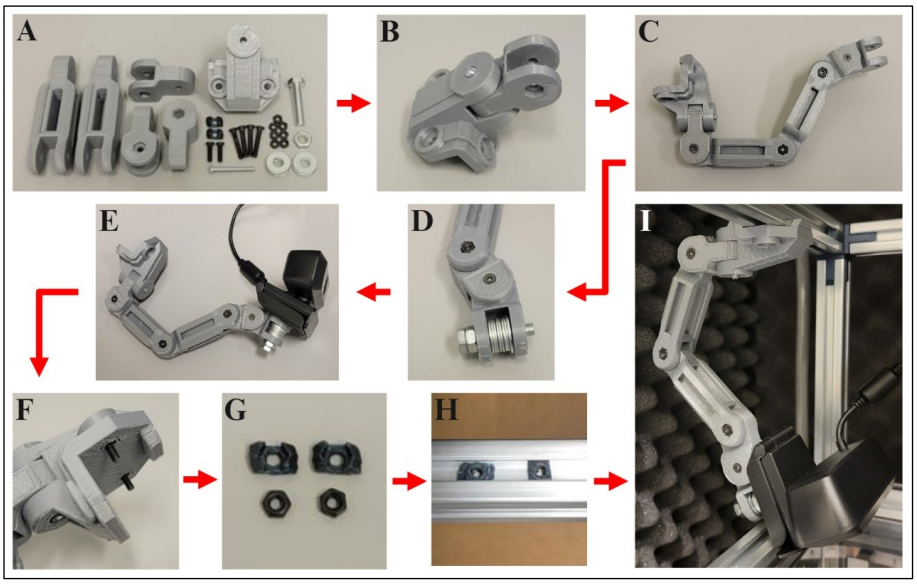


Figure 2. Instructions for assembling the camera mount. A) Gather the 3D-printed camera mount pieces, M3 10 mm machine screws (2×), M3 20 mm machine screws (4×), M3 30 mm machine screw (1×), M3 nuts (7×), ¼ in. diameter 20 threads/inch 1 in. long bolt (1×) with corresponding nut (1×) and washers (8×), and camera (1×). B) Connect the T-Slot mounting piece to the first segment of the articulating arm with the M3 30 mm screw and nut. C) Connect the remaining articulating arm pieces with the M3 20 mm screws and nuts. D) Thread the 1 in. bolt through the last articulating arm segment with a nut at the bolt's base and eight washers filling the gap. E) Screw in the camera's base to the threaded end of the exposed 1 in. bolt. F) Place two of the M3 10 mm screws through the holes of the plastic T-Slot rail mounting piece. G) Fit an M3 nut into each of the two small T-Slot rail adapter pieces. H) Slide the assembled T-Slot rail adapter pieces into the groves of the T-Slot rail (left) and rotate them 90° to lock them into place (right). I) Align the M3 10 mm screws within the T-Slot rail mounting piece with the T-Slot rail adapter pieces and screw them in, mounting the camera arm to the rail.

6. Creating and attaching the tensioner bar, commutator, and tether
    a. First, modify the commutator by cutting the exposed wires at the top and bottom of the device; the top group of wires should remain 8 in. long and the bottom group should remain 3 in. long. See Figure 3A for an example of the final product.
    b. Assign pin numbers (1–16) to each of the colored wires at the top with matching colors and numbers at the bottom end, leaving two extra wire colors not accounted for; these will become our ground and reference wires.
    c. Solder the top group of wires to two rows of eight male break away pins, arranging the top row pins as 15, 13, 11, 9, 7, 5, 3, and 1 from left to right when looking directly into the pins. The bottom row pins should be arranged as 16, 14, 12, 10, 8, 6, 4, and 2 from left to right when looking directly into the pins (see Figure 3A).
    d. **Critical:** When soldering any of the wires, cover each of them with a piece of 3 mm heat shrink tube that is just long enough to cover the exposed joints. This will help prevent any of the connections from becoming shorted together.
    e. Now on the bottom end of the commutator, solder all 18 wires to two rows of ten female stackable header pins. The top row pins should be X (blank), 1, 3, 5, 7, 9, 11, 13, 15, and X from left to right when looking directly into the pins. The bottom row pins should be G (ground), 2, 4, 6, 8, 10, 12, 14, 16, and R (reference) from left to right when looking directly into the pins (see Figure 3A).
    f. Wrap the top and bottom ends of the commutator in electrical tape to secure it (see Figure 3A).

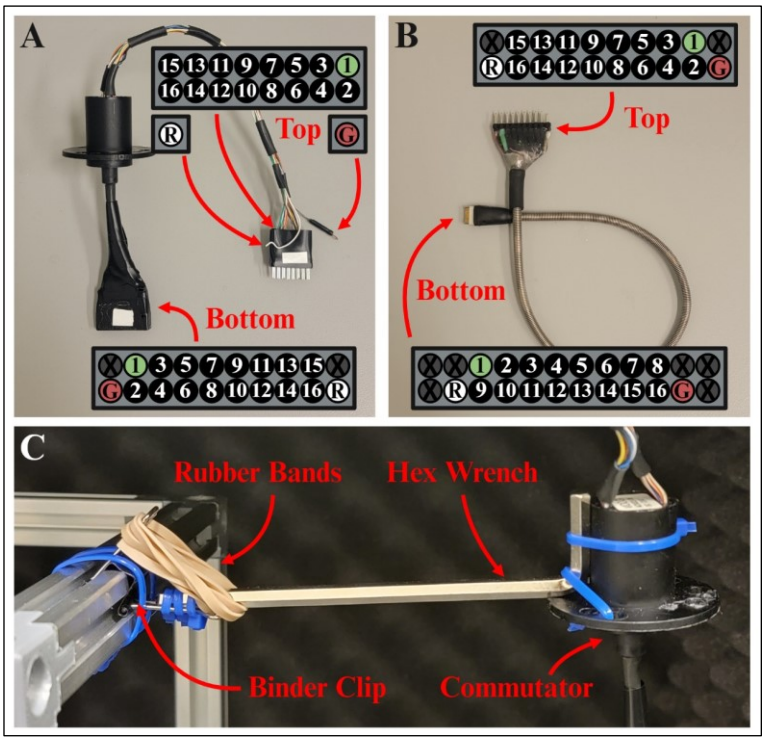


Figure 3. Modified commutator and tether cord. A) Image depicting the final product of the commutator modifications with labeled top and bottom end pinouts when looking directly into the pins. B) Image depicting the final product of the tether with labeled top and bottom end pinouts when looking directly into the pins. C) Side-view image of the cable tensioner bar complete with a ¾ in. binder clip, rubber bands, a hex wrench, and the commutator from panel B.

g. Finally, strip 0.5 cm from the ground wire at the top end of the commutator and leave the reference wire unmodified and unattached.

h. Moving onto the tether, begin by cutting the Omnetics tether adapter to be 11.5 in. long. Take note that each of the colored wires corresponds to a specific gold pin on the Omnetics adapter end. See Figure 3B for an example of the final product or Video 1 for a demonstration of how to fabricate the tether.

Critical: There are eight gold pins on the top row of the connector with two standoff pins on each side and ten gold pins on the bottom row with one standoff pin on each side. When the gold pins are facing toward you, the top row's pins are X, X, 1, 2, 3, 4, 5, 6, 7, 8, X, and X from left to right. The bottom row's pins are X, R, 9, 10, 11, 12, 13, 14, 15, 16, G, and X (see Figure 3B).

Video 1. Fabrication of the tether cord. Instructional video that outlines the steps required to fabricate the exact tether design used within this protocol.

i. Cut eleven 0.5 in. pieces of the 3 mm heat shrink tube and slide them over all of the Omnetics tether adapter wires, using a heat gun to heat them into place with ¼ in. gaps between each piece.
j. Cut a 10.5 in. piece of stainless-steel spring cable shielding and slide it over the Omnetics tether cord.
k. Using the disposable lighter, burn off the insulating layer of each wire at the ends, exposing ~3 mm of bare wire.
l. Solder the wires to two rows of ten male breakaway pins. The top row wires should match the female end of the commutator when plugged in: X, 15, 13, 11, 9, 7, 5, 3, 1, and X from left to right when looking directly into the pins. The bottom row wires should be as follows: R, 16, 14, 12, 10, 8, 6, 4, 2, and G from left to right when looking directly into the pins (see Figure 3B).
m. **Critical:** Before moving to the next step, use a multimeter set in continuity mode to ensure that each connection has continuity with only itself and no other pin.
n. Coat the exposed wires at both ends of the tether with clear quick-set epoxy to cement the wires in place. Let the epoxy harden for at least 1 h.
o. Cut six ½ in. pieces of 6.4 mm heat shrinking tube, then slide them onto the tether over the Omnetics end. One at a time, place one piece at each end of the tether and heat it in place with a heat gun. Continue this process until all six pieces are secured (three on each side).
p. Lastly, create the cable tensioner bar (Figure 1B.8). The cable tensioner used within this experiment is produced from a ¾ in. binder clip, rubber bands, 6-in. cable ties, and a single 5 mm hex wrench that is ~4.5 in. long (see Figure 2C).
q. Start by clipping a ¾ in. binder clip to the center of the top T-slot rail closest to the back of the chamber around the top corner of the rail that faces the middle of the cage.
r. Secure the clip in place with two 6 in. cable ties wrapped around each side.
s. Take the long side of the hex wrench and fasten it to the bottom metal loop of the binder clip using additional cable ties.
t. Double-wrap 3 rubber bands around the hex wrench and the top metal loop of the binder clip.
u. Finally, cable-tie the short side of the hex wrench to the commutator so that the commutator is suspended directly above the center of the operant conditioning chamber (Figure 1B.9).
v. Insert the soldered 20-pin end of the tether into the bottom of the commutator (Figure 1B.10).
w. Insert the top end of the commutator into the PlexStim stimulation cord with the top pins matching the side of the stimulation cord with the white arrow on it.
x. Clip the stimulation cord's metal clamp onto the exposed ground wire at the top of the commutator.

7. Connecting the oscilloscope

a. Place the oscilloscope on top of the sound-attenuating chamber (Figure 1B.11).
b. Insert its provided USB cable into the computer.
c. Plug in its provided power supply cord into a nearby outlet.

8. Connecting the electrical stimulator
   a. Place the PlexStim device inside the sound-attenuating chamber to the right of the operant conditioning chamber (Figure 1B.12).
   b. Plug in its provided power supply cord into a nearby outlet through the hole of the sound-attenuating chamber.
   c. Connect the two 4 ft. BNC cables (Figure 1B.13) between the stimulator and oscilloscope's voltage and current monitor channels through the same hole.
   d. Insert the stimulator's provided stimulation cable into the top of the commutator (Figure 1B.14).
   e. Connect the stimulator to the computer using the provided USB cable through the hole in the sound-attenuating chamber.
9. Wiring the ATmega328 microcontroller
   a. Follow the wiring diagram shown in Figure 4 to connect the ATmega328 UNO R3 microcontroller to the RGB LED strip and pneumatic solenoid peripherals.
   Critical: To utilize the software without reconfiguration, the red, green, and blue LED wires should be connected to PWM digital pins 9, 6, and 5, respectively, on the microcontroller. The signal, ground, and Vcc pins of the 5V activation relay module should be connected to pins digital 11, ground, and 5V on the microcontroller, respectively. If these pins are not available, they must be reassigned in the *ArucoDetection → MATLAB Behavior Program → Required Behavior Functions →* LEDColorChange.m and TriggerSolenoidAirPuff.m functions.
   Note: ATmega328 UNO R3 microcontroller was selected for its 5V logic, compatibility with MATLAB, (3×) PWM pins, (1×) digital pin, (1×) 5V source pin, and (1×) ground pin. TIP120 NPN transistors were selected for their NPN type, 5V logic, and current support for over 1A. The 5V activation relay module was selected to match the 5V logic of the ATmega328 UNO R3 microcontroller and its ability to support greater than 1A at 12V of power required by the pneumatic solenoid.

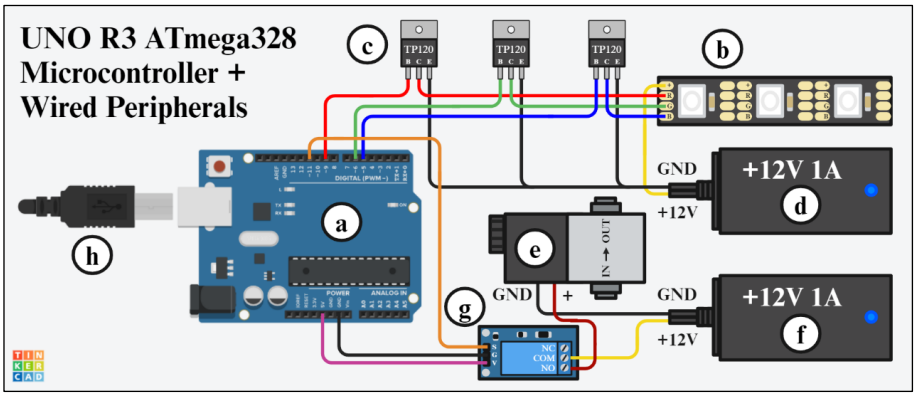


Figure 4. Wiring diagram between the microcontroller and peripherals. a) ATmega382 UNO R3 microcontroller. b) 12V 4-pin RGB LED strip with four contact pads labeled from top to bottom: +12V power (+), red LEDs (R), green LEDs (G), and blue LEDs (B). c) TIP120 transistors with three pins labeled from left to right: base (B), collector (C), and emitter (E). d) 12V 1A DC power supply for the RGB LED strip with two wires labeled from top to bottom as ground (GND) and positive (+12V). e) Pneumatic solenoid with two wires labeled from left to right as ground (GND) and positive (+). f) 12V 1A DC power supply for the pneumatic solenoid with two wires labeled from top to bottom as ground (GND) and positive (+12V). g) 5V activation relay module with three pins labeled on the left-hand side of the device from top to bottom as signal (S), ground (G), and Vcc (V); three screw terminals labeled on the right-hand side of the device from top to bottom as normally closed (NC), common (COM), and normally open (NO). h) USB type-B cable for supplying power to the microcontroller alongside data communication with the

computer.

b. Mount the microcontroller to the outside of the sound-attenuating chamber's right-side panel with double-sided foam tape (Figure 1B.15).
Note: A 3D-printed box can be made to house all the electronics before mounting to the chamber wall.
c. Insert the microcontroller's USB type-B cable into the computer.

10. Mounting the RGB LED strip
    a. Gather the 3D-printed RGB LED strip mounting kit parts and begin by gluing the elongated rectangular strips to the inside of the bracket pieces on the small side (two brackets per rectangle). Reference Figure 1A for further guidance.
    b. With double-sided foam tape, adhere the large end of the bracket pieces and rectangular strips to the outside of the left, back, and right side-panel walls of the conditioning chamber (~90% up from the base), keeping the rectangular strips level with the base of the chamber.
    c. From the outside of the sound-attenuating chamber, push the RGB LED strip through the hole in the chamber so that most of the LEDs are inside while the opposite end remains wired to the microcontroller on the outside.
    d. Thread the RGB LED strip through the bracket mounts, gluing the backside to each of the rectangular strips so that the LEDs face toward the center of the operant conditioning chamber (Figure 1B.16).
    e. Plug the 12V 1A power supply that is wired to the RGB LED strip into a nearby power outlet.
11. Connecting the pneumatic solenoid for air-puffing punishment
    a. Insert one end of the ¼ in. diameter vinyl tubing to the regulator on a medical-grade air tank (Figure 1B.17). The tank should be securely mounted in place outside of the behavioral chamber.
    b. Cut the vinyl tubing ~4 ft. from the opposite end and set aside.
    c. From the cut end that is connected to the air tank, attach it to the "IN" hole on the pneumatic solenoid with brass fittings (Figure 1B.18).
    d. Then, connect one end of the severed vinyl tube piece to the "OUT" hole on the pneumatic solenoid, again with brass fittings.
    e. Push the opposite end of that tube through the hole in the sound-attenuating chamber and hot glue it to the backside of the nose-poke module hole. The end result should contain the opening of the vinyl tube being flush, secured, and centered with the backside opening of the nose-poke module hole.
    Note: A piece of cardboard with a hole can be used to secure the vinyl tube while being glued in place; glue the cardboard piece itself to the backside of the module while sealing the space around the hole with additional hot glue.

B. Installing and configuring software packages

1. Behavioral task MATLAB code
    a. Visit the provided GitHub page (https://github.com/tomcatsmith19/ArucoDetection), click on the green < > *Code* button, and download the entire repository as a zip file to a desired location on the computer.
    b. Locate the downloaded zip file within File Explorer, right-click on the zip file, and select *Extract All*.
    c. From the extracted folder, open the Excel file located at *ArucoDetection → MATLAB Behavior Program → ExcelSessionData → AnimalWeightCheck.xlsx*.
    d. Within that file, replace the animal names and their corresponding weights in grams that are specific to your study within the designated cells.
    Note: For this protocol's specific behavioral paradigm, each animal's weight is monitored weekly due to mild food deprivation practices. The MATLAB behavioral program app possesses a button that evaluates if the animal's daily weight drops below 90% of its weekly recorded weight from the AnimalWeightCheck.xlsx file; if the animal passes the weight check, it may proceed with the experiment. All animals are fed supplemental food pellets after each behavioral session in quantities calculated by the behavioral app.
    e. Save the *AnimalWeightCheck.xlsx* file and close it.

2. MATLAB
   a. With a browser, navigate to the following URL on MathWorks's website (https://www.mathworks.com/products/new_products/release2023b.html), click the *Download Now* button, and follow their instructions for downloading and installing MATLAB (vR2023b) to your computer.
   Critical: Download the specified version of MATLAB; future versions may not be compatible.
   Note: You may need admin privileges for your machine to install this software.
   b. After installing MATLAB, open the application and set its active file directory to the *ArucoDetection → MATLAB Behavior Program* file path that was extracted from the GitHub repository downloaded in procedure steps B1a–b.
   c. Navigate to the *Add-Ons* drop-down arrow button within the *Environment* section of the home menu ribbon. Click the drop-down arrow and select *Get Add-Ons*.
   d. Within the pop-up window, search for and install the following: 1) *MATLAB Support Package for Arduino Hardware*, 2) *MATLAB Support Package for USB Webcams*, and 3) *Instrument Control Toolbox*. See the Software and Dataset section for version numbers.
   Note: Version numbers may differ as each is consistently updated.
   e. Close MATLAB and restart the computer.
   f. Open MATLAB again and, within the active directory folder, double-click to open the *BehaviorApp.mlapp* file; this will open the MATLAB App Designer window with the corresponding behavioral app already open.
   g. Make sure that the *Design View* button is selected instead of *Code View*. Then, navigate to the *Session Setup* tab within the behavioral app (Figure 5).

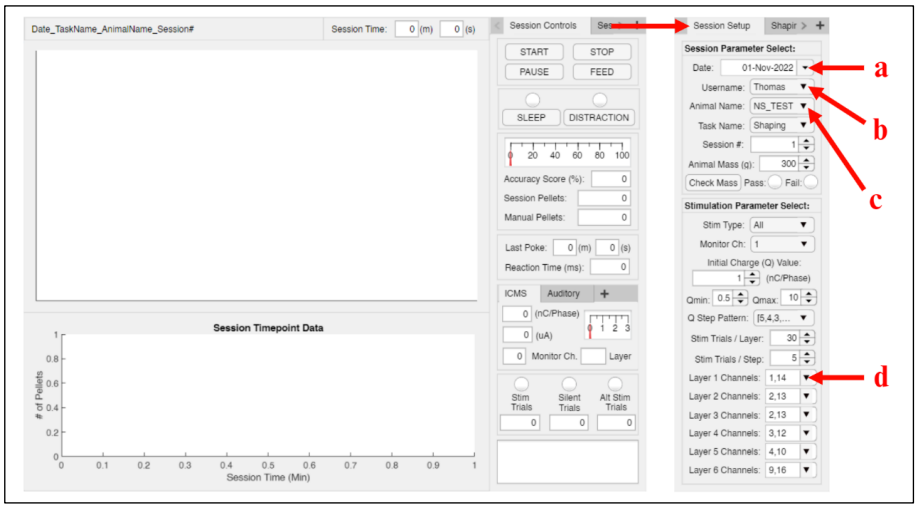


Figure 5. Navigating through the behavioral app within MATLAB app designer to update experimental parameters within the session setup tab. a) The drop-down box for selecting the current date. b) A drop-down list for selecting the name of the researcher conducting the current experiment. c) A drop-down list for selecting the name/identification of a specific animal. d) A drop-down list for selecting which microelectrode array channels will be simultaneously stimulated while the *Stim Type → Layer* option is chosen.

   h. Within the *Session Setup → Session Parameter Select* panel, update the listed values for *Date*, *Username*, and *Animal Name* (Figure 5a–c) to reflect the current date, the name of the researcher(s) who will be running the behavioral task, and the name of the animal(s) reflected within the *AnimalWeightCheck.xlsx* file that was updated in step B1d–e. To do so, double-click on each of the boxes. For *Date*, a pop-up calendar will appear; navigate to the desired date and click on it. For *Username* and *Animal Name*, click on any item in the list to reveal a positive and negative sign. Use

the positive sign to add names and the negative sign to remove names from the list.

i. Within the *Stimulation Parameter Select* panel, click into the *Layer 1 Channels* list (Figure 5d) and add any combination of stimulated channels (1–16) that are specific to the intended experiment by using the same positive and negative buttons described in the previous step.
Note: The remainder of the Layer 2–6 Channel boxes are currently non-functional for this version of the application.

j. Next, open the Windows Device Manager application from the computer's start menu. Then, click the drop-down menu arrow for the section labeled as *Ports (COM & LPT)*; make note of the COM port number that is displayed for which the UNO R3 ATmega328 microcontroller is connected.
Note: You can readily test this by noting the device names, unplugging the USB cable attached to the microcontroller, looking for the difference in device names, and then re-plugging the device.

k. Go back to the MATLAB behavior app window and switch the *Design View* option to *Code View*.

l. Locate line number 507: app.LED = arduino("COM5", "Uno"); and update the COM number to the one found in Device Manager.

m. Once finished, switch viewing modes back to *Design View* and save the newly updated version of the behavioral app.

n. Close the application.

3. PlexStim for electrical stimulator
   a. While using a web browser, navigate to Plexon Inc.'s website with the following URL: https://plexon.com/products/plexstim-electrical-stimulator-2-system/.
   b. Click the *PlexStim V2.3 – Windows 7 or Windows 10* link to download the PlexStim v2.3 installation file.
   c. Open the downloaded *StimulatorV2Setup.exe* file and follow the on-screen instructions.
   Note: You may need admin privileges for your machine to install this software.
   d. Restart the computer.
4. TekVisa for oscilloscope
   a. While using a web browser, navigate to Tektronix's website with the following URL: https://www.tek.com/en/support/software/driver/tekvisa-connectivity-software-v404/.
   b. Click the *Download File* button to download the correct TekVisa v4.0.4 installation file.
   c. Open the downloaded *TekVISA_404_066093809.exe* file and follow the on-screen instructions.
   Note: You may need admin privileges for your machine to install this software.
   d. Restart the computer.
5. Python
   a. While using a web browser, navigate to Python's website with the following URL: https://www.python.org/downloads/release/python-31011/.
   b. Scroll down until the *Files* section is located, then click on the *Windows installer (64-bit)* link to download the Python v3.10.11 installation executable file.
   c. Open the file and follow the prompted instructions.
   Critical: When prompted, be sure to check the box for *Add python.exe to PATH*. **This is a necessary step**.
   d. When prompted, select *install now*.
   Note: You will need admin privileges for your machine to install this software.
   e. When finished, click *close*.
   f. Restart the computer.
6. Visual Studio Code and ArUco Marker Tracking Code
   a. While using a web browser, navigate to the following URL: https://code.visualstudio.com/.
   b. Click on the *Download for Windows* button on the homepage to download the current Visual Studio Code application setup file.
   c. Open that file and follow the prompted instructions.
   Note: You may need admin privileges for your machine to install this software.
   d. Open the Visual Studio Code application and click on the *Extensions* button located on the left-hand side of the screen.

e. Search and install the newest versions of the following extensions: 1) *Python*, 2) *Pylance*, and 3) *Python Debugger*. See the Software and Dataset section for version numbers.
Note: Version numbers may differ as each is consistently updated.
f. Close the Visual Studio Code application.
g. Restart the computer.
h. Re-open the Visual Studio Code application.
i. Navigate to the top menu ribbon and select *Terminal* → *New Terminal*.
j. Within the newly opened terminal line, type *python --version* and press the enter button on the keyboard.
Critical: Ensure that the output text states "Python 3.10.11".
Troubleshooting: If Python is not found, ensure that it was properly installed and added to *PATH* during the installation instructions. If a different version of Python appears, be sure to install the following packages to Python v3.10.11 and not the other listed version. Additionally, if another Python version appears, be sure to verify that Python v3.10.11 appears as the selected Python version in the bottom right corner of the Visual Studio Code application window.
k. Next, type the following three commands into the terminal followed by the enter key after each line:
pip install numpy==1.25.0
pip install opencv-python==4.7.0.72
pip install scipy==1.13.0
Troubleshooting: If an error occurs that states that pip is not installed, then install pip (https://pip.pypa.io/en/stable/installation/).
l. Go to *File* → *Open Folder* at the top left of the screen and navigate to and open the already downloaded *ArucoDetection* folder from steps B1a-b. The corresponding files within that directory should now appear within the *Explorer* column of the Visual Studio Code interface.
Note: The drop-down arrow may need to be clicked to show each of the folders/files within the directory.
m. Find the *ArUco Tracking Program* folder, click its drop-down arrow to open the folder, and select the *AnimalDetect.py* file; this is the primary file that runs the ArUco tracking code in parallel with MATLAB.
n. Open MATLAB and type *webcamlist* into the command window followed by the enter key. This should output an array in text format, showing you the index value of each camera that is connected to the computer.
o. With those index values, make note of the two ArUco tracking cameras and subtract 1 from each of their index values found in MATLAB. MATLAB begins indexing at a value of 1, whereas Python begins at 0; the subtraction acts as a conversion to confirm the appropriate camera indexes.
p. Go back to the *AnimalDetect.py* file and update the number at the end of line 20: cap = cv2.VideoCapture(1) to the converted index value of the right-side ArUco tracking camera. Additionally, update the number at the end of line 23: cap2 = cv2.VideoCapture(3) to the converted index value of the left-side ArUco tracking camera.
Note: If the wrong cameras were chosen, they can always be updated again as needed.
q. Save the *AnimalDetect.py* file and close both Visual Studio Code and MATLAB.

C. ArUco marker mounting hardware

1. Assembling the hardware
a. Follow the instructions listed in step A3 to print the following ArUco marker mounting assembly hardware pieces: (1×) *ArUco Marker Mount*, (2×) *C-Clamp*, and (1×) *I-Bracket.*
b. Locate one ⅝ in. generic binder clip and carefully remove each of the two metal loops from the clip using tweezers or pliers.
c. In Windows File Explorer, open the *15x15mmArUcoMarker3.png* file from the *ArucoDetection* → *ArUco Setup Code* file path.
Note: Additional markers with various sizes can be produced by modifying the generateMarkers.py

file at this location within Visual Studio Code.

d. Open the image into any image processing application like Microsoft Word or Adobe Illustrator and print the marker on white stock printer paper, then cut it out.
Critical: Make sure that the print size is true to 15 mm × 15 mm. Some applications attempt to resize the image when printing, but the image file itself is set to 15 mm × 15 mm.
Optional: Place a piece of clear packing tape over the ArUco marker. This can help with the longevity of the marker, although a newly printed marker can always be re-glued overtop of the existing one.
Caution: Keep a thin white border around the marker when cutting it out of the computer paper to maintain some contrast if the experimental animals used do not have white fur.

e. Referencing Figure 6 for guidance, assemble the ArUco marker mounting pieces for a fit test.
Troubleshooting: Reprint any pieces that do not fit properly.

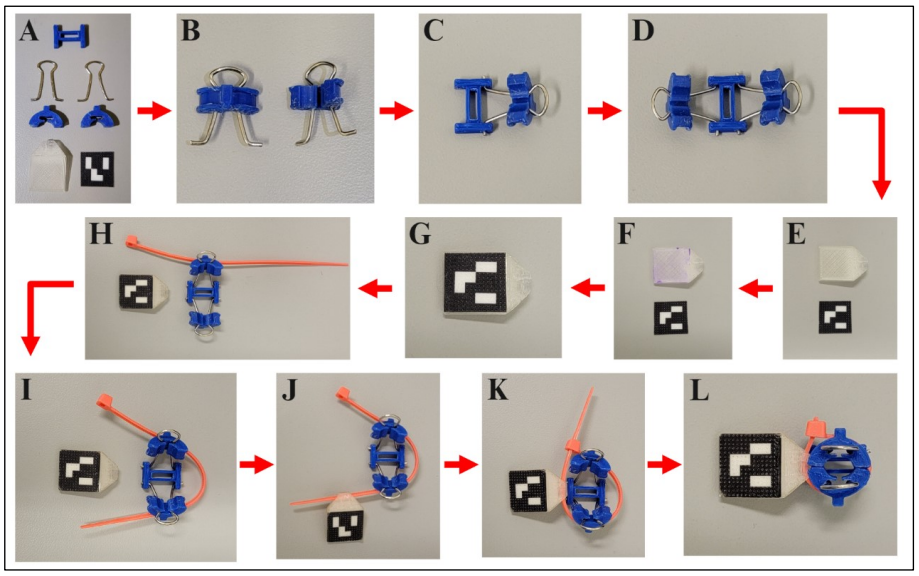


Figure 6. Instructions for connecting the ArUco marker mounting assembly. A) Gather each of the seven ArUco marker mounting pieces: (1×) I-Bracket, (2×) ⅝ in. binder clip wire loops, (2×) C-Clamps, (1×) ArUco marker mount, and (1×) 15 mm × 15 mm paper printed ArUco marker. B) Pinch the binder clip wire loops together to slot them into the C-Clamps. C) With your index finger and thumb, slide the C-Clamp down the binder clip wire loop until it almost reaches the protruding prongs at the end. Then, insert the prongs into the holes of the I-Bracket. D) Repeat the process described in panel C for the opposite side of the I-Bracket. E) Orient the ArUco marker mount and paper-printed marker as depicted in the image. The tip of the ArUco marker mount should slant down into the workspace surface. F) Place glue on top of the ArUco marker mount 15 mm × 15 mm square surface. G) Adhere the paper-printed marker to the ArUco marker mount. H) Place a 4 in. cable tie into the side hole of one of the C-Clamps. I) Wrap the cable tie around the assembly, threading it through the second C-Clamp hole. J) Slide the cable tie through the marker mount so that the marker is facing away from the assembly. K) Close the cable tie so that it is tightly bound around the assembly. L) Cut the excess length of the cable tie.

2. Securing the I-Bracket to the animal

a. While the animal is anesthetized, permanently glue the I-Bracket around the exposed microelectrode array 16-channel Omnetics connector port within the animal's dental cement head cap (Figure 7). This can be achieved by applying sparing amounts of super glue around each side of the Omnetics connector before slotting on the I-Bracket.
Note: The rectangular center hole and height of the I-Bracket were specifically designed to surround our microelectrode array's connector port for accessible attachment of the tether to the animal. At the time of surgery, the dental cement head cap was molded to expose enough of the Omnetics connector to fit the I-Bracket at a later post-recovery date. Custom I-Brackets may need to be produced to fit

differing devices.

Troubleshooting: If the I-Bracket does not sit flush with the surface of the Omnetics connector, you can gently file down the PLA plastic using a small nail file.

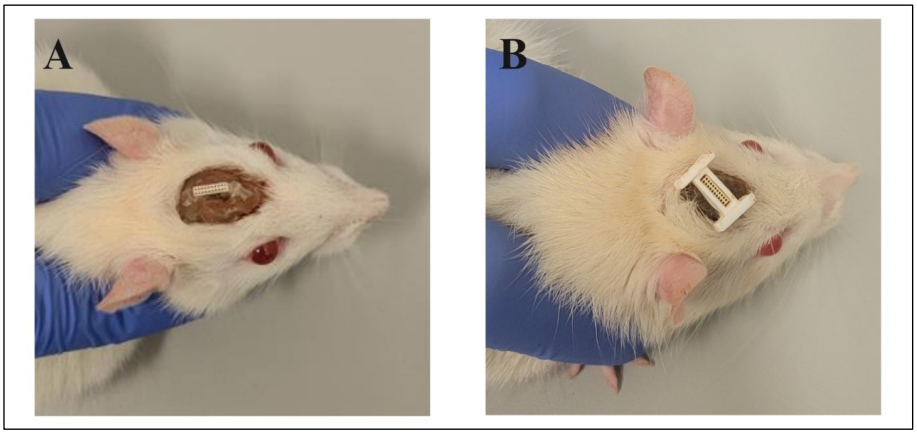


Figure 7. Permanently attaching the I-Bracket to a rat's head cap. A) An example of a rat that possesses an exposed Omnetics connector port without an I-Bracket attached. B) An example of a rat that has the I-Bracket super glued to the Omnetics connector port and dental cement head cap.

3. Mounting and dismounting procedures
   a. Follow the instructions outlined in Figure 8 for securely mounting and dismounting the ArUco marker hardware to an awake animal when running the behavioral task.
   Note: Steps should almost mirror that of Figure 6.
   b. Once the animal is properly tethered, close the operant conditioning door.
   c. For dismounting, carefully cut the cable tie between the C-Clamps with small scissors, then reverse the steps described in Figure 8.

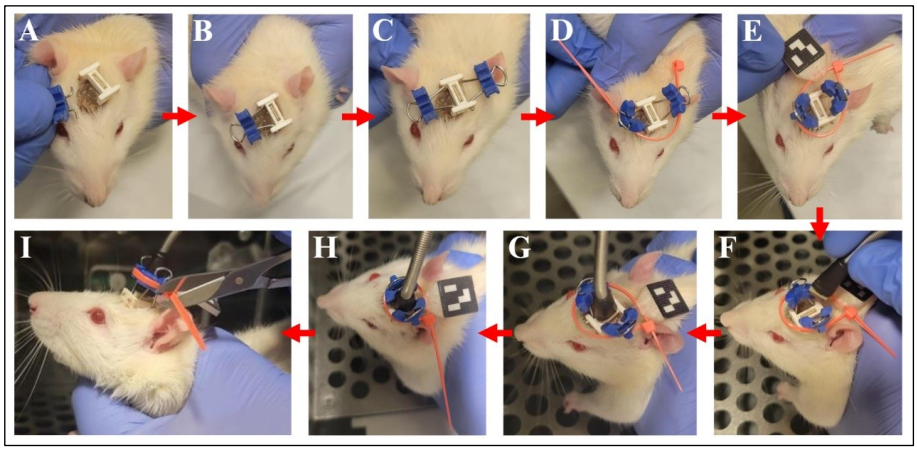


Figure 8. Instructions for mounting the ArUco marker assembly for behavior. A) With your index finger and thumb, slide the C-Clamp down the ⅝ in. binder clip wire loop until it almost reaches the protruding prongs at the end. B) While holding the animal still with your non-dominant hand, insert the prongs into the holes of the I-Bracket with your dominant hand and let go. C) Repeat the process described in panels A–B for the opposite side of the I-Bracket. D) Thread the 4 in. cable tie through the side holes of the C-Clamps, keeping the open end of the cable tie facing the animal's neck. E) Slide the cable tie through the marker mount so that the marker is facing away from the assembly and toward the center of the animal's neck. Then, barely close the cable tie so that it is loosely bound around the assembly. F) Hold the animal within the operant conditioning chamber and align the tether prongs with the microelectrode array

Omnetics port. G) Plug the tether into the Omnetics port. H) Tighten the cable tie so that it is tightly bound around the assembly, securing the position of the marker. The marker should remain aligned with the anterior to the posterior midline of the animal's neck as close as possible. I) Cut the excess length of the cable tie and release the animal.

D. Running the behavioral task and ArUco tracking code

1. Opening the code
   a. Without connecting the animal yet, power on the electrical stimulator and the oscilloscope.
   Caution: Make sure to always turn on the stimulator before plugging in the animal to the tether; turning on the device briefly passes current transients that can be seen on the oscilloscope.
   b. Open the medical-grade air tank + regulator valves to allow a small amount of low-pressure air to build at the pneumatic solenoid.
   c. Open the *BehaviorApp.mlapp* file from MATLAB as outlined in step B2f as well as the *AnimalDetect.py* file from Visual Studio Code as outlined in step B6m.
   d. In the App Designer window of MATLAB, click the green *Run* button located at the top of the screen. If you cannot see it, it is located within the *Editor* tab of the main header menu.
   e. Wait for the pop-up Vulintus code to finish loading and disappear; the behavior app should now be visible.
   f. Navigate back to Visual Studio Code and click the *Run Python File* play button that is located on the top right of the screen. The Python code will now attempt to connect to MATLAB and open individual windows for the ArUco marker tracking cameras.
   Troubleshooting: If an error occurs, please check that your cameras are functioning properly with the computer and that your *webcamlist* indexes from steps B6n–p are correct. If the error persists, wait a minute or two for MATLAB to finish initializing the behavior app, then click the *Run Python File* play button again. If Python still refuses to connect to MATLAB and says permission is denied, you may need to contact your administrator for internal host process port access.
2. Calibrating the cameras
   a. Check to make sure that the cameras are properly oriented so that the blue rectangles shown in each window cover half of the operant conditioning chamber, including the nose-poke and pellet dispenser modules. See Figure 9 for an example of proper camera alignment.

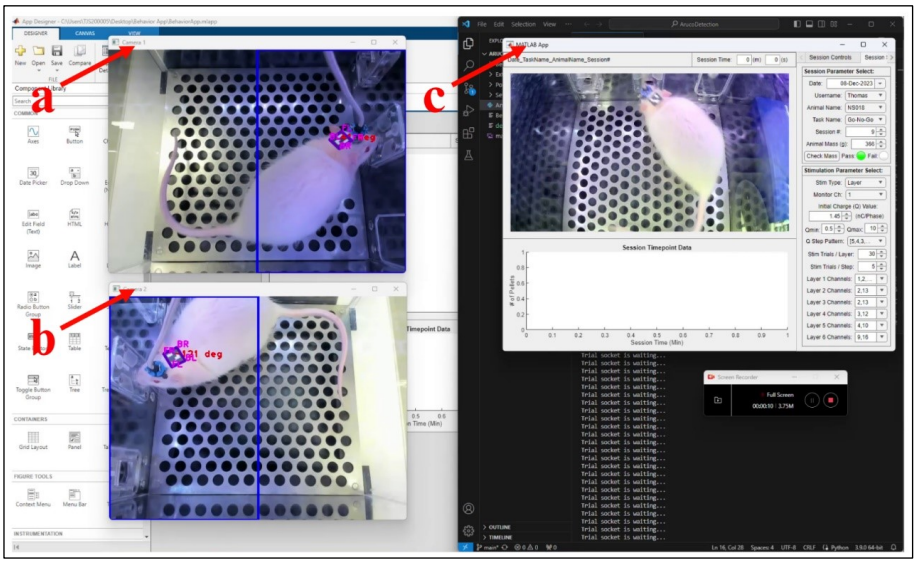


Figure 9. Examples of proper camera alignment for behavioral experiments. a) ArUco tracking camera #1 is mounted on the right side of the operant conditioning chamber and is aligned so that the blue rectangle passes through the midline of the chamber floor. The chamber modules should be located on the

right side of the window. b) ArUco tracking camera #2 is mounted on the left side of the operant conditioning chamber and is aligned so that the blue rectangle passes through the midline of the chamber floor. The chamber modules should be located on the left side of the window. c) The behavior camera is mounted on the front side of the operant conditioning chamber. The chamber modules should be located at the top of the window.

b. Place the ArUco marker mounting piece inside the operant conditioning chamber in different positions to test whether the marker is being properly tracked by the Python code. You should see a colored border around the marker and a red-degree angle listed within one or both tracking camera windows.
c. **Troubleshooting:** If the marker is not being tracked well, try the following: 1) clean the camera lenses and the operant conditioning chamber walls, 2) readjust the angle and height of the cameras so that their focus points are calibrated to the approximate rodent head height where the marker will likely be, 3) test the overall brightness of the chamber using a digital luxmeter (see Equipment section). The lux value within the chamber should read at least 3 lux [11], but the average is closer to ~25+ lux. If the value is too low, replace your RGB LED strip with a brighter one.

3. Testing the behavior code
   a. Once the applications are properly initialized and tracking the ArUco marker, begin selecting custom behavioral task and stimulation parameter settings within the *Session Setup* panel of the behavior app. Reference [13] regarding information behind the different behavioral task name selections.
   Note: For testing purposes, we suggest setting the following parameters: 1) Task Name → DetectAll, 2) Stim Type → Single, 3) Monitor Ch → 1, and 4) Initial Charge (Q) Value → 1 nC/Phase.
   b. Click back to the *Session Controls* tab within the app and click the *Start* button.
   c. Using your finger, begin interacting with the nose-poke module within the operant conditioning chamber, testing to see whether or not 1) a pellet is dispensed when poking after a stimulation trial, 2) an air puff is triggered when poking after a silent trial, and 3) that the RGB LED strip changes colors between white, green, and red depending on the trial state.
   d. Next, watch to see if a stimulus pattern is presented on the oscilloscope preceding a stimulation trial.
   Note: Since the animal is not plugged in, you may see a non-typical voltage transient pattern. Additional stimulation pattern parameters can be adjusted in the code under the Required Behavior Functions folder → ICMSandCapture.m file.
   e. Test to see if the *Feed* and *Pause* buttons work on the *Session Controls* panel of the behavior app.
   f. Finally, adjust the medical-grade air regulator to output the desired air pressure when an air puff is triggered.
   Troubleshooting: If any of the listed operations do not occur, please double-check your equipment setup.
   g. After testing has been completed, click the *Stop* button within the *Session Controls* panel of the behavior app. This will properly close both the MATLAB and Python applications.
4. Running experimental sessions with the behavior code
   a. Once the chamber has been fully tested and working as intended, you may connect the animal subject to the cage using the mounting/dismounting instructions outlined in step C3 and then repeat the steps listed above.
   Note: Specific behavioral task parameters can be modified by navigating to their corresponding tab within the behavior app. For the Stim Type option on the Session Setup panel, N/A = no stimulus presented to the animal, Single = only the channel specified by Monitor Ch., Layer = simultaneously stimulating the channels listed within the Layer 1 Channels drop-down selector, and All = simultaneously stimulating all 16 channels of the microelectrode array.

E. Locating post-session results

1. Behavioral performance data
   a. After a behavioral session has concluded and ended using the *Stop* button, an Excel file containing the raw behavioral data for each individual trial is saved into the *ArucoDetection → MATLAB*

Behavior Program → *ExcelSessionData* → *SessionDataSheets* folder. Locate this folder and open the specific Excel session file just produced.

b. The Excel file contains MATLAB variable data that was automatically generated during the behavioral session using the nose-poke sensor module, native MATLAB *clock* and array indexing functions, and the ArUco tracking program, all without the need for manual data recording. To find a particular data type of interest, navigate through each of the different sheets within the Excel file: *Sheet 1* includes a summary of the general session details, *Sheet 2* includes parameter details about each of the individual trials that were presented, and *Sheet 3* includes arrays for reproducing the *Session Timepoint Data* plot that was presented and updated in the behavior app after every trial. See Tables 1–3 for examples of each sheet.
Note: During each behavioral session, a 0.15-s delay follows the presentation of any stimulus to prevent the animal from coincidentally nose-poking during stimulus delivery. The reaction times recorded in Table 2 exclude this delay, allowing for flexibility if the user adjusts the delay duration. For accurate post-session analysis based on reaction time, the delay duration should be added to the final values.

Table 1. Exported behavioral session data found in Sheet 1. This table presents an example of raw data generated by the *BehaviorApp.mlapp* program in MATLAB that was exported to an Excel file. It includes general session summary values used for assessing behavioral performance, with all trials included. This data provides a comprehensive overview of session metrics, allowing for detailed analysis without pre-filtering or exclusion of trials.

Sheet 1	
Task Name	Go-No-Go
Researcher Name	Thomas
Session Number	17
Session Date	11-Jan-23
Animal Name	NS008
Animal Weight (g)	494
Stim Type	All
Session Pellets	128
Manual Pellets	0
Total Pellets	128
Accuracy Score (%)	93.66515837
Stimulus Trials	111
Silent Trials	110
Hits on Stim Trials	98
Misses on Stim Trials	10
Late Hits on Stim Trials	3
Hits on Silent Trials	1
Misses on Silent Trials	109
Catch Trials	281
Hits on Catch Trials	0
Total Timeout Time (s)	72
Initial Session Time (s)	63299.29
Ending Session Time (s)	66900.93
Total Pause Time (s)	0

Table 2. Shortened example of the exported behavioral trial-specific data found in Sheet 2. This table displays data from 15 sequential trials within an example behavioral session that was generated in MATLAB and exported to Excel using the *BehaviorApp.mlapp* program. It includes the raw values for each individual trial, providing a detailed record of the animal's responses and the corresponding session metrics. This trial-specific data is essential for analyzing the progression and consistency of behavior throughout the session.

Sheet 2										
Trial	Trial Type (Stim=1, Silent=2, AltStim=3)	Distraction Annotation (active=0, sleep=1, distract=2)	Stim Trial	Silent Trial	Alt Stim Trial	Response (Miss=0, Hit=1, Late=2)	Reaction Time (s)	Stimulus Value (nC/Phase or kHz)	Layer	Monitor Channel
1	1	0	1	0	0	1	0.016	3	0	1
2	2	0	0	1	0	0	0	0	0	1
3	1	0	2	0	0	1	0.018	3	0	1
4	3	0	0	0	1	1	0.018	3	0	1
5	2	0	0	2	0	0	0	0	0	1
6	2	0	0	3	0	0	0	0	0	1
7	1	0	3	0	0	1	0.018	3	0	1
8	1	0	4	0	0	1	0.025	3	0	1
9	2	0	0	4	0	0	0	0	0	1
10	1	2	5	0	0	0	0	3	0	1
11	1	0	6	0	0	1	0.018	3	0	1
12	2	0	0	5	0	0	0	0	0	1
13	3	0	0	0	2	1	0.184	2.044	0	1
14	1	0	7	0	0	1	0.033	3	0	1
15	1	0	8	0	0	1	0.014	3	0	1

Table 3. Shortened example of the exported behavioral plot data found in Sheet 3. This table presents data from 15 sequential trials (as shown in Table 2) within an example behavioral session that was generated in MATLAB and exported to Excel via the BehaviorApp.mlapp program. It includes raw plot values for each trial type—stimulation, silent, and alternative stimulation—along with their corresponding X and Y-axis values, listed in order of presentation. The session comprised eight stimulation trials, five silent trials, and two alternative stimulation trials.

Sheet 3					
Stim X-Axis (min)	Stim Y-Axis (s)	Sil X-Axis (min)	Sil Y-Axis (s)	AltStim X-Axis (min)	AltStim Y-Axis (s)

0.138583333	0.016	0.397966667	0	21.3667	0.022
0.540683333	0.018	1.008183333	0	23.99553333	0.184
1.336533333	0.018	1.183083333	0	-	-
1.57115	0.025	1.829566667	0	-	-
2.072966667	0	2.474333333	0	-	-
2.2156	0.018	-	-	-	-
2.854116667	0.033	-	-	-	-
3.087666667	0.014	-	-	-	-

2. Stimulation waveform data
   a. To find records of the individual voltage transients produced from each stimulation trial, navigate to the *ArucoDetection → MATLAB Behavior Program → StimDataOutput* folder.
   b. Within that folder, a list of additional folders will be produced for each behavioral session that was run. Open the folder specific to a behavioral session that just concluded and here you will find two copies of each stimulus: 1) a MATLAB data file containing the points needed to reproduce an individual waveform and 2) a .png file containing the graphed image of that waveform.
   Note: Only the voltage waveforms are saved to these files and not the current. Since the generated pulse is current-controlled, only the voltage waveforms are needed.
3. ArUco engagement data
   a. The data to determine how well the ArUco marker tracking program scored behaviors of distraction versus engagement can be found within the same session data sheets described in step E1. Open one of the session sheets and navigate to *Sheet 2*. Here, the *Distraction Annotation* column within that sheet contains the classification of each individual trial with a value of "0" indicating engagement and a "2" indicating distraction (see Table 2 for an example).
   Note: This column does not contain ground-truth data. Compare these results to human-annotated data to quantify performance metrics.
4. ArUco tracking performance data
   a. After a behavioral session concludes, the *AnimalDetect.py* ArUco tracking code in Visual Studio Code will produce a text file called *detection_results.txt* within the same directory. Open this file to view the number of total video frames in which the ArUco marker was detected by camera #1 exclusively, camera #2 exclusively, both cameras, or neither.
   Caution: This file is overwritten after each behavioral session.

F. Verification of ArUco tracking setup

1. Once the behavioral chamber is fully operational, the user can evaluate the performance of the ArUco tracking setup by calculating accuracy metrics, comparing the automated scoring program against human-generated annotations. To do this, the user must manually score each trial of a behavioral session, marking whether the animal was "engaged" or "distracted."
   Note: Screen recording programs can assist with human annotations.
2. Afterward, open the corresponding Excel session file and review the *Distraction Annotation* column in *Sheet 2* (see Table 2 for an example). Against your annotations, tally true positives (both human and program agree on engagement), true negatives (both agree on distraction), false positives (program marks engagement, human marks distraction), and false negatives (program marks distraction, human marks engagement) to fill out the components of a confusion matrix.
3. Once generated, the custom MATLAB file found at *ArucoDetection → Analysis Code → Model_Classification_Metrics.m* can be used to calculate the following accuracy metrics: accuracy, precision, sensitivity, specificity, F1-score, d-prime, and Matthew's correlation coefficient (MCC). Upon starting the program, it will prompt the user to input their four confusion matrix values (one at a time), then output the accuracy metric values as text within the terminal.
   Note: We leave it up to the user to determine satisfactory accuracy metrics. However, we suggest having an overall accuracy of at least 90% for robust performance. As a benchmark, we previously achieved an

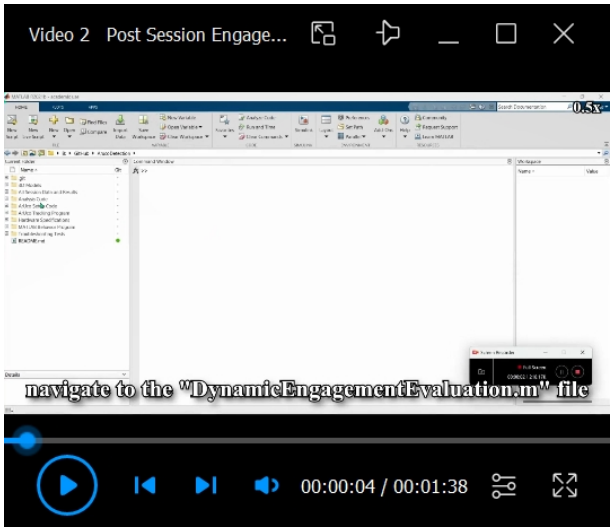
overall accuracy of 98% [14].

Troubleshooting: If the expected accuracy value was not met, check the total number of ArUco marker-identified frames (described in step E4). This should inform the user if one or both cameras are underperforming in terms of consistently tracking the marker. If sufficient frames are being detected, see step D2 for further information about recalibrating the cameras.

Data analysis

A full description of the data analysis procedures can be found within the *Materials and Methods* section for the following research articles: 1) Smith et al., Real-Time Assessment of Rodent Engagement Using ArUco Markers: A Scalable and Accessible Approach for Scoring Behavior in a Nose-Poking Go/No-Go Task, *eNeuro* [14], and 2) Smith et al., Behavioral Paradigm for the Evaluation of Stimulation-Evoked Somatosensory Perception Thresholds in Rats, *Frontiers in Neuroscience* [13].

1. Automated scoring of engagement
   The automated assessment and scoring of animal engagement are processed in real-time during each behavioral session using the *ArucoDetection → ArUco Tracking Program → AnimalDetect.py* program. This program continuously evaluates the ArUco marker's position and orientation to determine whether the animal is "engaged" or "distracted" during any individual trial. Specifically, if the marker is detected within the region of interest—from the chamber's midline to the wall containing the nose-poke and reward modules—and oriented toward that wall (yaw angle 5°–175°; 90° indicating orthogonality between the marker's edge and wall), then the animal is scored as "engaged"; otherwise, it is scored as "distracted." Furthermore, to be considered "engaged", the detected ArUco marker must only meet the listed requirements for a single captured frame. However, for a trial to be scored as distracted, the animal must be distracted throughout the entire trial duration. The results of this scoring process can be found in the *Distraction Annotation* column data from *Sheet 2* of the exported Excel session data sheet (see step E3).
2. Post-session engagement analysis
   Only after the user is satisfied with the ArUco classification performance metrics outlined in section F, should they proceed with the overall post-session behavioral engagement analysis. This process involves combining the *Distraction Annotation* column data from *Sheet 2* of the exported Excel session data sheet (see Table 2) with the x-axis trial values from *Sheet 3* (see Table 3) to create a sequential array of trial response data. The data is then analyzed using a rectangular kernel convolution to detect critical transition points between engagement and distraction, identifying the time point where the probability of engagement drops below 50%, thereby determining task disengagement and an optimal task duration for sustained engagement. For this analysis, use the custom MATLAB file found within *ArucoDetection → Analysis Code → DynamicEngagementEvaluation.m* to evaluate engagement over time in the nose-poking go/no-go behavioral task. To utilize the file, simply run the code in MATLAB and type in the duration of the session being analyzed. Then, when prompted, select the behavioral session file that needs to be analyzed from the *ArucoDetection → MATLAB Behavior Program → ExcelSessionData → SessionDataSheets* folder. Once the file is fully processed, a MATLAB plot showing the probability of engagement over time will appear. Inside of this plot, the time point at which animal disengagement is calculated will be shown as a blue vertical line. Furthermore, the probability of engagement array values for the session will be displayed in the terminal window as text. See Video 2 for an example of how to run this program effectively. Note that, when using this program to determine a maximum session duration, this analysis should be averaged across multiple sessions and animals to inform of this decision. However, this program can also be used to further define individual session exclusion criteria; in addition to the ArUco tracking program's standard distraction labeling, the user may choose to exclude all trials following the point where the animal was deemed completely disengaged.



Video 2. Running the post-session engagement analysis program. Instructional video that outlines the steps required to process the exported behavioral session data sheets for determining task disengagement and optimal task durations based on the automated scoring of animal engagement data.

3. Routine behavioral analysis
   Alongside engagement evaluations, behavioral task performance can also be routinely analyzed through post-session programs. Although the standard behavioral program does automatically calculate the overall accuracy for each session, it does not account for the exclusion of distracted trials. To calculate the adjusted accuracy, the user must tally the number of distracted trials for each session type (stim, silent, or alt-stim), then subtract those values from the "missed" category of each session type, providing the updated confusion matrix values: true positives (hits on stim trials), true negatives (misses on silent trials), false positives (hits on silent trials), and false negatives (misses + late hits on stim trials) found within the exported Excel session file following each session. The location of the dataset required for producing this confusion matrix is described in step E1 (see Table 1 for an example). Once generated, the custom MATLAB file found at *ArucoDetection → Analysis Code → Model_Classification_Metrics.m* can be used to calculate the following revised accuracy metrics: accuracy, precision, sensitivity, specificity, F1-score, d-prime, and Matthew's correlation coefficient (MCC). Upon starting the program, it will prompt the user to input their four confusion matrix values (one at a time), then output the accuracy metric values as text within the terminal.

Validation of protocol

This protocol or parts of it has been used and validated in the following research article(s):

- Smith et al. [13]. Behavioral Paradigm for the Evaluation of Stimulation-Evoked Somatosensory Perception Thresholds in Rats. *Frontiers in Neuroscience*. (Figures 1, 3, and 6).
- Smith et al. [14]. Real-Time Assessment of Rodent Engagement Using ArUco Markers: A Scalable and Accessible Approach for Scoring Behavior in a Nose-Poking Go/No-Go Task. *eNeuro*. (Figures 1–3).

1. Validation of hardware components
   Through combined research efforts across multiple studies, including Smith et al. [13,14], the behavioral chamber has been utilized in over 500 sessions, totaling more than 250 h of robust performance, enduring the animals' pulling, biting, and scratching. During this time, the only component that required replacement was the tether, which was replaced three times. Additionally, the ArUco assembly pieces have been used in over 300 sessions, accumulating more than 150 h of use, with the only replacement being the paper-printed ArUco pattern, which was reprinted twice.

2. Validation of ArUco engagement assessment and scoring
   Smith et al. [14] validated the ArUco engagement assessment and scoring procedures described in this protocol, demonstrating the system's accuracy and reliability. In this study, the same ArUco tracking system described in this protocol achieved an overall classification accuracy of 98.3%, precision of 98.5%, sensitivity of 99.7%, specificity of 70.8%, and F1-score of 99.1% when compared with human annotations across ~1,000 individual trials. These results indicated strong classification performance when applied during the go/no-go behavioral task featured in the *BehaviorApp.mlapp* program.

To further validate the use of this protocol without needing to set up the equipment, researchers can reproduce the findings of the 2024 article [14] by accessing the raw behavioral session data within the relevant GitHub repository (https://github.com/tomcatsmith19/ArucoDetection) under the folder *ArucoDetection → All Session Data and Results*. The raw data can then be analyzed using the provided analysis code within the *ArucoDetection → Analysis Code* folder. The corresponding videos for these sessions can be provided upon request.

General notes and troubleshooting

General notes

1. Automation and confirmation biases
   The overall behavioral system was designed to automatically score and record consistent variables such as nose-poking instances, timestamps, electrical stimulation outputs, and estimates of animal engagement. These automated processes help to eliminate confirmation biases during the data collection phase, especially when distinguishing trials of engagement from distraction. This ensures a consistent evaluation approach applicable for large-scale studies involving multiple human evaluators. However, while confirmation biases are minimized during data collection, additional biases may arise during post-session analysis if the exported data is manually edited by the user before being processed in the provided analysis code or outside sources.
2. System setup and flexibility
   The behavioral chamber setup described in this protocol was designed as a flexible starting point rather than a rigid framework. Although step-by-step instructions for reproducing our exact chamber are provided, we understand that each setting may require unique specifications like differing stimulators or stimulation patterns, microelectrode array connectors, number of cameras, etc. Therefore, on the software side, all MATLAB and Python code has been commented or described within this protocol in hopes of easing the customization process with the bulk of the complete program split between the two main files: MATLAB → *BehaviorApp.mlapp* and Python → *AnimalDetect.py*. Taking this one step further, the base ArUco tracking components can be extracted from the behavior app, leaving researchers with the ability to insert it into their own configurations. From the *BehaviorApp.mlapp* file, the only code that is required to receive distraction results from the provided *AnimalDetect.py* file are the following isolated commands:

```
% Server socket setup with Python code for ArUco marker detection
import java.net.ServerSocket
    import java.io.*
    server_socket = ServerSocket(9999);
    client_socket = server_socket.accept;
    input_stream   = client_socket.getInputStream;
    d_input_stream = DataInputStream(input_stream);
    output_stream   = client_socket.getOutputStream;
    d_output_stream = DataOutputStream(output_stream);

% Send a 1 to Python script to begin ArUco maker detection
d_output_stream.writeUTF(num2str(1));
```

```
% Send Python script a 2 to stop tracking ArUco markers
d_output_stream.writeUTF(num2str(2));

% Receive ArUco marker distraction results from Python script
data_received = str2double(d_input_stream.readUTF());

% Close all ArUco maker detection tracking and Python code.
% Disconnect the server socket and clear variables.
d_output_stream.writeUTF(num2str(4));
pause(5);
input_stream.close;
d_input_stream.close;
client_socket.close;
server_socket.close;
```

However, if Python is the only desired language, then extract the socket class code from the *AnimalDetect.py* file. An example of such code in which an ArUco marker is continuously tracked until the key 'q' is pressed can be seen below:

```
import cv2
import numpy as np

font = cv2.FONT_HERSHEY_PLAIN
dictionary = cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_4X4_50)

# camera setup
cap = cv2.VideoCapture(1)
cap.set(cv2.CAP_PROP_FOCUS, 20)
cap.set(cv2.CAP_PROP_ZOOM, 0)
print("Camera connected")

while True:
    # Read frame from the camera
    success, frame = cap.read()
    if not success:
        break

   # Detect ArUco markers in the frame
   corners,   marker_ids,   rejected   =   cv2.aruco.detectMarkers(frame,
dictionary)

   # if the marker was found
   if corners:
        for corner, marker_id in zip(corners, marker_ids):
            corner = corner.reshape(4, 2)
            corner = corner.astype(int)

            top_right, top_left, bottom_rght, bottom_left = corner

            # Calculate radian yaw angle of the pose
            delta_pos = top_left - top_right
            yaw = np.degrees(np.arctan2(delta_pos[1], delta_pos[0]))
```

```
                # Calculate centroid of the marker
                centroid = np.mean(corner, axis=0).astype(int)

                # Print yaw value in the center of the marker
                yaw_text = f"{int(yaw)} deg"
                cv2.putText(frame,yaw_text,tuple(centroid),font,1.3,(0,0,255)
            ,2)

                # Draw marker outline and label corners
                cv2.polylines(frame,[corner],True,(255,0,255),3,cv2.LINE_AA)
                cv2.putText(frame,"FL",tuple(top_right),font,1.3,(255,0,255),
            2)
                cv2.putText(frame,"FR",tuple(top_left),font,1.3,(255,0,255),2)
                cv2.putText(frame,"BR",tuple(bottom_rght),font,1.3,(255,0,255
            ),2)
                cv2.putText(frame,"BL",tuple(bottom_left),font,1.3,(255,0,255
            ),2)

        # show camera frame with ArUco tracking
        cv2.imshow("Camera", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# close camera stream
cap.release()
cv2.destroyAllWindows()
```

On the hardware side, most items such as the sound-attenuating chamber, operant conditioning chamber, T-Slot rails, commutator, camera mounts, and the cameras themselves can each be substituted without affecting the application. In addition, the ArUco marker simply needs to be flat, fixed to the animal's head in the orientation displayed within Figure 9, and hold a direct line-of-sight with the cameras. If the reader is unable to utilize the ArUco mounting hardware in step C1, the reader may investigate other methods such as water-soluble glue or a custom helmet to rigidly and robustly fix the marker to the animal's head. In terms of electronics, devices meeting the criteria in step A9a will also be viable. Furthermore, if an existing setup utilizes a 3.3V logic microcontroller a level shifter may be utilized to shift the logic voltage to 5V. This overall flexibility enables researchers to tailor the setup precisely to their experimental setup without being constrained by a predefined configuration.

3. Custom behavioral task integration

   To extend the functionality of the pre-loaded MATLAB behavior app with custom behavioral tasks, the code utilizes a structured approach within a switch case framework (lines 550–12388 from the *BehaviorApp.mlapp* file). Here, each behavioral task is encapsulated within its own case statement, facilitating direct modular expansion. By adding a new case to the switch statement and including the name of that case (Task Name) in the dropdown list located within the *Session Setup* tab → *Task Name* dropdown box of the app (see procedure steps B2f–i), researchers can seamlessly integrate additional behavioral tasks into the application. The existing code provides a template and examples from the pre-loaded tasks, offering guidance for developing and integrating new tasks. This approach ensures that the MATLAB behavior app remains adaptable and scalable, supporting the incorporation of novel experimental paradigms.

Acknowledgments

This work was supported in part by the National Institutes of Health, National Institute for Neurological Disorders and Stroke (R01NS110823, GRANT12635723, J.R.C. and J.J.P.), diversity supplement to parent grant (A.G.H-R.), a Research Career Scientist Award (GRANT12635707, J.R.C.) from the United States (US) Department of Veterans Affairs Rehabilitation Research and Development Service, and the Eugene McDermott Graduate Fellowship from The University of Texas at Dallas (202108, T.J.S.). This protocol was derived from the original research paper "*Real-Time Assessment of Rodent Engagement Using ArUco Markers: A Scalable and Accessible Approach for Scoring Behavior in a Nose-Poking Go/No-Go Task*" [14].

Competing interests

The authors declare no competing financial interests.

Ethical considerations

All animal handling, housing, and procedures were approved by The University of Texas at Dallas IACUC (protocol #21-15) and in accordance with ARRIVE guidelines [15].

References

1. Marsh, D. M. and Hanlon, T. J. (2007). Seeing What We Want to See: Confirmation Bias in Animal Behavior Research. *Ethology.* 113(11): 1089–1098.
2. von Ziegler, L., Sturman, O. and Bohacek, J. (2020). Big behavior: challenges and opportunities in a new era of deep behavior profiling. *Neuropsychopharmacology.* 46(1): 33–44.
3. Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W. and Bethge, M. (2018). DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat Neurosci.* 21(9): 1281–1289.
4. Moro, M., Marchesi, G., Hesse, F., Odone, F. and Casadio, M. (2022). Markerless vs. Marker-Based Gait Analysis: A Proof of Concept Study. *Sensors.* 22(5): 2011.
5. Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M. and Mathis, M. W. (2019). Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nat Protoc.* 14(7): 2152–2176.
6. Pereira, T. D., Tabris, N., Matsliah, A., Turner, D. M., Li, J., Ravindranath, S., Papadoyannis, E. S., Normand, E., Deutsch, D. S., Wang, Z. Y., et al. (2022). SLEAP: A deep learning system for multi-animal pose tracking. *Nat Methods.* 19(4): 486–495.
7. Vagvolgyi, B. P., Jayakumar, R. P., Madhav, M. S., Knierim, J. J. and Cowan, N. J. (2022). Wide-angle, monocular head tracking using passive markers. *J Neurosci Methods.* 368: 109453.
8. Liang, G., Chen, F., Liang, Y., Feng, Y., Wang, C. and Wu, X. (2021). A Manufacturing-Oriented Intelligent Vision System Based on Deep Neural Network for Object Recognition and 6D Pose Estimation. *Front Neurorobot.* 14: e616775.
9. Menolotto, M., Komaris, D. S., Tedesco, S., O'Flynn, B. and Walsh, M. (2020). Motion Capture Technology in Industrial Applications: A Systematic Review. *Sensors.* 20(19): 5687.
10. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.* 47(6): 2280–2292.
11. Hu, D., DeTone, D. and Malisiewicz, T. (2019). Deep ChArUco: Dark ChArUco Marker Pose Estimation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 8428–8436.

12. Sampathkrishna, A. (2022). ArUco Maker based localization and Node graph approach to mapping. *cs.RO.* doi.org/10.48550/arXiv.2208.09355.
13. Smith, T. J., Wu, Y., Cheon, C., Khan, A. A., Srinivasan, H., Capadona, J. R., Cogan, S. F., Pancrazio, J. J., Engineer, C. T., Hernandez-Reynoso, A. G., et al. (2023). Behavioral paradigm for the evaluation of stimulation-evoked somatosensory perception thresholds in rats. *Front Neurosci.* 17: e1202258.
14. Smith, T. J., Smith, T. R., Faruk, F., Bendea, M., Tirumala Kumara, S., Capadona, J. R., Hernandez-Reynoso, A. G. and Pancrazio, J. J. (2024). Real-Time Assessment of Rodent Engagement Using ArUco Markers: A Scalable and Accessible Approach for Scoring Behavior in a Nose-Poking Go/No-Go Task. *eNeuro.* 11(3): ENEURO.0500–23.2024.
15. Percie du Sert, N., Ahluwalia, A., Alam, S., Avey, M. T., Baker, M., Browne, W. J., Clark, A., Cuthill, I. C., Dirnagl, U., Emerson, M., et al. (2020). Reporting animal research: Explanation and elaboration for the ARRIVE guidelines 2.0. *PLoS Biol.* 18(7): e3000411.