Autoinjector 2.0 User Manual

Version: 1.0

Date: 2022-12-15

Author: Jacob O'Brien

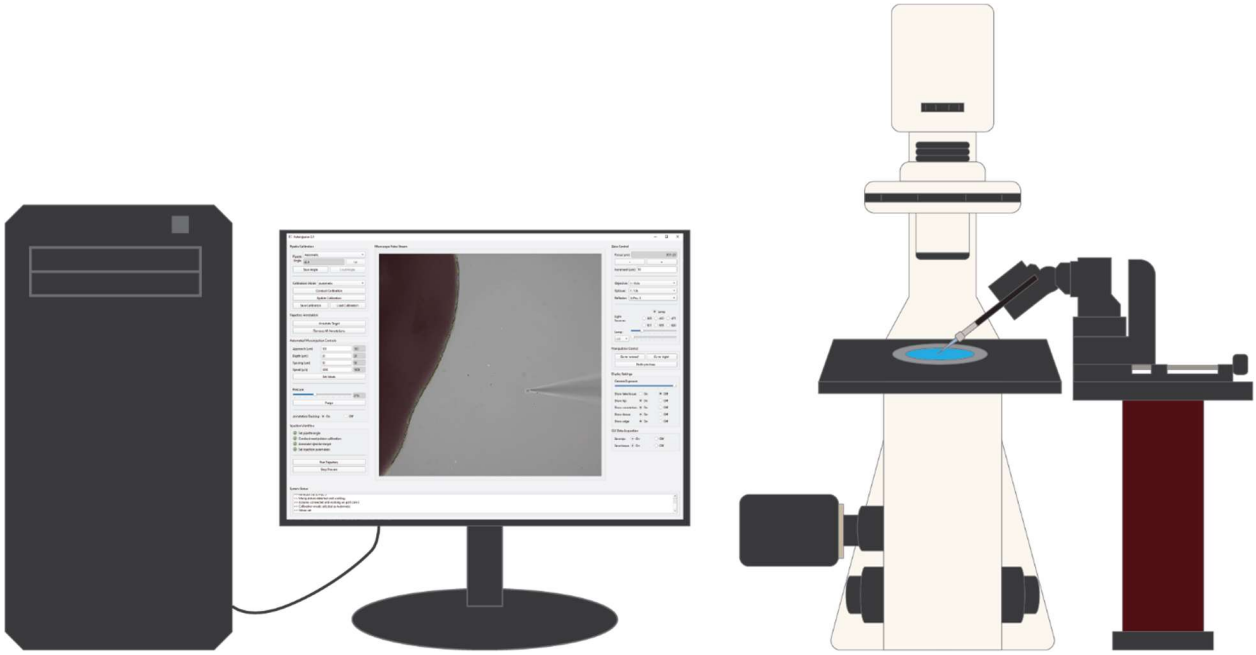


Table of Contents

1 Introduction

Manual microinjection is a challenging and tedious task that is as much an art as it is a science. To conduct manual microinjection, a technician performs a series of nuanced steps to manipulate a tiny needle to target individual cells for injection. For instance, the technician starts by using a pipette puller to create glass capillary needles with a specific morphology for the targeted cell population. Then they carefully fill the glass capillary needle with the injection solution. Next, they insert the needle into the manipulator, and they navigate the needle next to the tissue to be injected without colliding with the tissue or other debris. Finally, they attempt injections by inserting the needle into the tissue, pressurizing the needle to inject the solution, retracting the needle, readjusting the needle to a new position, and repeating this process for numerous cells. All the while, the technician tries to maintain the same exact depth, duration, and pressure for each injection to ensure uniform experimental conditions for all injected cells. This is not to mention that the needle will frequently clog which requires the technician to perform this whole series of steps over-and-over!

Autoinjector 2.0 aims to ease the microinjection burden by removing some of technical responsibilities from the user and passing these responsibilities onto a robotic system. Certainly, some of the tasks must still be performed by the user like making glass capillary needles, filling the needle, and initially navigating to the tissue. However, other tasks can be entirely performed by the robotic system, or at least made easier. With Autoinjector 2.0, you can perform experiments that were previously unfeasible because of the difficulty and slowness of manual microinjection. Moreover, it's even faster than its predecessor (Autoinjector 1.0) and includes greater functionality to automate your microinjection experiments.

My aim for this document is to provide you with the necessary information for understanding how the Autoinjector 2.0 works, and how to use it for microinjection experiments. In section **2**, I overview the hardware and software of the Autoinjector 2.0, briefly describe the microinjection strategy, and compare Autoinjector 2.0 to version 1.0. Then section **3** describes the principles of robotic microinjection with Autoinjector 2.0 by going slightly more in-depth to the robotic microinjection strategy. After this, in section **4**, I list the step-by-step guidance for conducting automated microinjections. Sections **2** and **3** give you an understanding for the "what" and the "why" of robotic microinjections while section **4** gives you the "how". I highly suggest reading all three of these sections, but if you are simply trying to run microinjection experiments then section **4** is all you need. After the step-by-step guidance, I provide some information on troubleshooting in section **5**. Finally, the interested reader can refer to the appendices for more

technical information on the Autoinjector 2.0 software and details about its various automated processes. I think everyone would find **APPENDIX A**, **APPENDIX B**, and **APPENDIX C** useful where I describe some pipette program parameters and explain every widget in the Autoinjector 2.0 applications.

2 Autoinjector 2.0 Description

Autoinjector 2.0 is an automated and vision guided system for injecting single cells in organotypic slices. It has been tested on 200µm – 300µm slices of embryonic mouse brain tissue and human brain organoids. Autoinjector 2.0 is an updated iteration of Autoinjector 1.0 that introduces new features to improve microinjection automation and system usability. Specifically, Autoinjector 2.0 includes pipette tip detection for automatic calibration, tissue detection for automatic target annotation, robotic 3D (rather than just 2D) injection targetting, target tracking for accurate microinjections of moving injection targets, and high-pressure pipette unclogging.

2.1 Autoinjector 2.0 hardware configuration

Autoinjector 2.0 consists of four main components: a computer for controlling the system, an inverted microscope for visualizing the injections, an injector/manipulator for physically conducting the injections, and pressure controller for applying positive pressure to deliver the injection payload. A pictorialized schematic of the Autoinjector 2.0 is shown in Figure 1. As shown, the computer connects to all other components so it can send and receive commands between devices, and the pressure controller modulates the air supply before delivering positive pressure to the manipulator/micropipette for injections.

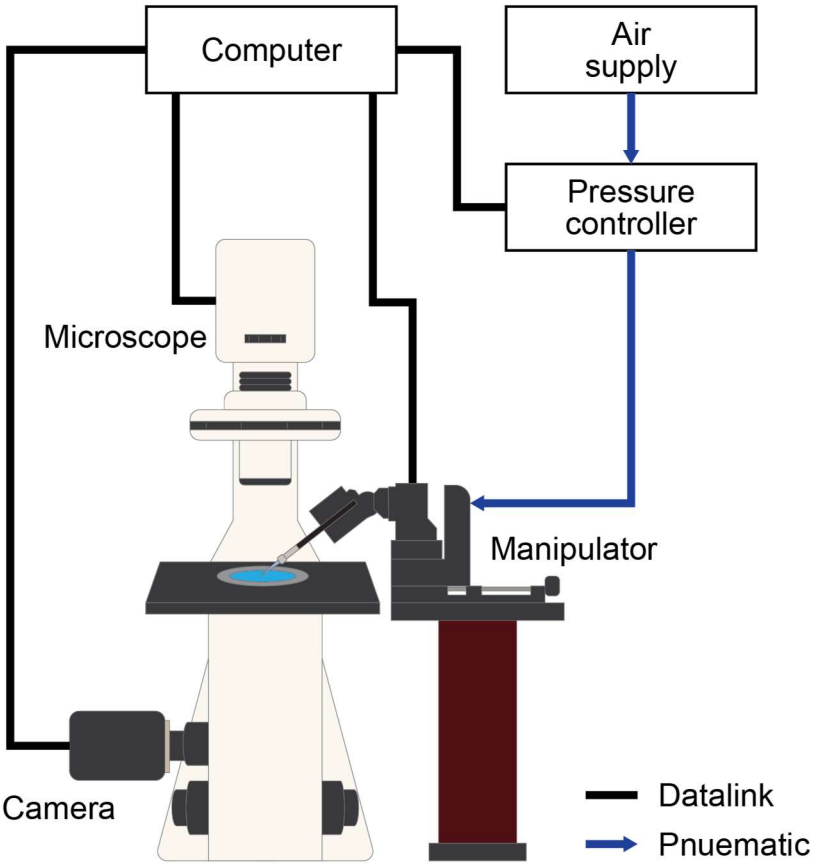


Figure 1. Pictorial schematic demonstrating the main Autoinjector 2.0 components.

A more detailed schematic of the Autoinjector 2.0 is shown in Figure 2. The computer coordinates between two software packages simultaneously: the custom Autoinjector software (section **2.2**) and the microscope software. The Autoinjector software provides a user interface and allows the user to control the system. The microscope software handles direct communication with the majority of the motorized/automated microscope components including the objective changer, optovar changer, reflector changer, focus controller, stage controller, LED controller, and lamp controller. Both software packages interface with each other; this permits the Autoinjector software to indirectly control the automated/motorized microscope components through the microscope software. The camera is the only microscope component that directly interfaces with the Autoinjector software. This allows the Autoinjector software to attain a live video stream for displaying the injections.

The injector consists of a motorized manipulator and a micropipette. The manipulator holds the micropipette (which is the injection needle), and it moves the micropipette in 3D space with nanometer resolution. The motorized manipulator is controlled by the Autoinjector software to

conduct injections. The motorized manipulator is a Sensapex uMp-4 and it has 4-axes of movement: x, y, z, and d. The x, y, and z axes are standard co-perpendicular axes that are capable of 3D movement. The d-axis is located along the micropipette's axis (in the x-z plane), and it enables the manipulator to move the micropipette in and out along the micropipette's axis which is desirable while injecting cells.

The pressure controller interfaces with the Autoinjector software through a microcontroller (an Arduino Uno). This microcontroller subsequently controls an electronic pressure regulator and a solenoid valve. As the air supply passes through the pressure controller, it first encounters a mechanical regulator to reduce the air pressure to the operational level of the electronic pressure regulator (3-3.5psi). The electronic pressure regulator further reduces the pressure to the desired injection pressure (0-2psi) as designated by the user using the Autoinjector software. Finally, the solenoid valve acts as an electronic gate to allow/disallow the pressure from being sent the micropipette containing the injection solution. An alternative pathway allows the high-pressure air supply to pass directly to the solenoid valves which permits high-pressure unclogging for the micropipette. More information on the pressure controller is available in **APPENDIX H**, and more information on the unclogging is available in section **3.3.2**.

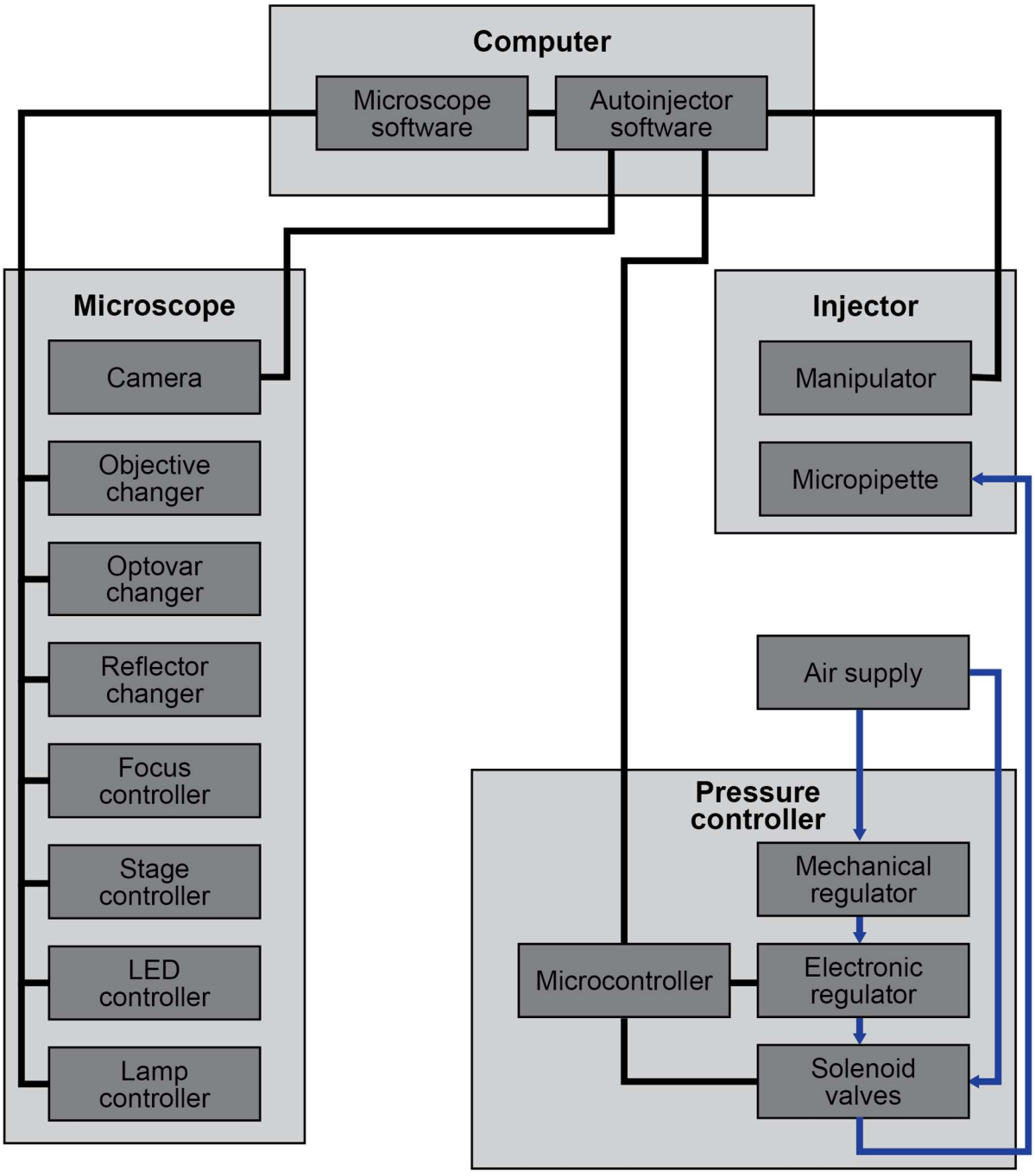


Figure 2. Detailed schematic of Autoinjector 2.0 displaying the main components and their subcomponents. The schematic also shows how the software coordinates between various devices.

2.2 Autoinjector 2.0 software

The Autoinjector 2.0 software is how the user interacts with the robotic system. It provides a graphical user interface (GUI) that enables the user to configure the Autoinjector system and run microinjection experiments. The software consists of two main applications: the configuration application ("configure_autoinjector.py" Python file that can be launched with the

“configuration.ps1” PowerShell script) and the main Autoinjector application (“application_minimal.py” Python file that can be launched with the “autoinjector.ps1” PowerShell script). The configuration application allows the user to specify several parameters that are necessary for Autoinjector 2.0 operations. The main application is the primary user-interface that allows the user to control the Autoinjector 2.0 system and run microinjection experiments. These two applications are detailed in the following sections.

2.2.1 Configuration application

The configuration application provides a user interface that enables the user to specify several important parameters that dictate how the Autoinjector system runs. The parameters define which hardware to interface with and some important software settings. The Autoinjector system will not work properly without “knowing” these parameters, so these parameters must be specified first before ever running the main application. Typically, the configuration application only needs to be run during the first-time startup of the system, but it can also be used to redefine system parameters in case you change the system hardware.

The configuration application is shown in Figure 3. The user can hover their mouse over a widget (interactable item) to show a brief description of that parameter, and the “?” will display more detailed information about that parameter. A more detailed explanation of the configuration application is available in section **APPENDIX B**.

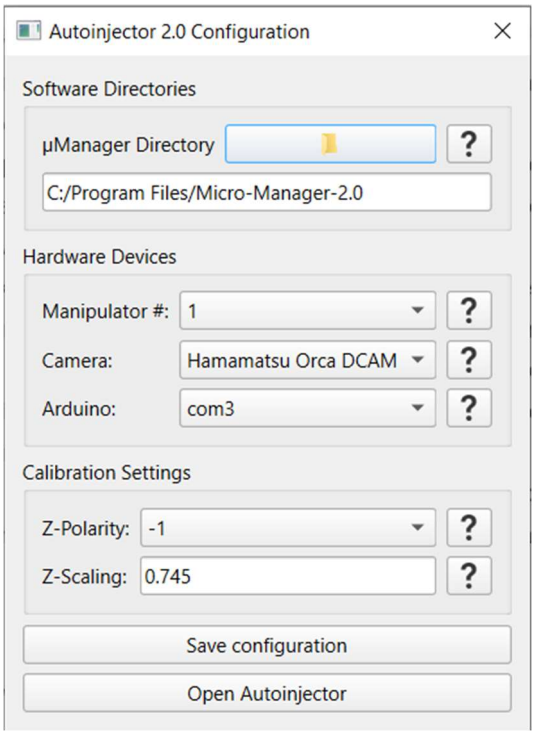


Figure 3. Autoinjector 2.0 configuration application. This application is used to specify important parameters for the Autoinjector 2.0 system.

2.2.2 Main Autoinjector application

The main application is the GUI that allows the user to control the system. It displays a video feed from the microscope camera to visualize the microinjection process. The GUI also includes various widgets (buttons, dropdowns, and other interactable items) that allow the user to control the microinjection procedure. An image of the main Autoinjector application is shown in Figure 4.

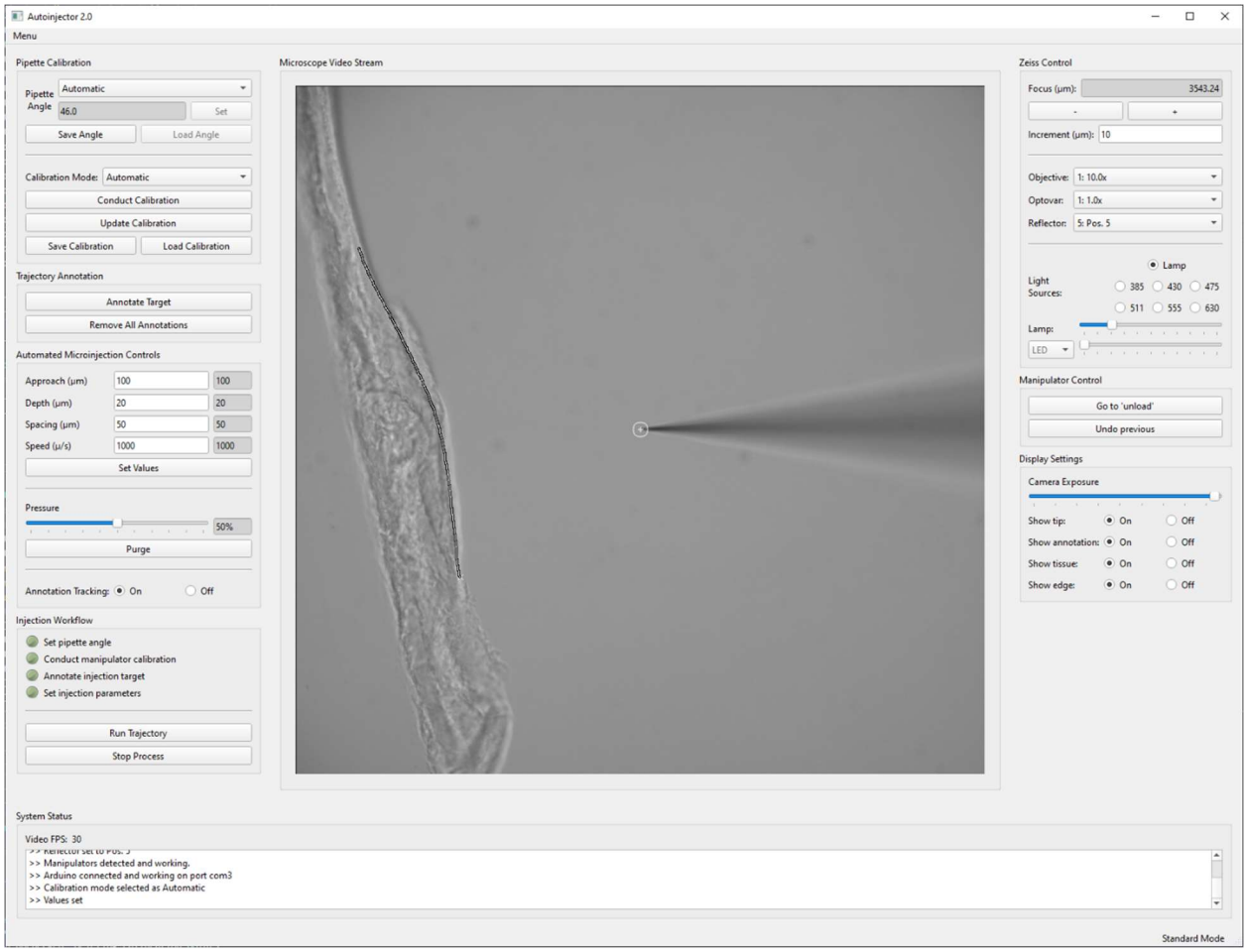


Figure 4. Main Autoinjector 2.0 application where the user controls the system. A pipette and a piece of agarose are shown in the video.

Widgets are grouped within the GUI by the functionality they provide. These widget groups are divided into: calibration, annotation, video display modifiers, data acquisition, video feed, microscope control, manipulator control, injection parameters, and injection workflow. A brief description of the purpose of these widgets is given below. A more detailed explanation of the main Autoinjector application and a description of every widget is available in **APPENDIX C**.

- **Calibration:** These widgets facilitate the micromanipulator-microscope calibration process which is required so the system knows where the microinjection pipette is located in 3D space. More information on calibration is available in section **3.1**.
- **Annotation:** These widgets allow the user to manually or mark the desired injection targets. This defines the 3D locations of the injection targets so the system knows where to go during injections. More information on annotation is available in section **3.2**.

- **Injection Parameters:** These widgets control the behavior of the microinjection procedure. For instance, the user can change the injection pressure or the speed of the manipulator while performing injections. More information is available in section 3.3.
- **Injection workflow:** These widgets allow the user to run the microinjection process. More information is available in section 3.4.
- **Video feed:** This display's the microscope camera's video, so users can watch the microinjection process. The user can also interact with the video feed for the calibration and annotation processes.
- **Microscope control:** These widgets permit the user to have computer-control of the microscope like changing the focal plane or the epifluorescent illumination.
- **Manipulator control:** These buttons provide useful shortcut functions like moving the manipulator to the "unload" position where the user can swap the microinjection pipette. This eliminates the need for the user to manually use the micromanipulator control wheels to move to the "unload" position.
- **Video display modifiers:** These widgets change auxiliary information that is displayed in the video feed. For instance, users can toggle whether to show the injection target annotation.
- **Data acquisition:** These control what images and data are saved during the injection process. These widgets are only available in developer mode as explained in Table 4.

2.3 Autoinjector 2.0 microinjection strategy

To conduct automated microinjections, Autoinjector 2.0 follows the same microinjection strategy as Autoinjector 1.0 as shown in Figure 5. First, the user manually brings the tissue section and the pipette tip into focus in the microscope's FOV. Then the user runs a microscope-manipulator calibration process to define the location of the pipette tip. This calibration process allows the Autoinjector system to "know" the location of the pipette tip and guide the tip to any location in the microscope's FOV for conducting microinjection. Next, the user conducts an annotation process to define the location of the microinjection targets along the tissue's edge. Finally, now that the system "knows" the location of the pipette tip and the location of the injection targets, the system can run the automatic microinjection procedure. During the automatic microinjection process, the system will guide the pipette to the annotation and sequentially inject along the annotation line. User defined parameters dictate how deep the pipette will penetrate the tissue, how much spacing to include between sequential injections, and the speed of the injection process.

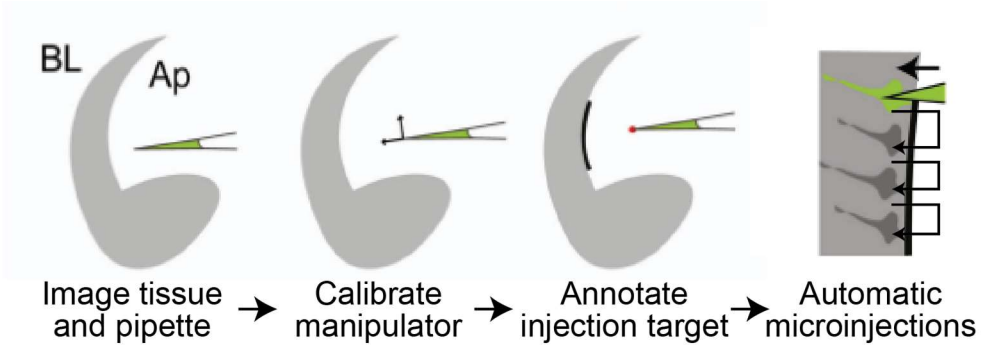


Figure 5. Generalized strategy for robotic, automated microinjections with Autoinjector 2.0. First the user images the tissue and the pipette in the microscope. Then the manipulator and pipette are calibrated to the microscope, so the system "knows" where the pipette tip is located in 3D space. Next the injection targets are annotated to define their 3D locations. Finally, the system can automatically guide the pipette to the targets to conduct microinjection.

2.4 Comparison of Autoinjector 2.0 to original Autoinjector

While the Autoinjector 2.0 follows the same general microinjection strategy as Autoinjector 1.0, Autoinjector 2.0 introduces a variety of improvements to enhance the automation of the original system. A comparison of the differences in automation between Autoinjector 1.0 and Autoinjector 2.0 is shown in Figure 6.

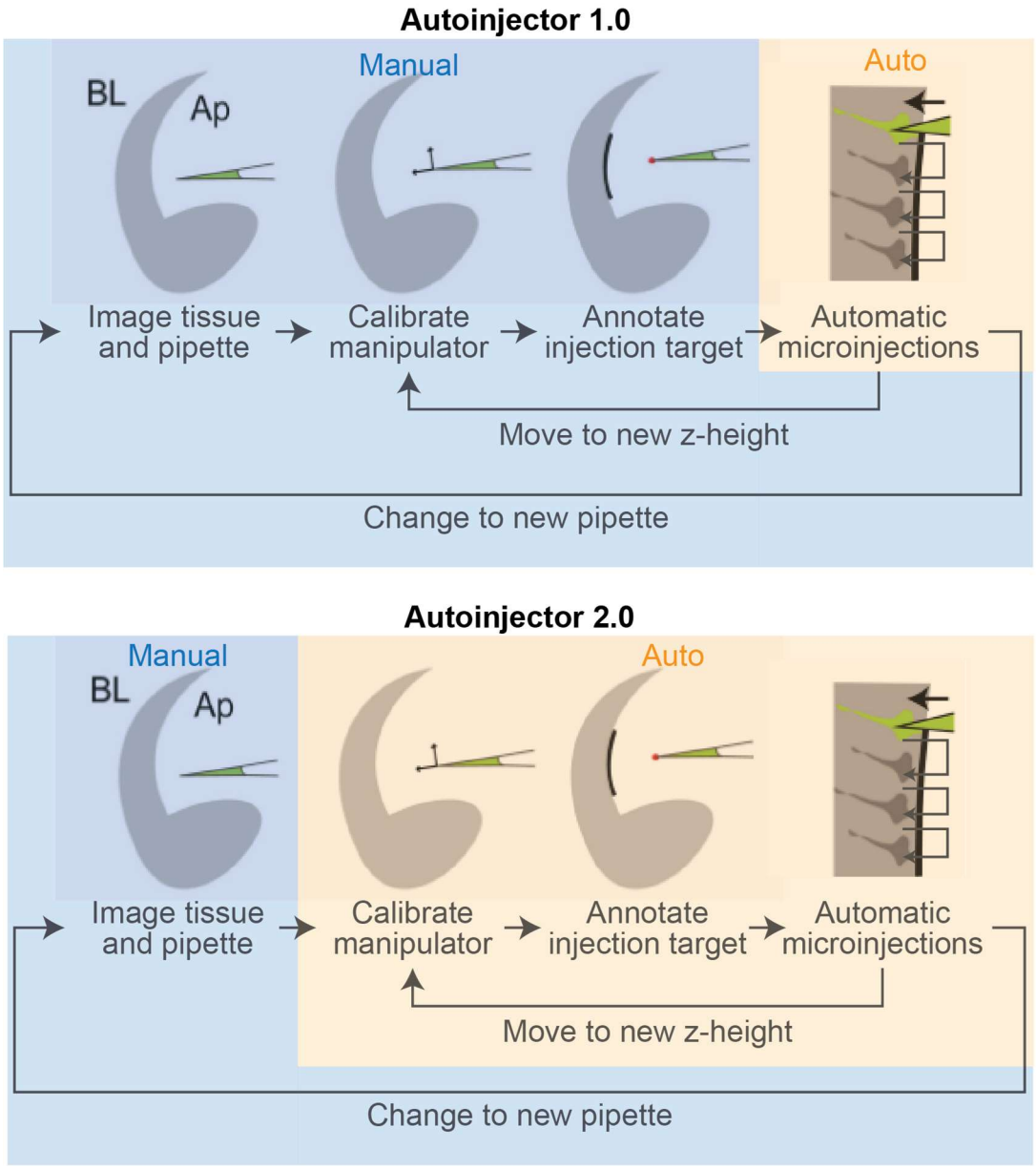


Figure 6. Comparison of automation between Autoinjector 1.0 and Autoinjector 2.0. While Autoinjector 1.0 and Autoinjector 2.0 use the same general microinjection strategy, Autoinjector 2.0 leverages greater levels of automation in the calibration and annotation processes to increase the speed and consistency of injections.

Autoinjector 2.0 improves upon the original Autoinjector system by including several hardware and software updates that enabled greater levels of automation. The main improvements of Autoinjector 2.0 include:

- Upgrading the fully manual microscope to a Zeiss Axio Observer 7 inverted microscope with automated hardware features including computer controllable objective changer, optovar changer, reflector changer, focus controller, and stage. In addition, the

transmitted light source and epifluorescence LED light source are computer controllable. Having a computer controllable microscope enabled some of the software improvements detailed below like automatic 3D target annotation by using a microscope z-stack.

- Modifying the pressure control system to include a computer controllable option for high-pressure unclogging of the pipette. Sending a high-pressure pulse to the pipette can help clear obstructions that would cause a user to change pipettes. Therefore, this reduces the amount of time spent changing pipettes and recalibrating the system.
- Expanding the microscope-manipulator calibration process to account for the microscope and manipulator's z-axes thereby enabling 3D calibration. This enables 3D microinjection targetting without needing to update the calibration when changing microscope focus heights as was required for Autoinjector 1.0.
- Adding a neural network for pipette tip detection to enable automated microscope-manipulator calibrations. These automated calibrations are quicker (and potentially more consistent) than the manual user calibrations.
- Adding a neural network and computer vision system for detection and classification of tissue/tissue edges for automated target annotation. Additionally, this process can be done automatically in 3D using the computer controllable microscope for conducting a z-stack. This automated annotation process is much faster than the manual user annotation process.
- Adding computer vision annotation tracking for accurate targetting during tissue displacement caused by the microinjection process. This enables more consistent and accurate injections when tissue displacement occurs.

3 Principles of robotic microinjection with Autoinjector 2.0

This section dives more in-depth to the injection strategy discussed in section **2.3**. It elaborates on the "what" and the "why" of the various steps in the robotic microinjection procedure so that you can understand the purpose for these steps. Moreover, this section also explains why and when you would select certain options/widgets in the GUI during the microinjection process.

3.1 Microscope-manipulator calibration

The goal of calibration is to make a "map" so the system knows how to move the micromanipulator to any location in the microscope's FOV. This means that the given coordinates of an injection position as viewed in the microscope, the system can move the manipulator to inject that position.

The microscope and micromanipulator have distinct, non-coincident coordinate systems, so the calibration process makes a map that converts between these coordinate systems. In general, the idea of calibration is to determine how to transform (rotate/flip/scale and then translate) the manipulator coordinate system so that it matches the microscope coordinate system. These transformations (rotate/flip/scale/translate) define the "map" between microscope and the manipulator. A simplified depiction of how calibration makes a "map" to convert between the coordinate systems is shown in Figure 7. Once this "map" is made, the system can use this "map" to determine where the manipulator is located relative to the microscope (and vice versa).

While it is not necessary to understand the details of how the calibration works, it is important to understand that the quality of the calibration affects the injection positioning accuracy. To continue the map analogy, consider two scenarios where you are trying to tell your friend how to navigate from their house/apartment to your house/apartment. In the first scenario, they are using a hand drawn map that you drew in one minute. In the second scenario, they are using Google/Apple maps on their phone. It will be much easier for your friend to navigate to your house/apartment when they are using Google/Apple maps because the map is more accurate. **Likewise, a quality calibration will mean the micropipette will more accurately reach the injection positions because the "map" is more accurate. Therefore, it is important that the user takes care during the calibration process to generate an accurate, high-quality calibration.**

If you are interested in the mathematical details of calibration, you can reference **APPENDIX D**.

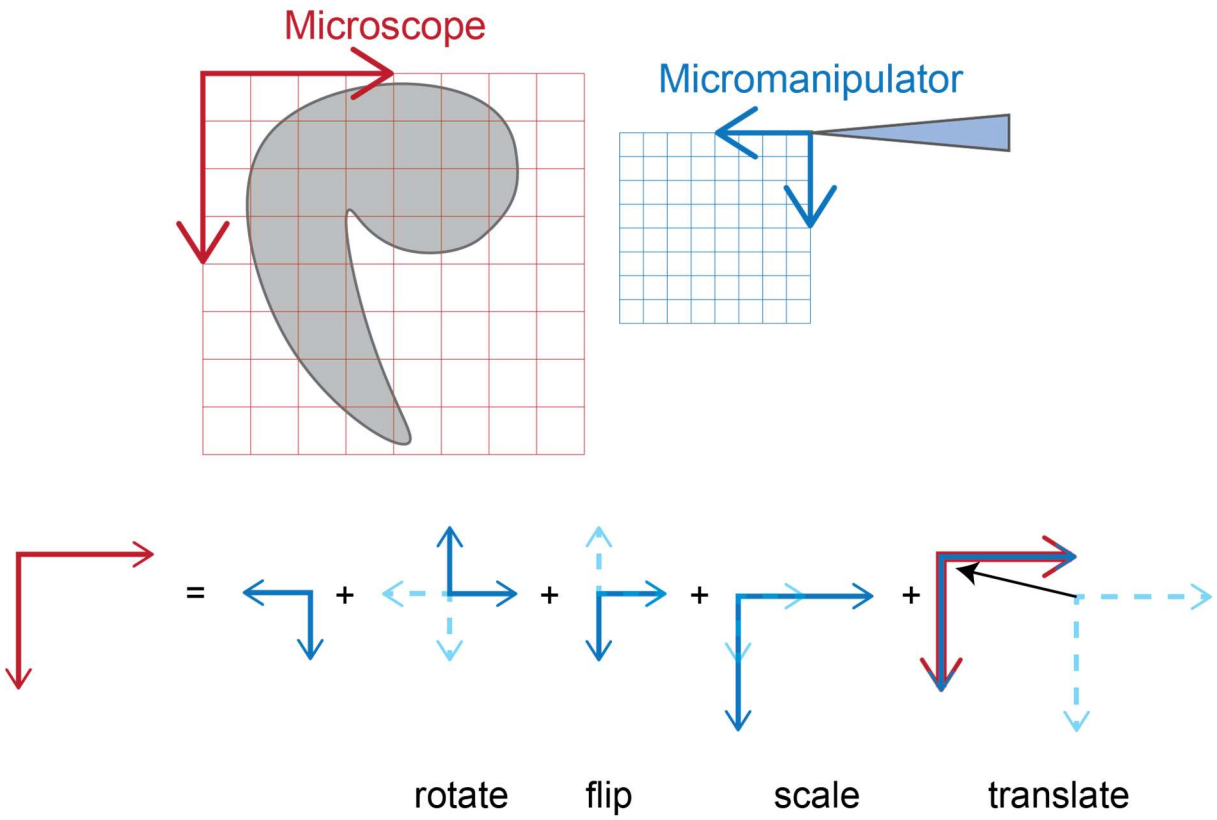


Figure 7. Calibration makes a "map" to convert between the micromanipulator and microscope. In this example, the "map" consists of rotating, flipping, scaling, and translating the micromanipulator coordinate system to coincide with the microscope coordinate system.

3.1.1 Calibration process overview

The calibration process consists of noting the coordinates of the **in-focus** pipette tip in the microscope coordinate system and in the manipulator coordinate system for various pipette tip positions within the microscope FOV. Using math, the system calculates the “map” (rotation, flip, scaling, and translation) between the lists of paired coordinates. If you are interested, mathematical details of the calibration process are described in **APPENDIX D**.

To implement this calibration process in the user interface, the user must click on the **in-focus** pipette tip (or the tip must be automatically detected) to note the coordinates. Clicking on the tip saves the pixel coordinates of the clicked position and saves the manipulator coordinates when the click occurred. A schematic of the process is shown in Figure 8. **Each time the user clicks on the tip, they should click on the same position on the pipette tip to ensure a high-quality and accurate calibration!** This process must be repeated for at least three non-colinear points in the microscope FOV but having more than three points improves calibration accuracy. The “Semi-Auto.” and “Automatic” calibration processes are programmed to generate

five different points. When the user completes the calibration, then the map is computed and used for the microinjection process.

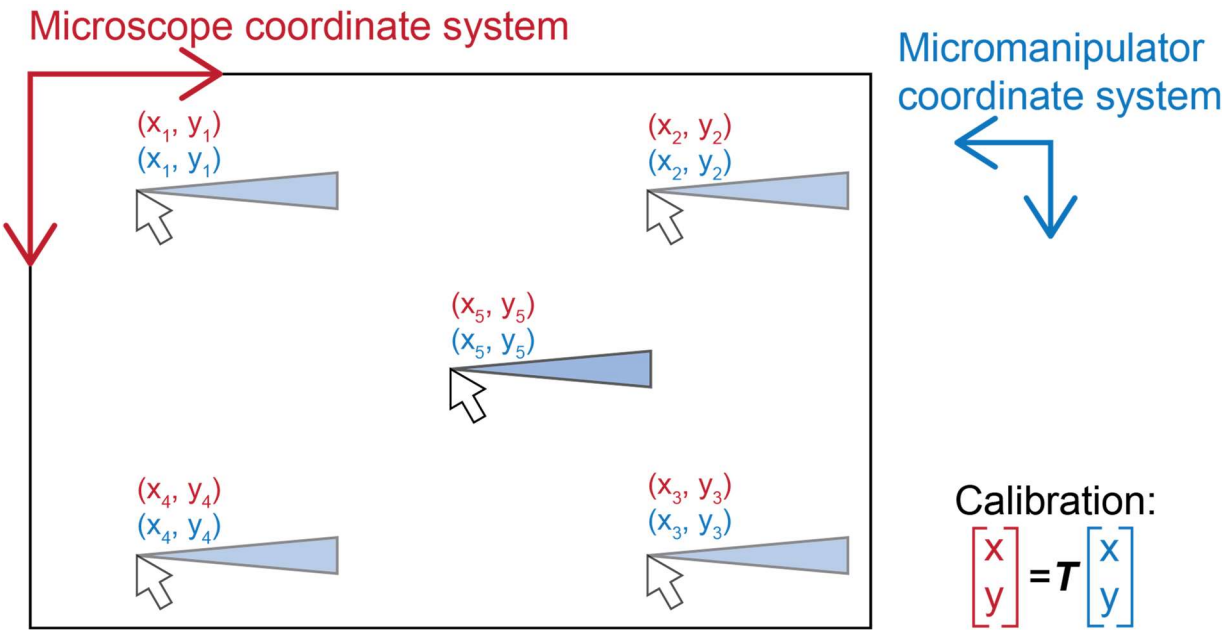


Figure 8. Schematic of the microscope-micromanipulator calibration process. The user clicks on the pipette tip (or a computer vision algorithm detects the tip) in the video display to generate a list of paired microscope and micromanipulator coordinates. Then a calibration can be computed that constructs the map (designated as "T" or "transformation") between these coordinates.

3.1.2 Calibration modes (Automatic, Semi-Auto., and Manual)

Autoinjector 2.0 has three different modes for conducting the calibration process (Automatic, Semi-Auto. and Manual). These modes dictate how the system generates the list of microscope and manipulator coordinates for different positions of the pipette tip. The differences between these modes stems from the varying levels of automation in the calibration sub-processes as summarized in Table 1. It is safe to mostly use "Automatic" mode and switch to "Semi-Auto" mode if the system is struggling to accurately detect the pipette tip location. It is not recommended to use "Manual" mode.

1. **Automatic:** The system will automatically move the pipette tip to different positions in the microscope FOV and it will automatically detect the pipette tip location. In addition, the system will autofocus the pipette tip before starting the calibration process. The system detects the pipette tip location by using a YOLOv5 neural network trained on images of the pipette tip. Details on how the pipette tip detection neural network are available in **APPENDIX E**.

2. **Semi-Auto:** The system will automatically move the pipette tip to different positions in the FOV, but the user must manually click on the pipette tip to register its location. Moreover, the user must manually focus the pipette tip before starting the process.
3. **Manual**: All processes are manual. The user must manually move the pipette tip, manually click on the tip to register its location, and manually focus the pipette tip before starting calibration.

Table 1. Comparison of calibration modes by evaluating the automation level of the calibration sub-processes. For Automatic mode, all sub-processes are automated. In Semi-Auto mode, only the movement of the pipette between positions is automated. In Manual mode, everything must be completed manually.

	Focus pipette	***Register position of pipette tip***	***Move pipette to new position***
Automatic mode	Automatic	Automatic	Automatic
Semi-Auto. mode	Manual	Manual	Automatic
Manual mode	Manual	Manual	Manual

3.1.3 Qualitatively assessing calibration accuracy

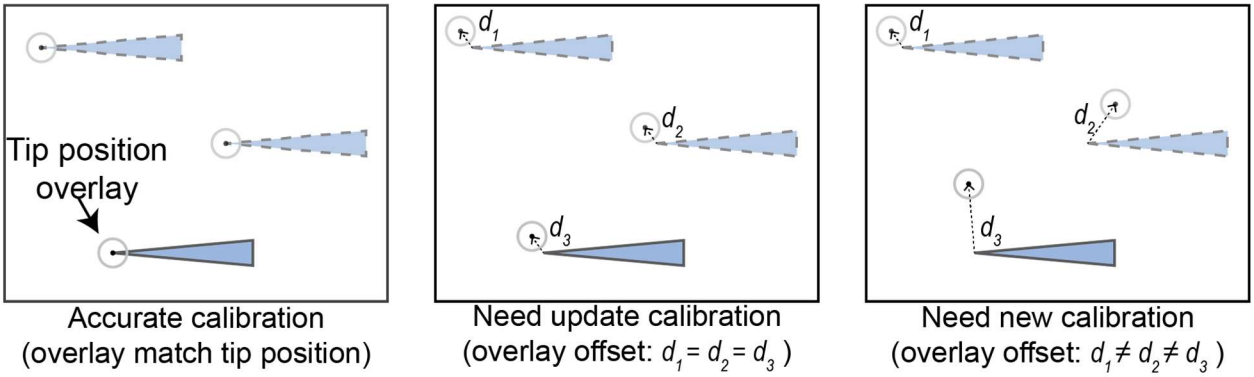
The Autoinjector 2.0 software has a feature that will show an overlay of the calibrated tip position in the video display (see “Show tip” in Table 12). This feature uses the manipulator position and the calibration map to compute the pixel coordinates of the pipette tip position, and it displays these pixel coordinates as a small circle overlay in the software’s video display. This shows where the Autoinjector 2.0 system thinks the tip is located according to the microscope-manipulator calibration. The position of the circle shows the (x, y) position of the calibrated tip position and the size/color shows the z-position relative to the current focus plane. Red means the system thinks the tip is below the current focus plane and blue means that the system thinks it is above the current focus plane.

You can qualitatively assess the accuracy of the calibration by observing this tip position overlay. After calibration, ensure the “Show tip” feature is on to display the tip position overlay. As you move the pipette tip throughout the field of view (by using the manipulator control wheels), you should observe that the tip position overlay tracks the pipette tip and coincides with the tip everywhere within the field of view. If it coincides with the true tip position, then the calibration is accurate. An illustration of an accurate calibration is shown in Figure 9. If the overlay doesn’t coincide with the true tip position, then you may need to update the calibration or conduct an entirely new calibration.

3.1.4 New versus update calibration

The Autoinjector has two ways to change the calibration: conduct a new calibration to create a brand new “map” (new rotation/flip/scale/translation) or update an existing calibration “map” (only a new translation). Updating the calibration is nice because it is quicker than a brand-new calibration. Moreover, when changing pipettes, usually loading the previous calibration and updating the calibration is sufficient for an accurate calibration. However, sometimes updating a calibration will not fix calibration inaccuracies.

You can determine whether you need to update the existing calibration or conduct a brand-new calibration by observing the tip position overlay. If the tip position overlay doesn’t coincide with the true tip position but it is offset from the true tip by the same constant distance everywhere in the FOV, then an update calibration will fix the problem. If the tip position overlay is offset from the true tip position and the displacement between the two changes throughout the FOV, then you need to conduct a brand-new calibration. An illustration of the update versus new calibration scenarios are shown in Figure 9.



Accurate calibration (overlay match tip position)

Need update calibration (overlay offset: $d_1 = d_2 = d_3$)

Need new calibration (overlay offset: $d_1 \neq d_2 \neq d_3$)

Figure 9. Demonstration of calibration accuracy and solutions for inaccurate calibrations. When the calibration is accurate, the tip position overlay (as computed by the calibration) will match the true tip position in the video display. When the calibration is inaccurate, the overlay will be offset from the true tip position. It can be corrected with a calibration update when the overlay is offset at a constant distance from the tip throughout the field of view. A brand-new calibration is needed when the overlay offset distance changes throughout the field of view.

3.1.5 Manipulator Z- and D-axes calibration and manipulator angle

Autoinjector 2.0 does not do a calibration process to relate the manipulator’s Z- and D-axes to the microscope coordinate system. This is because it assumes a constant rotation/flip/scaling between the manipulator Z- axis and the microscope Z- axis, and a known (and accurate) angle for the manipulator’s injection axis. **Therefore, if the pipette tip is in-focus during the calibration process, the Z- and D- axis positioning should be accurate for the Autoinjector 2.0 at Human Technopole without doing Z- or D- calibration procedures.**

These assumptions/simplification makes the calibration process easier for the user because they don't need to go through the steps for calibrating the Z- and D-axes but comes at the cost of flexibility. For instance, Z-axis positioning will be inaccurate if refractive index mismatch changes between the objective and the immersion media (aka if you change to a water/oil immersion objective or if you don't submerge the tissue in aqueous media). **You can determine if you have Z- or D-axes calibration errors if the pipette tip is in-focus during the calibration process, but it is out-of-focus when the tip is at the final injection position during the microinjection process.** If this is an issue, feel free to modify the z-scaling in the configuration, calibration functions, and/or contact Jacob if you need more information or help.

The Autoinjector software has the option of specifying the manipulator injection axis angle. This is the angle that is mentioned above that is used in the calibration functions. The manipulator has its own method for measuring the axis angle which is what is used in "Automatic" angle mode. This "Automatic" mode should always be sufficient unless you have reason to believe the angle is inaccurate. If you believe the "Automatic" angle is inaccurate, then you can manually specify the angle you believe to be correct.

3.2 Injection trajectory annotation

The annotation process defines the injection locations. This process is necessary for the Autoinjector 2.0 system to know where to guide the pipette tip for microinjections. In Autoinjector 1.0, the user could only manually annotate a single injection trajectory at a time. In contrast, Autoinjector 2.0 enables the user to annotate multiple injection trajectory for back-to-back injections, and it also provides the ability to automatically annotate the tissue edges using a neural network.

3.2.1 Manual annotation mode

In manual annotation mode, the user can manually draw injection trajectories in the user interface. After opening the annotation window, the user can left-click-and-draw along the tissue edge to manually annotate the injection trajectory (just like in Autoinjector 1.0). A schematic of the manual annotation is shown in Figure 10. The user can continue to annotate additional injection trajectories by drawing more annotations along desired injection trajectories. Moreover, the user can even change focus heights between annotations to annotate trajectories at different z-heights; the Autoinjector 2.0 will use the 3D microscope-manipulator calibration to accurately target these 3D annotations during the microinjection process. Annotations can be removed in the annotation window by middle-mouse-button clicking near the annotation that you want to remove.

3.2.2 *Automatic annotation mode*

In addition to manual annotation mode, the user can use a neural network to automatically detect and annotate tissue edges for microinjection. Autoinjector 2.0 permits two ways to do these automatic annotations: the user can run a single automatic annotation on the current field of view, or the user can set up a z-stack to automatically annotate each plane in the z-stack. A comparison between the automatic annotations is shown in Figure 10. More information on the details of the neural network and automatic annotations are available in **APPENDIX F**.

For the single automatic annotation, the user simply brings the desired tissue section into focus and then clicks “Run single auto-annotation”. The neural network will automatically detect and annotate the desired edges in the current image. Then they can change focus height and repeat conduct single automatic annotations at different focus heights to generate a set of annotations in 3D. Annotations can also be removed by middle-mouse-button click near the annotation. They can even augment automatic annotations by manually drawing additional annotations.

For z-stack automatic annotations, the user must specify the z-stack parameters before running the annotation process. This starts by specifying the starting and finishing heights of the z-stack by focusing on the start and finish height and clicking buttons to specify each. Finally, the user sets how many sections to include in the z-stack. With the z-stack limits specified and the number of sections set, the user can run the automatic z-stack annotation. It will automatically focus on each section in the z-stack (separated by uniform spacing), and the neural network will automatically detect and annotate the desired edges in that section before moving to the next. Like before, the user can augment automatic annotations by manually drawing additional annotations, and the user can remove annotations with middle-mouse button click.

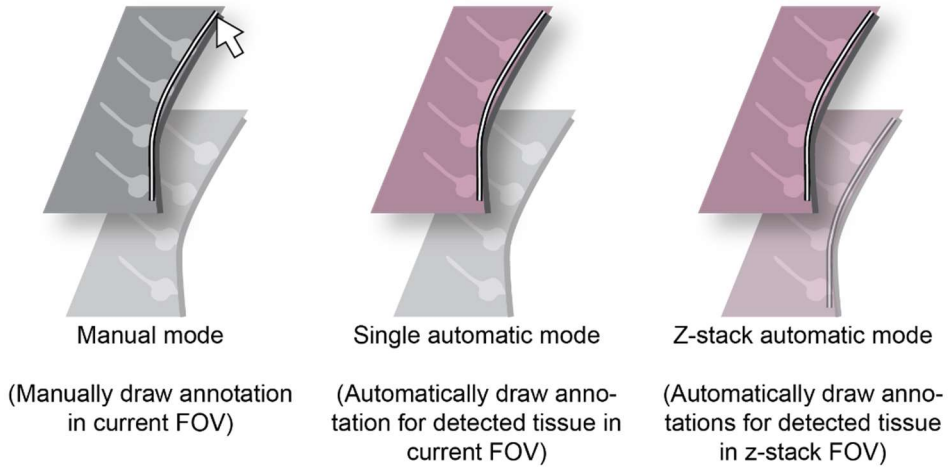


Figure 10. Comparison between annotation modes. In manual mode, the user draws the annotation. In automatic modes, a neural net

3.2.3 Reachable vs Basal vs Apical edge detection

When running automatic annotations, the user must specify which edges to annotate from a list of choices including “reachable”, “basal”, and “apical”. These choices will change which detected edges are automatically annotated. Moreover, the edges that are automatically annotated will change depending on whether the microscope-manipulator calibration is completed. A summary of the differences between these options follows below.

- **Reachable:** Non-biased edge annotation that will annotate any edges that are accessible by the pipette. If the calibration has not been completed, all detected edges will be annotated. However, if calibration is completed, then only edges that the pipette can touch without colliding with other items/tissue will be annotated. For example, if a piece of tissue is detected between the pipette and a detected edge, it will not annotate the edge because the pipette would collide with the tissue if it tried to inject that edge. Moreover, if the detected edge makes a very shallow angle with the pipette (like if the edge is nearly parallel) then the edge will not be annotated. A schematic of reachable versus unreachable edges is shown in Figure 11.
- **Apical:** Automatic annotation will only annotate “reachable” edges that are classified as apical (their curvature is mostly concave). If calibration is incomplete all apical edges are annotated.
- **Basal:** Automatic annotation will only annotate “reachable” edges that are classified as basal (their curvature is mostly convex). If calibration is incomplete all basal edges are

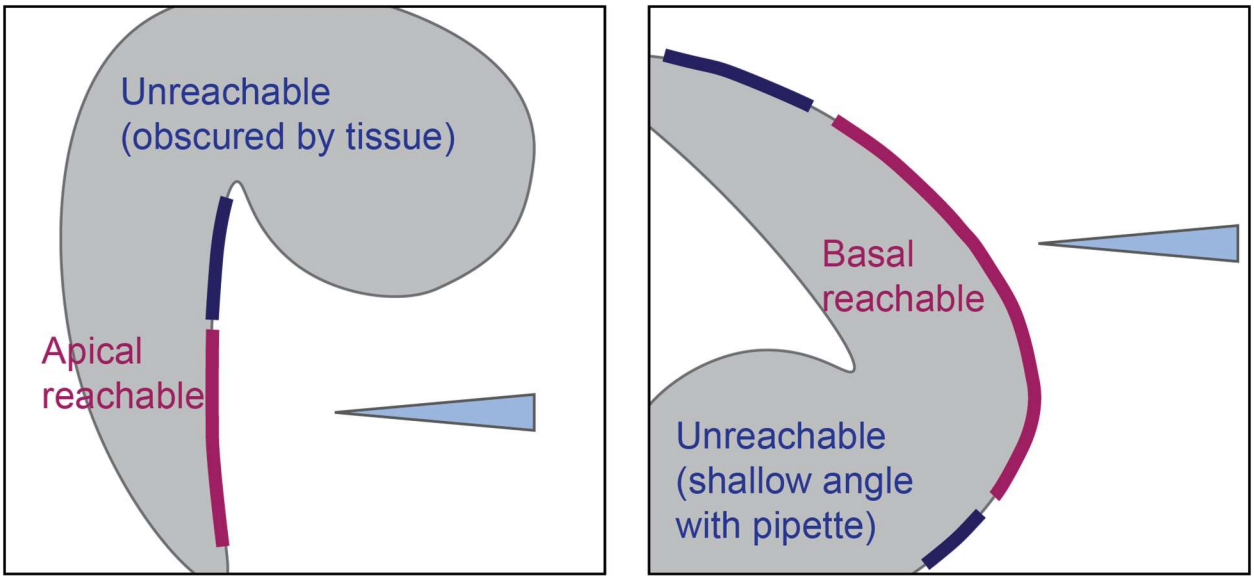


Figure 11. Example of reachable versus unreachable edges for automated annotations. Reachable edges can be accessed by the pipette without colliding with other tissue. Moreover, reachable edges are nearly perpendicular to the pipette.

3.2.4 Viewing the neural network detections

During the automatic annotation process, the user can see the results of the neural network detections. If the “Show tissue” and “Show edges” features are enabled (see Table 12), then the user interface will display the output of the neural network as an overlay on the video screen. The detected tissue will be shown as a semi-transparent maroon overlay, the detected apical edges will be shown as a semi-transparent blue overlay, and the detected basal edges will be shown as a semi-transparent yellow overlay. Some examples of the neural network detection overlays are shown in Figure 12.

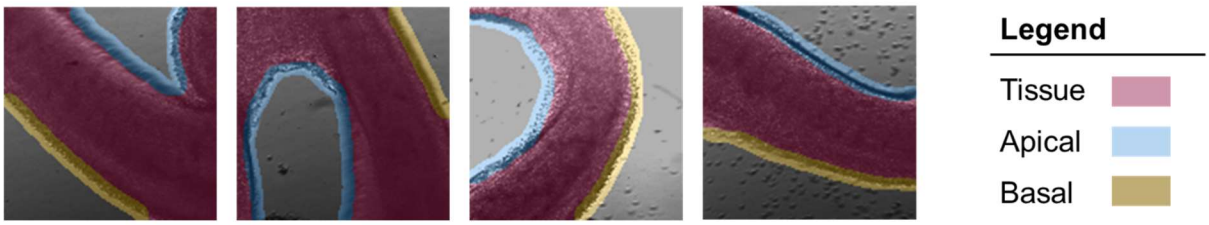


Figure 12. Overlays depicting neural network tissue/edge detections (mouse tissue). In the Autoinjector user interface, detected tissue will be maroon, detected apical edges will be blue, and detected basal edges will be yellow.

3.3 Automated injection parameters

Before the user can start doing automated microinjections, they need to specify how the system will conduct the microinjection process. This can be accomplished by setting various parameters in the Autoinjector software that dictate how the manipulator will conduct the injection trajectory,

the pressure at which to inject the solution, and whether to actively track the injection targets to compensate for tissue displacement.

3.3.1 Injection trajectory parameters

The injection trajectory parameters specify how the system will approach and inject along the annotated injection trajectory. The parameters that control this process are the “approach” distance, “depth” distance, “spacing” distance, and manipulator “speed”. These parameters are summarized below and shown in Figure 13. In the Autoinjector 2.0 software, there are entry boxes for entering these parameters. Next to the entry boxes are display boxes that indicate the currently set values. These display boxes will appear red/pink if the parameters are not set or if the values in the entry boxes do not match the currently set parameter values.

- **Approach:** This specifies the distance away from the annotation that the pipette will start its injection approach towards the annotation. This is the distance away from the annotation, not the tissue edge. Example: If your annotation is 20μm beyond the tissue edge, and the approach distance is 100μm, then the pipette will start approaching the tissue 100-20=80μm away from the tissue edge.
- **Depth:** This is the distance past the annotation that the pipette will penetrate the tissue. This is the distance past the annotation, not the tissue. Example: If your annotation is 20μm beyond the tissue edge, and the depth is 30μm, then the pipette will penetrate 30+20=50μm into the tissue.
- **Spacing:** This is the amount of space between injection locations along the annotation. The pipette will always inject at the beginning of annotation, but all subsequent injections will be separated by this spacing.
- **Speed:** This is the speed of the manipulator. All manipulator movements including injection, retraction, and moving between approach positions will use the same speed.

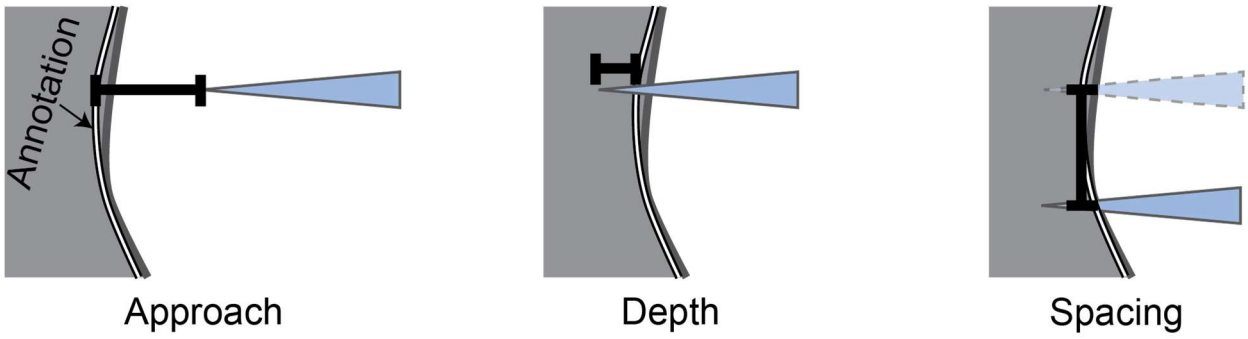


Figure 13. Depiction of different injection trajectory parameters. Approach specifies how far from the annotation that the pipette will start approaching for injections. Depth specifies the distance past the annotation that the pipette will penetrate. Spacing specifies the distance between injection locations along the annotation.

3.3.2 Pressure

The pressure serves two purposes for the Autoinjector 2.0. First, it provides a compensation pressure that constantly expels injection solution out of the pipette tip to prevent clogging. This prevents tissue and other debris from infiltrating the tip and causing the pipette to clog. Moreover, the compensation pressure reduces the time that the injection solution spends inside the pipette and mitigates aggregates that may clog the tip. **Because of these reasons, the pressure should be applied before the pipette is submerged into the tissue media to reduce the risk of clogging.** Secondly, the pressure delivers the injection solution inside the cell once the cell is penetrated by the pipette tip.

The Autoinjector 2.0 uses the same pressure value for compensation pressure and injection pressure, and this pressure value is specified by the pressure slider. The true percentage value corresponds to approximately the slider's percentage value of the electronic pressure controller's full pressure range. For example, at Human Technopole, the pressure controller has a range of 0-2psi, so 50% on the slider corresponds to approximately 1psi.

In addition to the pressure slider, there is a "Purge" button that can facilitate unclogging a clogged pipette. This sends a half second high-pressure pulse of 30 psi (or whatever the air supply pressure is) to the pipette. This high-pressure pulse can potentially dislodge tissue stuck to the tip or any aggregates that may be clogging the pipette. Usually this is only a temporary solution, but it can potentially extend the "life" of the pipette for a couple more injection trials.

See **APPENDIX H** for more information on the pressure controller details.

3.3.3 Annotation tracking

During microinjections, the tissue may move when it is penetrated/poked by the pipette. Sometimes this isn't an issue because the tissue springs back to its original position, but other

times the tissue is permanently moved and doesn't spring back. This permanent displacement of the tissue can reduce injection accuracy because the system is programed to inject where the annotation is located; once the tissue is misaligned with the annotation, then the microinjections will miss their target. The user can compensate for tissue displacement by modifying several factors like the size/weight of the tissue, the density of tissue being injected (for instance, the basal surface of organoids seems denser than the apical surface), and the depth that the pipette penetrates the tissue. In addition, the user can also use the Autoinjector's annotation tracking feature to compensate for tissue displacement.

This annotation tracking feature can compensate for tissue displacement by using a computer vision algorithm to track points located on or near the drawn annotation. Therefore, when the tissue moves, the annotation tracker will attempt to update the annotation to realign with its originally drawn position relative to the tissue. With the annotation properly aligned with the tissue, the pipette can successfully penetrate the displaced tissue. This annotation tracking feature works on both manual and automatic annotations. A schematic of annotation tracking is shown in Figure 14. More details on the annotation tracker are available in APPENDIX G.

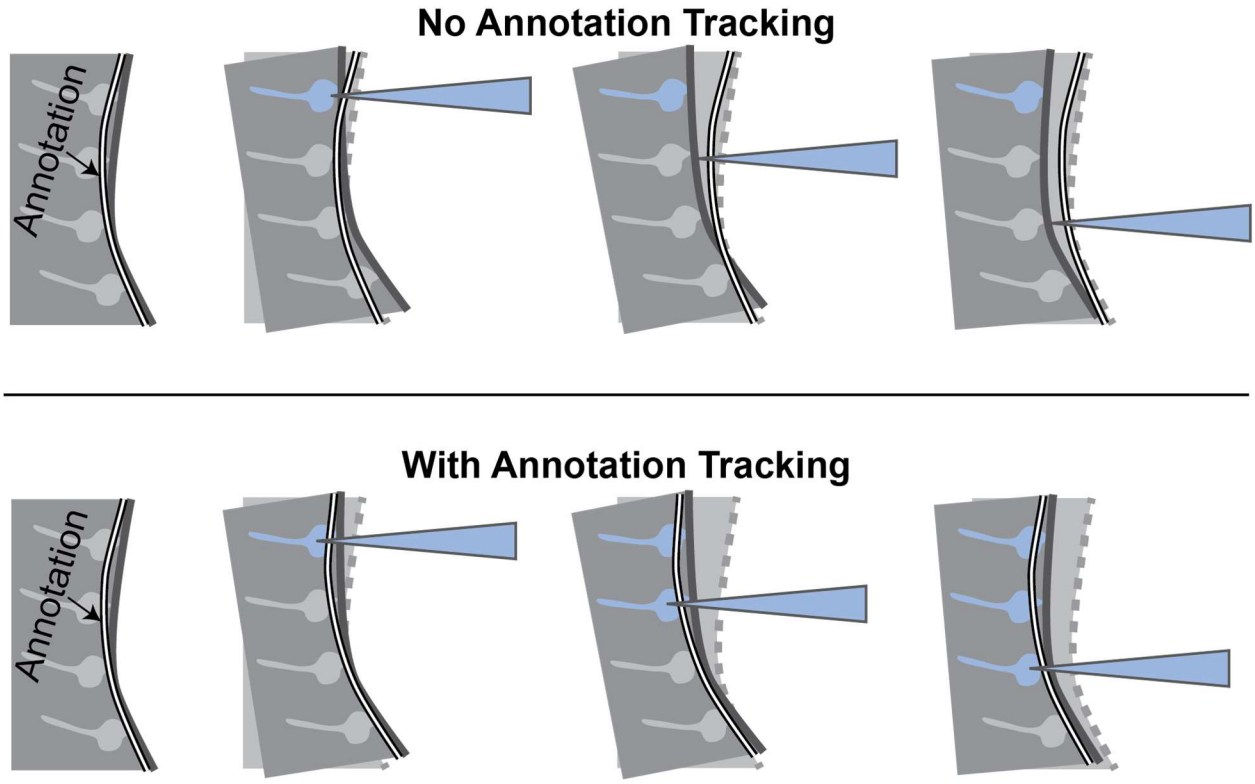


Figure 14. Demonstration of annotation tracking for tracking tissue displacement. Without annotation tracking, the tissue is displaced by the pipette, but the injection trajectory annotation remains fixed at its original location and the

pipette injects along this inaccurate annotation. With annotation tracking, the injection trajectory annotation is updated to track the displaced tissue which enables the pipette to inject into the tissue.

It is important to note that the system will track only the annotation that is currently being injected. This is because the user can annotate trajectories at multiple z-heights, so the other annotations may be out-of-focus thereby making those annotation's tracking points un-trackable. Likewise, the annotations are only tracked during the microinjection procedure because the system always focuses on the currently injected annotation; however, before microinjection is started, the user may be doing something where all annotations are out-of-focus making the annotations un-trackable. For these reasons, the annotation tracker waits to generate tracking points for the annotation until it is about to start injecting. **Therefore, it is important that the annotation is aligned with the tissue before starting injections.**

Despite only tracking a single annotation at a time, the system updates all annotations according to the displacement of the currently tracked annotation. Thus, the annotation tracking should work properly even when attempting to track multiple annotations. However, if the annotation that is being tracked is short (such that there are not many tracking points), then the tracking accuracy may decline.

3.4 Conducting automated microinjections

After completing calibration, annotation, and setting the injection parameters, the system is ready to start injections: it "knows" the location of the pipette tip, the location of the injection targets, and how to run the injection trajectory. All that remains is to tell the system to start injecting.

3.4.1 Running the injection process

The Autoinjector 2.0 system has several "status light indicators" to indicate the completion of the important processes that are necessary for conducting microinjections (calibration, annotation, and setting parameters). These "status light indicators" will be red for a specific process if that process has not been completed or something changed to invalidate that process. The indicators are green when all processes are completed, and the system is ready for injections.

Once all indicators are green, the user can begin the automated injection procedure by clicking "Run Trajectory". At any time, the user can stop the injection trajectory by pressing "Stop Process". When "Stop Process" is clicked, the system will finish its current injection and then retract out of the tissue.

During microinjections, the Autoinjector 2.0 GUI changes to a restricted state where some widgets are disabled/unusable. This is to prevent the user from clicking a button or starting an action that would interfere with the injection process. For instance, during injections the user will not be able to change the microscope focus height using the software because this could affect annotation tracking (if it is enabled). **The user should not manually adjust the microscope features that are disabled in the user interface. E.g., if the focus height is disabled, don't manually turn the microscope focus knob.** The user interface will revert to its fully usable state when injections are complete.

3.4.2 Automated microinjection trajectory

During automated microinjections, the system will sequentially inject along all the annotations. It will start by focusing on the first annotation, and then inject along that trajectory as defined by the injection trajectory parameters (section **3.3.1**). After it has completed injections on the first annotation, the system will focus to the next annotation and direct the pipette to start injections at the endpoint of the next annotation that is closest to the last injection position of the last annotation. It will then inject along this annotation until it has completed injections. This process repeats for all annotations, and it is demonstrated in Figure 15. Once the system has finished injecting all annotations, the pipette will be retracted from the tissue and the process will stop.

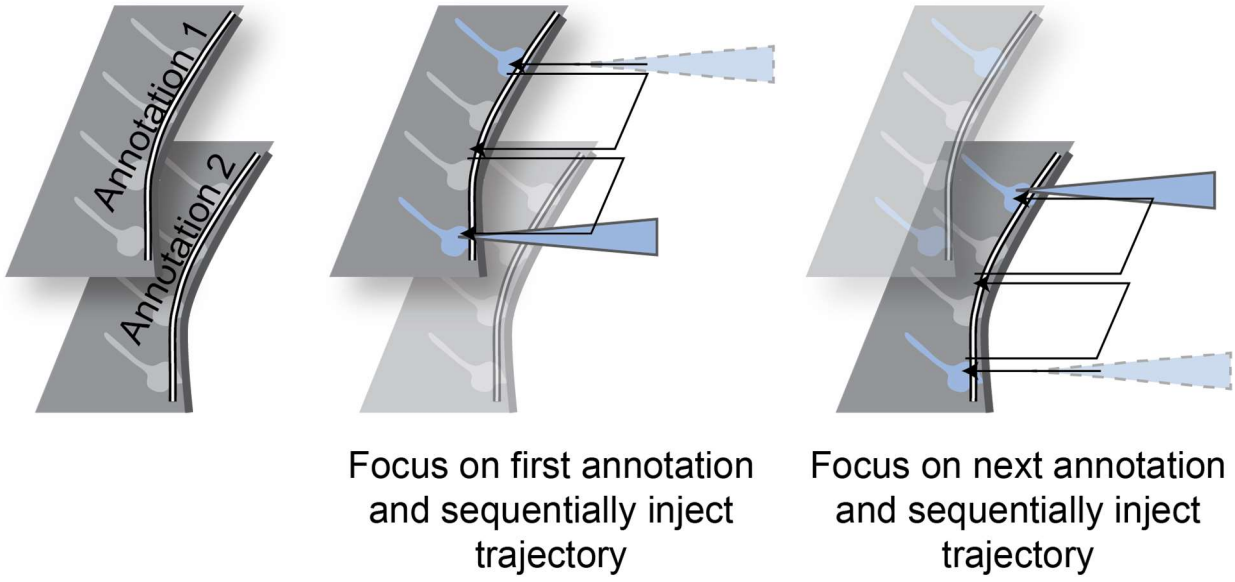


Figure 15. Demonstration of stereotypical automated microinjection trajectory. The system starts by serially injecting along the entire first annotation. Then it will focus on the next annotation and serially inject along the entire annotation. This process repeats for all annotations.

3.4.3 Handling pipettes: unclogging and changing pipettes

The pipette will likely get clogged after 50-200 microinjections depending on the pipette geometry and injection solutions. As mentioned in section **3.3.2**, the Autoinjector 2.0 has a "Purge" button that will send a high-pressure pulse to the pipette to unclog the pipette. However, eventually even the "Purge" button will not clear the clog and you will need to change the pipette to continue injections.

Autoinjector 2.0 has a feature that will use the manipulator to automatically retract the pipette from the tissue media and make it easier to change pipettes as shown in Figure 16. The "Go to 'unload'" button (Table 11) will automatically raise the manipulator's Z-axis and retract the manipulator's D-axis to provide space between the pipette and the microscope. Once the manipulator is at the "unload position" you can slide the manipulator away from the microscope and change the pipette. After you have inserted a new pipette and slid the manipulator back to its original position, you can click "Undo previous" (Table 11) to move the manipulator axes back to their original locations when "Go to 'unload'" was pressed. (There will actually be a slight offset in the original z-axis location to prevent collision with the petri dish when the new pipette is longer than the previous pipette). These features increase the speed of changing pipettes because the user doesn't need to manually move the manipulator out of the tissue media and move it back.

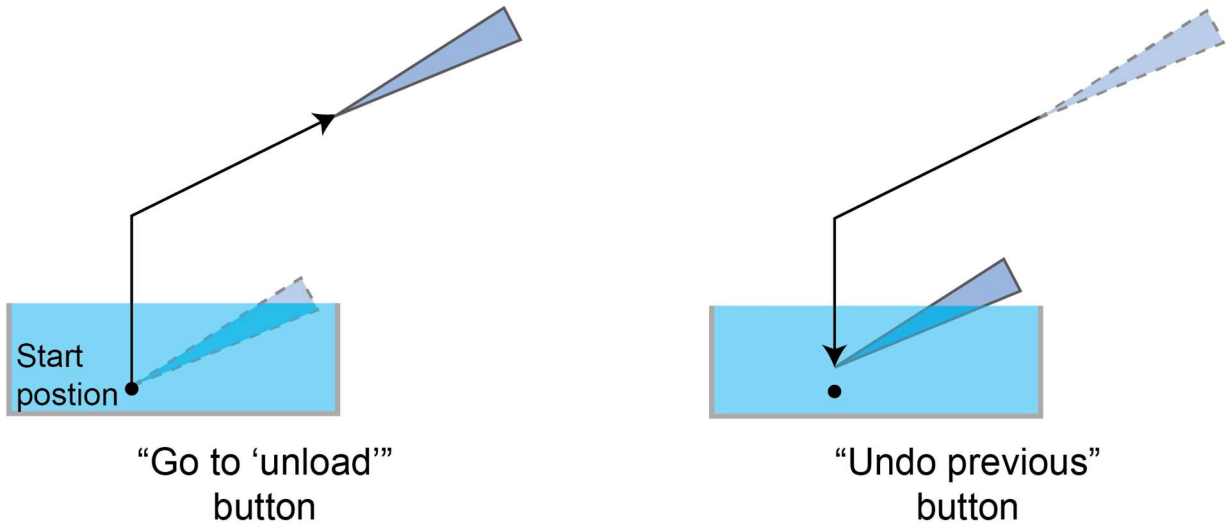


Figure 16. Demonstration of Autoinjector 2.0 features for changing pipettes. "Go to 'unload'" will raise and retract the manipulator's z- and d-axes to remove the pipette from the tissue media for changing pipettes. "Undo previous" will move the manipulator back to its original position with a small offset in the z-axis to prevent the pipette from colliding with the petri dish (which could happen if the new pipettes is longer than the previous).

4 Autoinjector 2.0 Operational Procedure

Using Autoinjector 2.0 consists of: pre-injection preparations, starting-up hardware (computer, microscope, manipulator, and pressure controller), using Autoinjector 2.0 software, and shutdown. These steps are verbosely detailed in the following subsections. The final subsection provides a condensed list of all the necessary operational steps (and a compilation of graphics) that can be used as a quick reference once the user is familiar with the system.

4.1 Install Autoinjector 2.0 and Configure Software

Before completing any of the following steps, the software must be installed from GitHub. The installation tutorial is available on the GitHub project page. You must also run a first time configuration (**2.2.1** and **APPENDIX B**).

4.2 Pre-injection preparations

The goal of pre-injection preparations is to have "everything" prepared, so the user will not be distracted by miscellaneous tasks once injections are started. Consequently, pre-injection preparations will vary with each experiment, but it generally consists of making injection solution, pulling glass capillary pipettes, and preparing tissue.

1. Make the injection solution[1].
2. Pull glass capillary pipettes on the pipette puller. Some reference programs are available in **APPENDIX A**.
3. Gather micropipette and micropipette tips for filling the glass capillary pipettes.
4. Gather necessary tissue solutions/media and chambers for storing tissue.
5. Prepare tissue for injection.

4.3 Startup hardware

To use Autoinjector 2.0, its four main hardware components (computer, microscope, manipulator, and pressure controller) must be turned on. The computer should be turned on first but turning on the remaining hardware systems can be completed in any order. However, it is particularly important to follow the correct procedure for turning on the microscope hardware. Failing to do so will make it impossible for the Autoinjector 2.0 to interface with the camera and/or microscope.

[1] To reduce clogging, microinjection solution is typically centrifuged at 4C for 30min at 16,000g, and the supernatant is separated.

4.3.1 Turn on the computer

1. Start the computer and login to your account.
2. If you're stuck here, call 113.

4.3.2 Turn on the microscope

It is particularly important to turn on the camera last (after the microscope and its software are started)! Microscope start up instructions are shown in Figure 17.

1. Turn on the motorized stage power supply: SMC 2009.
2. Turn on the focus controller power supply: Focus Controller 2. If the front panel button does not turn the device on, there is a power switch located on the back panel.
3. Turn on the microscope power supply: Power Supply 232.
4. Turn on the microscope power button.
5. Start the Zeiss ZEN pro software and wait for it to finish its startup configuration (up to 2 minutes). Ensure the stage and objective will not collide with anything during the initial calibration[2] and click "Calibrate Now" in ZEN to begin the microscope calibration.

> ⚠ *During microscope calibration, the microscope stage is at risk of colliding with the manipulator or other nearby components. Ensure it will not collide with anything!*

6. Turn on the camera. (If the camera was turned on before starting ZEN, you must turn the camera off, close the Zeiss ZEN software, and then restart at step 4).
7. Set the light path to the camera's port. If you don't do this, the video display in the Autoinjector 2.0 software will not show anything.

[2] During the microscope calibration, the objective "lowers" as far as it goes, and the stage moves "back and left" as far as it goes. Then the stage returns to its starting position and the objective returns to its starting position.

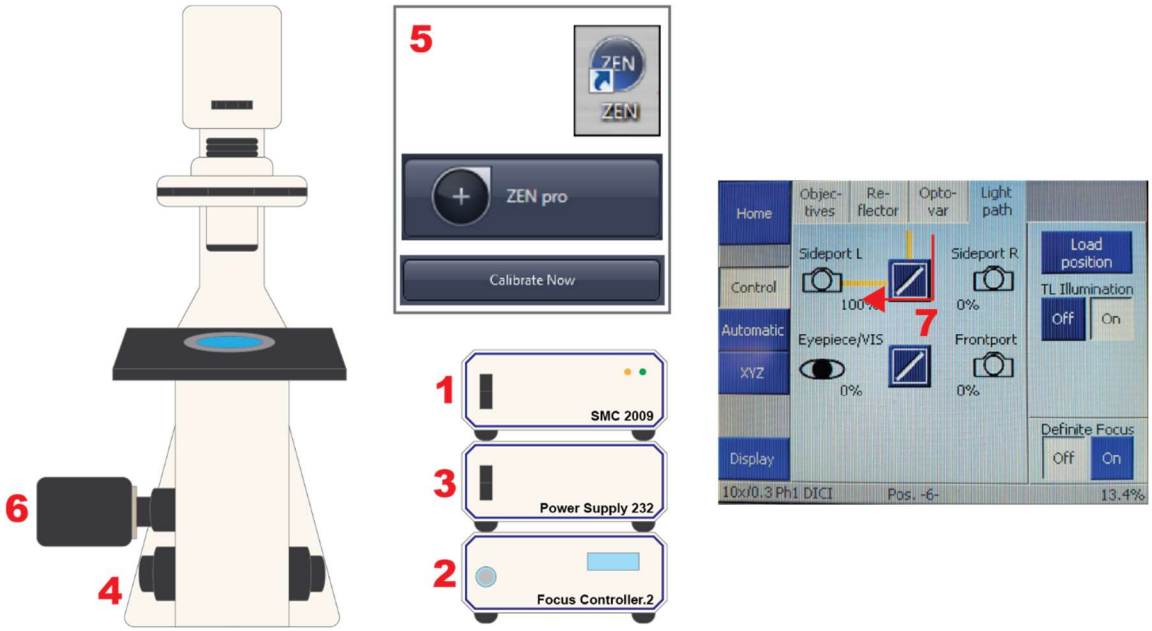


Figure 17. Microscope startup procedure. 1. Turn on stage power supply. 2. Turn on focus controller power supply. 3. Turn on microscope power supply. 4. Turn on microscope. 5. Open Zeiss ZEN Pro software and calibrate. 6. Turn on camera. 7. Set light path to camera.

4.3.3 Turning on manipulator

Manipulator startup instructions are shown in Figure 18.

1. Turn on the Sensapex manipulator by press-and-hold the middle button.
2. Open the manipulator information page by clicking the manipulator icon.
3. Verify the manipulator is connected and working ("1" displayed in orange circle[3]).
4. Perform a micromanipulator "load calibration"[4] (may take up to 10 minutes). This can be performed hours before injection. For instance, this can be performed in the morning when you arrive before tissue preparation.

⚠ *During manipulator "load calibration", the manipulator is at risk of colliding with the microscope or nearby objects. Ensure it will not collide with anything!*

[3] This "1" corresponds to the manipulator device number in the configuration app (Table 3).
[4] During a "load calibration" the manipulator will initially move all axes to a value of 10,000. Then it will sequentially move each individual axis through its whole range of motion (0 to 20,000) while holding all other axes at 10,000.

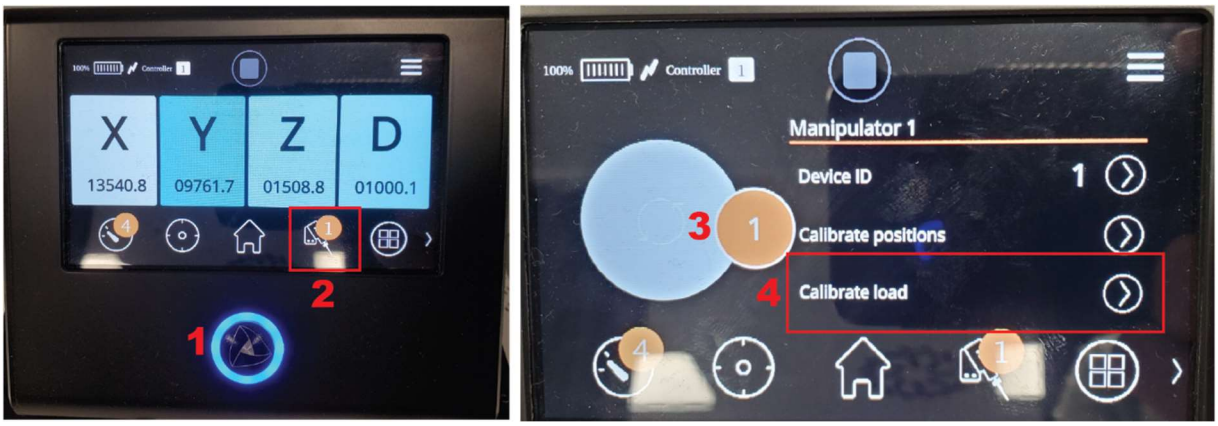


Figure 18. Manipulator startup procedure. 1. Turn on manipulator. 2. Open manipulator info page. 3. Ensure device is connected (circle is orange with "1"). 4. Calibrate manipulator.

4.3.4 Turn on pressure controller

Pressure controller startup instructions are shown in Figure 19.

1. Connect the power supply cord.
2. Flip the pressure controller power switch to on (which is when "|" is depressed).
3. Connect the USB cord.
4. Turn on the air supply.
5. Adjust the air supply regulator until the air supply gauge on the wall reads 20-30psi.
6. Adjust "DOWN REGULATOR" until the "INTERMEDIATE PRESSURE" reads 2.5-3.5psi.

> ⚠ *You are at risk of damaging the pressure controller components if the gauge on the wall exceeds 35psi or the "INTERMEDIATE PRESSURE" gauge exceeds 3.5-4psi!*

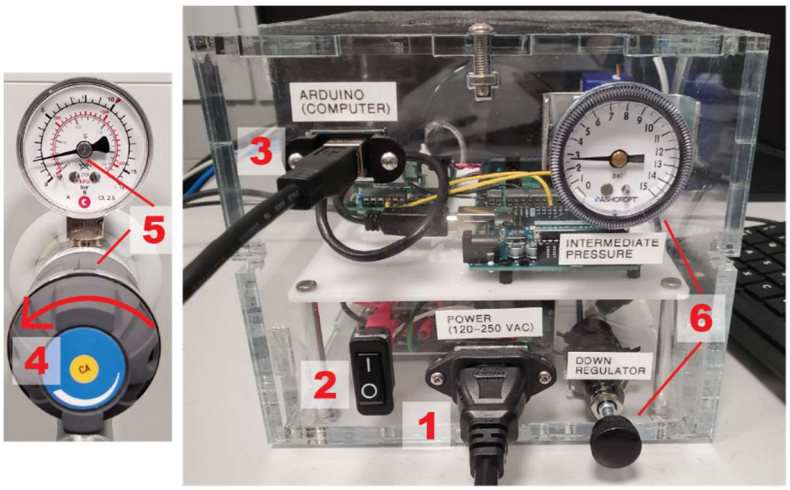


Figure 19. Pressure controller startup. 1. Connect power cable. 2. Turn on switch. 3. Connect Arduino/computer cable. 4. Turn on air supply. 5. Adjust air supply to ~30psi. 6. Down regulate to ~3psi.

4.4 Using Autoinjector 2.0 software

This section details the entire injection procedure. In general, using the Autoinjector 2.0 consists of opening the software, loading the pipette, submerging it into the tissue media, calibrating the manipulator, annotating the target, setting the injection parameters, and finally running the injection trajectory. Understandably, this can be daunting because of the large number of relatively complicated steps, but Autoinjector 2.0 aims to ease this burden. With a little practice, you will become proficient and Autoinjector 2.0 will help you achieve large-scale microinjection.

4.4.1 Open the Autoinjector 2.0 software

Autoinjector 2.0 software is located at: “C:\Users\Public\Documents\envs\Autoinjector_2”

1. Option 1: Right click on “autoinjector.ps1” and select “Run with Powershell”.[5] It may take 30 – 90 seconds to open the first time.
2. Option 2: Right click on “autoinjector.ps1”, select “Edit”, and click the green triangle button.[6] It may take 30 – 90 seconds to open the first time.

4.4.2 Load the pipette with microinjection solution

1. Using a 0-2um micropipette and Eppendorf microloader tips, draw-up 0.7um – 1.2um microinjection solution into the microloader tip.
2. Insert the microloader tip into the glass capillary pipette until the tip reaches the glass capillary pipette’s shoulder.
3. Gently depress the micropipette plunger to fill the glass capillary pipette with solution.

4.4.3 Submerge the pipette into tissue media

When handling the Sensapex manipulator, it is important to only handle the back and base! See Figure 20.

1. Tilt the Sensapex “up” to make it easier to insert the pipette into the pipette/electrode holder.
2. Insert the pipette into pipette/electrode holder until it is fully inserted.
3. Finger-tighten the screw to “lock-in” the pipette.
4. Slowly tilt the Sensapex “down” while ensuring the pipette will not hit the microscope.

[5] If an error occurs, the application will abruptly shut down without showing any context for why the error occurred.
[6] If an error occurs, the error will be printed in the terminal. This can be screen captured and saved for debugging purposes.

⚠ *Watch the pipette tip as you tilt the Sensapex "down" to prevent the pipette from colliding with the microscope. Raise the Sensapex "Z" and/or retract the "D" to prevent a collision.*

5. Drag pressure slider in Autoinjector 2.0 software to approximately 25% – 50% to pressurize pipette before submerging to prevent clogging.
6. Push the Sensapex all the way towards the microscope and lock it in place.
7. Advance "X" and "D" to approximately 15,000.
8. Lower "Z" until the pipette is submerged in the tissue media.
9. Sweep "Y" while watching the video feed until the pipette shank/tip is visible.
10. Use the "X", "Y", "Z" manipulator wheels to bring the pipette into focus.

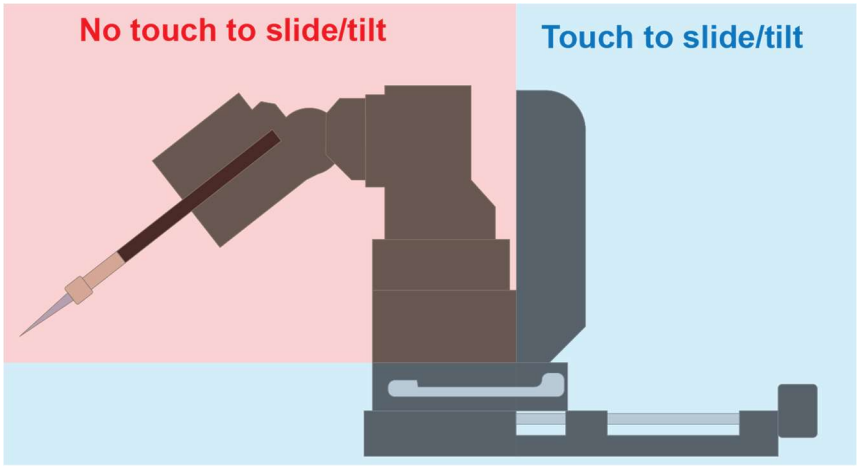


Figure 20. Where to handle the manipulator when you are sliding and tilting the manipulator.

4.4.4 Conduct manipulator-microscope calibration

1. Select "Automatic" for the pipette angle to automatically read the angle from the manipulator.
2. Select the desired calibration mode. Suggested: "Semi-Auto" or "Automatic".
3. Click "Conduct Calibration" to open the calibration window.
4. Run the calibration procedure. The tip should be in-focus and visible in the FOV, but "Automatic" calibration will autofocus the tip before calibrating.

⚠ *Ensure the FOV is free of any tissue or obstructions. During a "Semi-Auto" and "Automatic" calibration, the pipette will automatically move to the edges of the FOV, and it will collide with anything present in the FOV.*

a. For “Semi-Auto” calibration mode, click precisely on the pipette tip as the pipette automatically moves to the FOV corners and the center. Also click when it returns to the center.
b. For “Automatic” calibration mode, click “Run Automatic Calibration” and wait as the pipette automatically moves to the FOV corners and the center.
c. For “Manual” calibration mode (unrecommended), click precisely on the pipette tip. Move the pipette tip position with the manipulator wheels, click on the tip, and repeat for at least 3 tip positions.

⚠ *For “Semi-Auto” and “Manual” calibration, it is very important to be precise and consistent with where you click on the tip. Inconsistency will affect calibration accuracy.*

5. Click “Complete Calibration” or “Save and Complete Calibration”[7] to leave the calibration window and use the calibration for microinjection processes.[8]

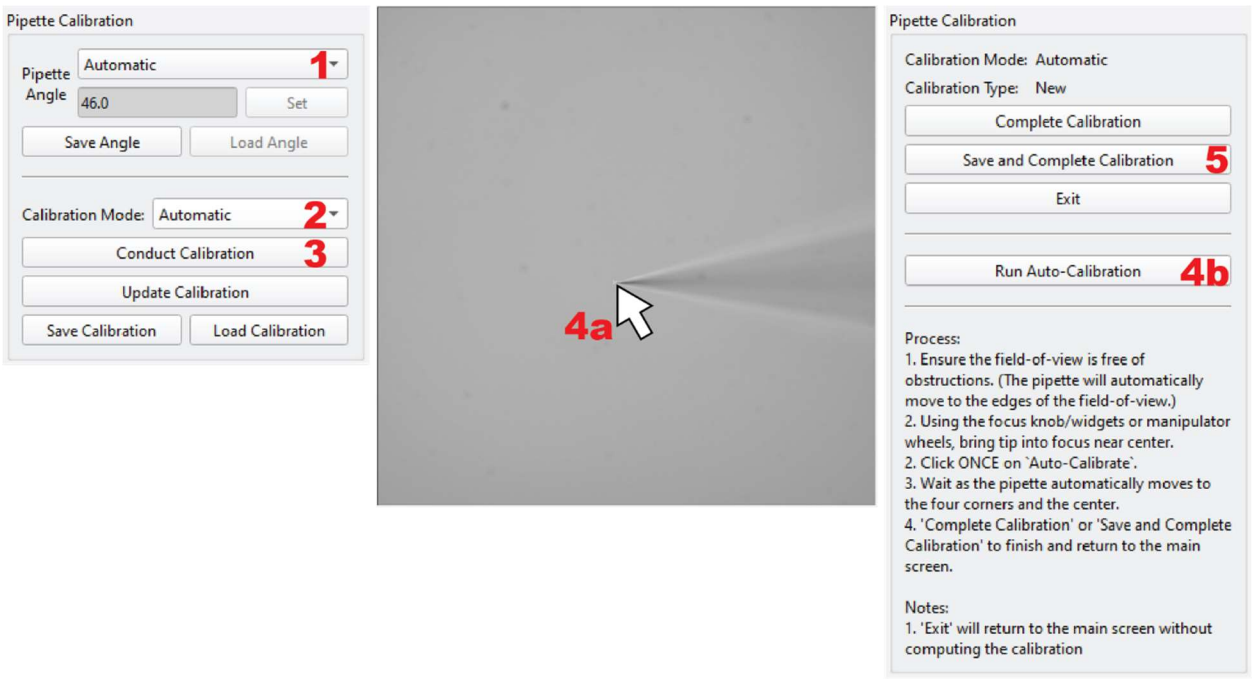


Figure 21. Microscope-manipulator calibration process. 1. Set pipette angle. 2. Select calibration mode. 3. Open calibration. 4a. Click on tip for semi-auto calibration. OR 4b. Run auto-calibration. 5. Save and finish calibration.

[7] “Save and Complete Calibration” (or “Save calibration” in the main window) will save the calibration to a file. This will enable the calibration to be loaded even after the Autoinjector 2.0 application is closed and reopened.
[8] “Exit Calibration” will leave the calibration window without using the calibration for microinjection.

4.4.5 *Annotate injection targets*

1. Bring the desired tissue section into focus and click “Annotate Target” button to open the annotation window.
2. Select Annotation Mode – either “Manual” or “Automatic”.
3. Annotate the microinjection targets.

> ⚠ *If you want to conduct injections at a single point: draw a very short annotation (but it needs to be longer than 3 pixels). In the next section, set a large “spacing” parameter (like 1000) and 0 “depth” parameter.*

   a. For “Manual” annotation mode, left-click-and-drag along the tissue edge to inject. This process can be repeated at numerous focus heights to inject at different z-positions.
   b. For “Automatic” annotation mode:
      i. Select the type of edge to inject. “Reachable” – any type of edge that can be reached by the pipette. “Apical” or ”Basal” – the tissue’s apical or basal edge as detected by the tissue detection program. See section **3.2.3** for more information.
      ii. Option 1: individual annotations. “Run Single Auto. Annotation” to perform automatic tissue detection and annotation on the current focal plane. This process can be repeated at numerous focal planes to inject at different z-positions.
      iii. Option 2: z-stack annotation. “Set first focus” and “Set last focus” to set the z-stack limits. Set “Number of slices” to define number of z-stack levels. “Run Z-Stack Auto. Annotation” to perform automatic tissue detection and annotation on each focal plane in the z-stack.
4. Click “Complete annotation” to use the annotations for the microinjection process.[9]

[9] “Exit annotation” will leave the annotation window without using the annotation for microinjection.

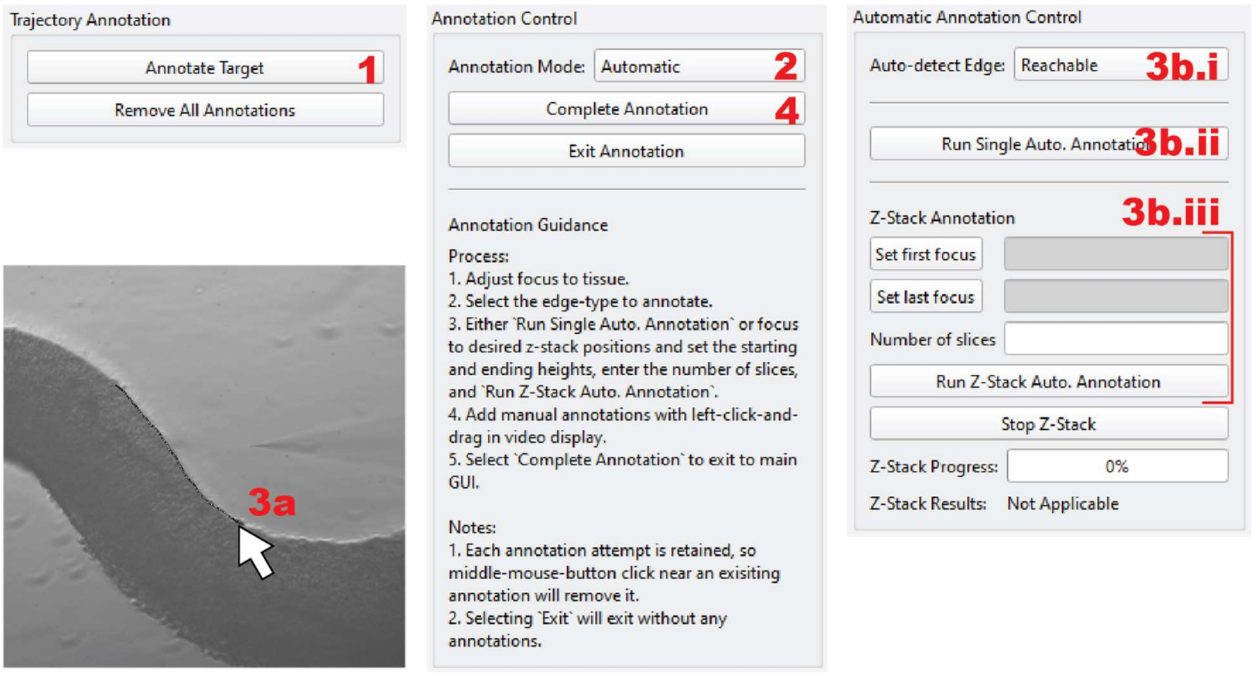


Figure 22. Target annotation process. 1. Open annotation window. 2. Select annotation mode. 3a. Manual Mode: left-click-and-drag along tissue edge. OR 3b.i. Auto mode: Select edge to detect. 3b.ii. Auto mode: Detect edge in current focal plane. OR 3b.iii Auto mode: Set z-stack limits, number of slices, and detect edges in z-stack. 4. Complete annotation.

4.4.6 Set injection parameters and run injections

1. Enter numeric values for the parameters: “Approach”, “Depth”, “Spacing”, and “Speed”.
    a. “Approach” (recommended 75 – 150) is the distance from the tissue that the pipette starts the injection trajectory.
    b. “Depth” (recommended 10 – 20 for apical and 30 – 45 for basal) is the distance into the tissue that the pipette injects.
    c. “Spacing” (recommended 30 – 75) is the distance between injection sites.
    d. “Speed” (recommended 750 – 1500) is the speed of the manipulator.
2. Click “Set Values” to use the parameter values for the microinjection process.
3. Drag the “Pressure” slider to the desired value (recommended 30 – 75%). The red electronic manometer should display 0.5 – 1.75psi.
4. Select “Track Annotations” if you believe tissue movement during injections will cause the tissue to misalign with the drawn annotations. This feature is available for both manual and automatic annotations.

⚠ *“Track Annotations” will only track the annotations while the manipulator is injecting, so ensure that the annotations align with the tissue before starting the injections.*

⚠ *"Track Annotations" for injection trajectories with multiple annotations (3 or more annotations) may compromise the camera frame rate during tracking.*

⚠ *Tracking accuracy may be low if the annotations are short or if the tissue doesn't have much visible texture.*

5. Confirm all indicators in the "Injection Workflow" panel are green. Repeat any processes for which the indicators are red.
6. Click "Run Trajectory" to start the injection procedure. Click "Stop Process" at any time to stop the injection procedure.
    a. If you notice that the pipette tip is out-of-focus at the injection positions, follow the advice in the trouble shooting.

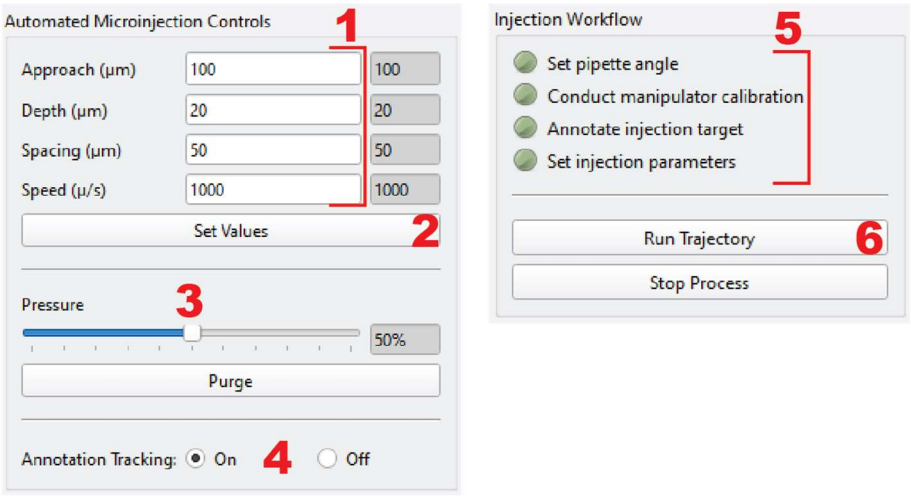


Figure 23. Running injections. 1. Input numerical values for injection parameters. 2. Set values for injection procedure. 3. Set pressure. 4. Enable/disable annotation tracking. 5. Verify all tasks completed (green). 6. Run injection trajectory.

4.4.7 Unloading/reloading pipettes

1. Adjust the microscope stage to a FOV that is clear of tissue or other debris.
2. Click "Go to 'unload'" to move the manipulator to the unload position.[10] This will turn off the pressure.
3. Unlock the manipulator, slide it away from the microscope, and re-lock the manipulator.
4. Tilt the Sensapex "up", loosen the pipette clamp, remove the pipette, and discard it in sharps disposal.

[10] This moves the manipulator to Z=1000 and then D=1000.

5. Perform steps in sections **4.4.2** and **4.4.3** up to fill a new pipette with injection solution and insert in into the manipulator. Conduct all steps up through “Push the Sensapex all the way towards the microscope and lock it in place”.
6. Click “Undo previous” to move the manipulator to the position it was at when “Go to ‘unload’” was clicked.[11] This will turn the pressure back on to what it was before “Go to ‘unload’”.
7. Bring the pipette tip into focus.
8. Click “Update calibration” to open the calibration window.
9. Click on the tip to mark the new pipette tip position.
10. Click “Complete Calibration” or “Save and Complete Calibration” to leave the calibration window and use the updated calibration for microinjection processes.
11. Adjust the microscope stage back to the FOV with tissue.
12. Repeat sections **4.4.5** and **4.4.6** to annotate microinjection targets and run the injection trajectory.

4.4.8 Video screen capture of Autoinjector 2.0 software

This is used if you want to record a screen capture of the injection process. This only works on Windows computers.

1. Autoinjector 2.0 software must be open and running.
2. Ensure that the Autoinjector software window is the currently active window (click on the software window).
3. Press “win+g” on your keyboard to open the recording toolbar. (“win” is the key that looks like the Windows logo and it’s typically the second or third key from the bottom left).
4. Press the record button.

4.5 Shutdown

Shutting down the Autoinjector 2.0 is essentially the reversed process of the startup procedure. It is important to turn off the wall air supply because leaving it on can cause unnecessary stress

[11] If the manipulator position was (X=14000, Y=9000, Z=12000, D=16000) when “Go to ‘unload’” was clicked, then it will return to this position upon “Undo previous”. However, slight deviations in the length of the pipette or the small deviations in where/how the manipulator is locked will cause the pipette to not be in the exact same position as before “Go to ‘unload’”. (Actually, the manipulator goes to a slightly modified Z-position of Z=11500. This small displacement prevents the pipette from colliding with the bottom of the tissue chamber if the new pipette is longer than the previous pipette.)

on the tubing couplings/fittings. This may lead to leaks and require replacing the couplings and fittings.

1. Close the Autoinjector 2.0 software.
2. Turn off the air supply.
3. Turn off the pressure controller.
    a. Disconnect the Arduino/computer cord.
    b. Turn the power switch off.
    c. Disconnect the power cord.
4. Turn off the manipulator. To turn it off you need to press and hold the center button until the shutdown window is shown.
5. Turn off the microscope.
    a. Turn off the camera.
    b. Close the Zeiss ZEN software.
    c. Turn off the microscope.
    d. Turn off the microscope power supply: Power supply 232.
    e. Turn off the ancillary devices: Focus Controller 2 and SMC 2009.
6. Turn off the computer.

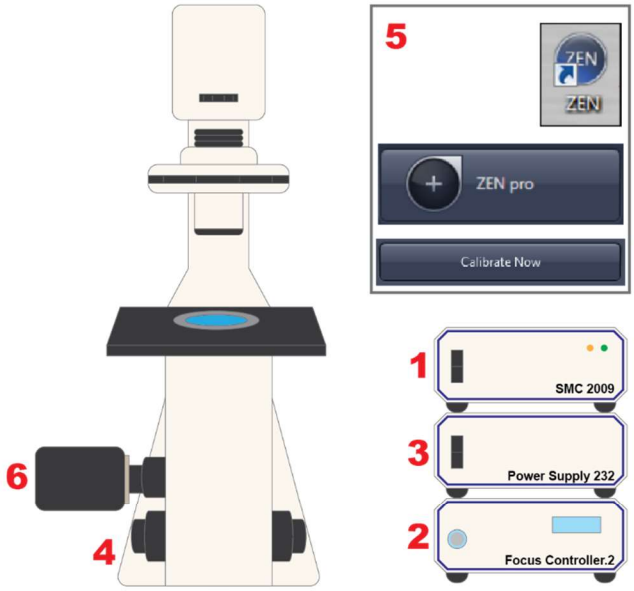
4.6 Condensed Procedure

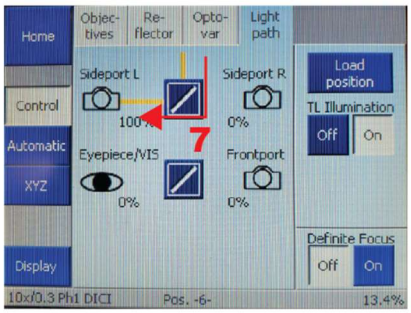
This section briefly describes the whole procedure and shows a compilation of figures depicting the necessary steps. This can be used as a reference once you are proficient, and don’t need the highly detailed guidance above.

1. Pre-injection preparations
    a. Make injection solution.
    b. Pull glass capillary pipettes.
    c. Gather supplies for filling glass capillary pipettes.
    d. Prepare tissue.
2. Microscope hardware startup
    a. Turn on microscope hardware (excluding camera).
    b. Start ZEN software and perform microscope calibration.
    c. Turn on camera.
3. Autoinjector 2.0 hardware startup
    a. Turn on Sensapex manipulator. (Optional load calibration).
    b. Turn on pressure controller electronics.

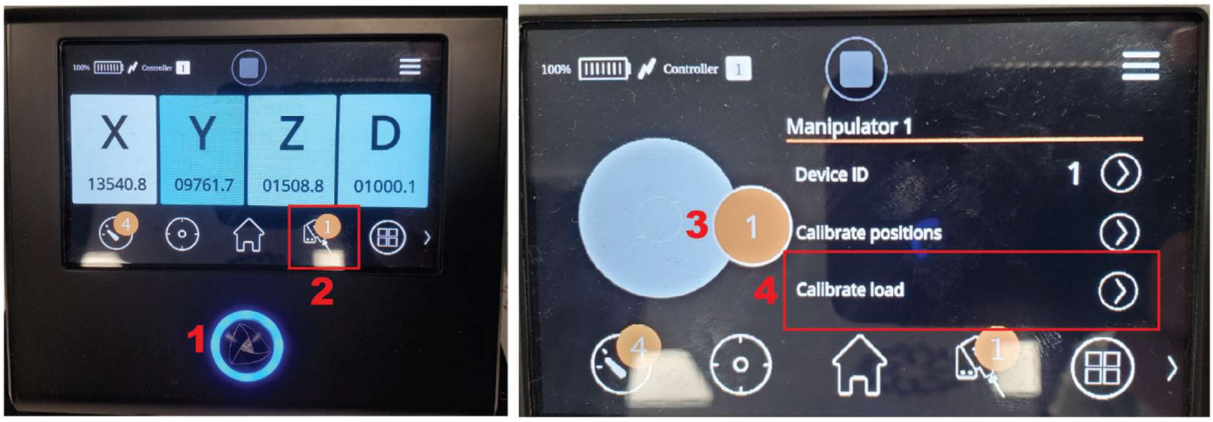
    c. Open air supply.
    d. Adjust air supply gauge to 20-30psi and “INTERMEDIATE PRESSURE” to 2.5-3.5psi.
4. Open Autoinjector 2.0 software.
5. Fill pipette with injection solution and insert into electrode/pipette holder.
6. Adjust pressure slider in Autoinjector 2.0 software to apply compensation pressure.
7. Manually submerge pipette into tissue bath and bring into focus.
8. Conduct pipette-microscope calibration.
9. Focus on tissue section of interest.
10. Annotate injection target.
11. Bring pipette into focus and update calibration.
12. Set automated injection parameters.
13. Run Trajectory.
14. “Go to ‘unload’” to change pipettes and “Undo previous” to re-submerge pipette.
15. Update calibration
16. Repeat at step 9.

Microscope startup

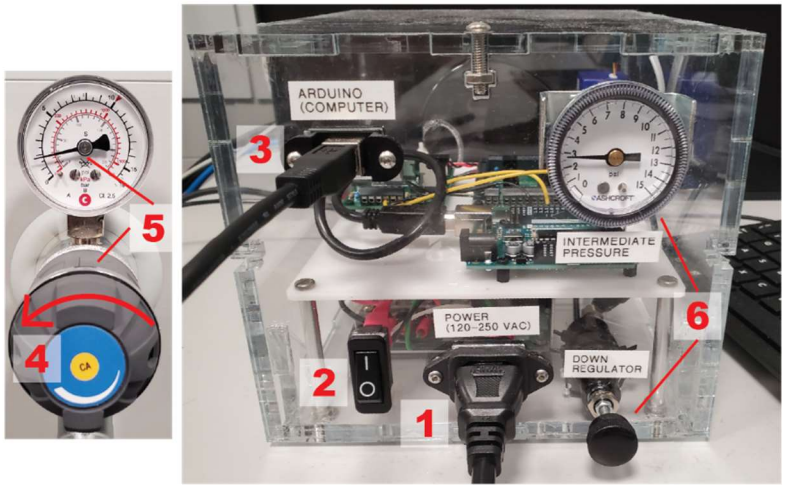




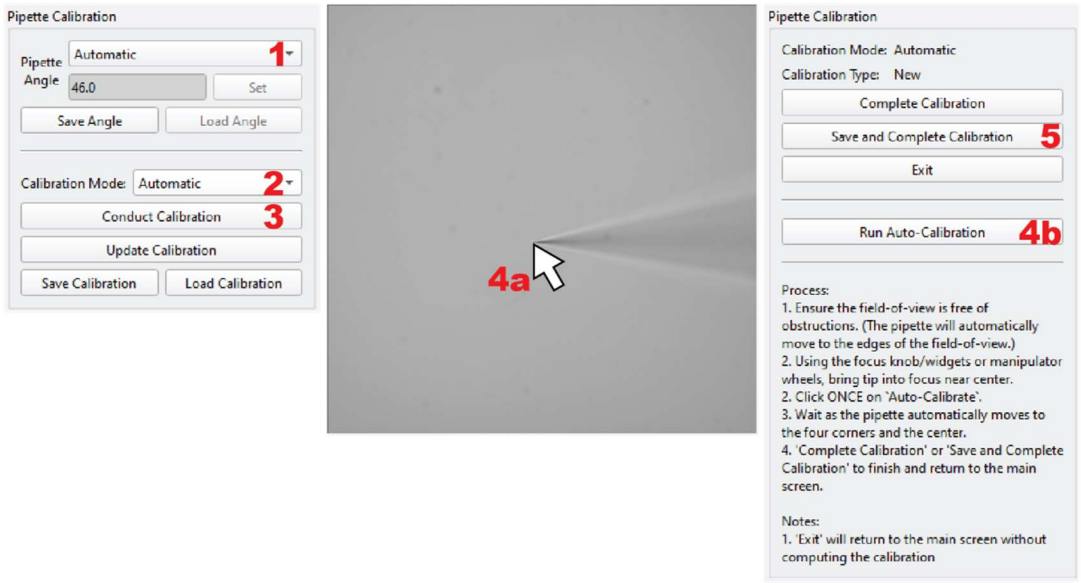
Manipulator startup



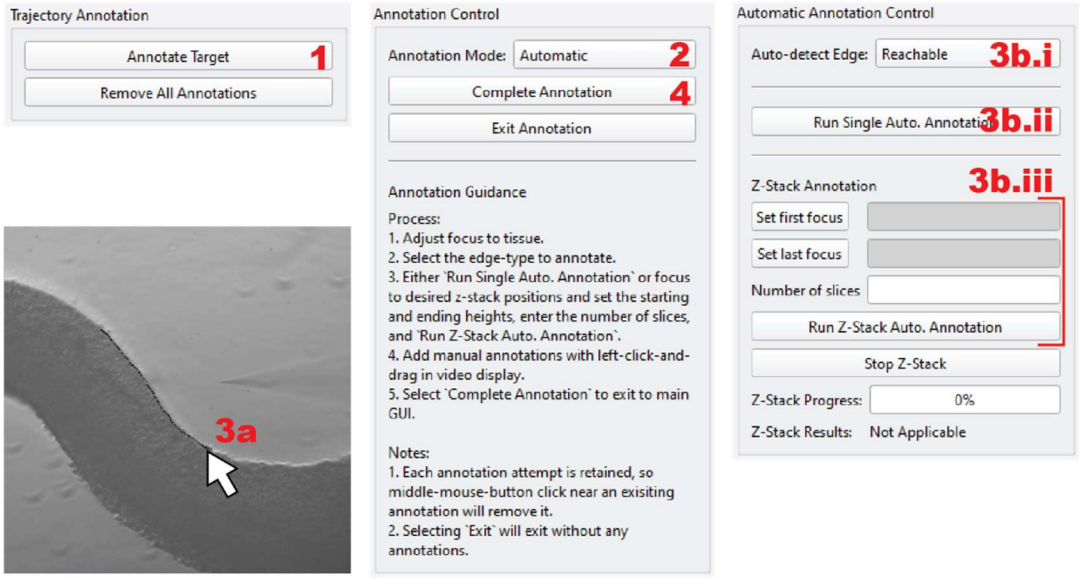
Pressure controller startup



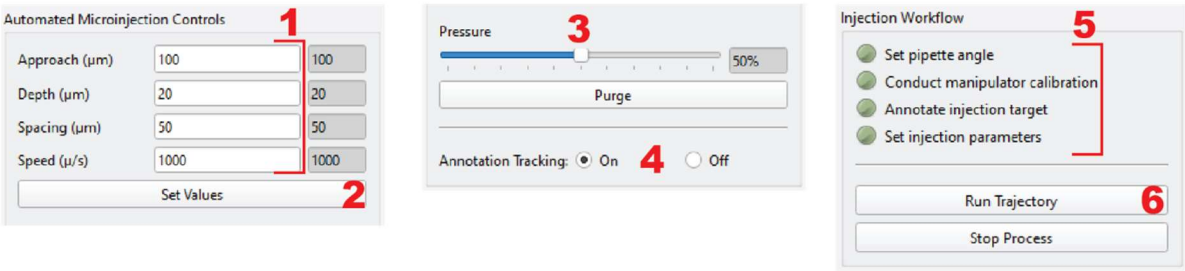
Microscope-manipulator calibration



Injection target annotation



Set parameters and run injection



5 Troubleshooting

Problem: Camera feed is black. I the video display in the GUI doesn't display anything.

Potential Solutions:

- Ensure the Microscope light path is set to the camera's port.
- Ensure the transmitted light illumination (lamp) is on.
- Ensure the transmitted light illumination (lamp) power is sufficient.

Problem: The pipette is filled with fluorescent dye, but I can't see the dye fluorescence.

Potential Solutions:

- Ensure the correct LED excitation wavelength is selected for the specific fluorophore.
- Ensure adequate LED excitation power.
- Ensure the active Microscope Reflector is compatible with the fluorophore.
- Fluorescent dye concentration is too low.
- Pipette is clogged.

Problem: The pipette is clogged and not injecting/expelling any injection solution from the tip.

Potential Solutions:

- Try to clear the clog by pressing the "Purge" button to send a high pressure pulse to the pipette.
- Try very gently touching the pipette tip to the bottom of the petri dish (NOT RECOMMENDED!)
- Change pipettes to a new, unclogged pipette.

Problem: The pipette tip is out-of-focus at the injection positions (when it should be in-focus).

Root Cause: This is caused by an inaccurate z-calibration.

Potential Solutions:

- Ensure the pipette tip is in-focus when you conduct a new calibration or update a calibration.
- Before running injections, update the calibration with the pipette tip in-focus at the same microscope focus height as the annotations.

- Open the configuration app and change the z-scaling parameter. The default z-scaling parameter at Human Technopole is 0.741 for air immersed objective and aqueous solution immersed tissue and 1.0 for air immersed objective and air immersed tissue.

Problem: Tissue is moving during injections making my injections inaccurate.

Potential solutions:

- Increase the size/weight of the tissue.
- Conduct injections in a less dense part of the tissue.
- Reduce the penetration depth.
- Turn on annotation tracking (section **3.3.3**).
- Manually compensate for tissue movement by using the microscope stage joystick to realign the tissue with the annotation.

Helpful tips maybe

1. Fill the pipette horizontally while touching the micorloader tip to the filament
2. Use the stage joystick to compensate for tissue movement.
3. Single point injections by drawing a short annotation (must be at least 3 pixels), specify spacing as large (500) and depth to 0. This will always inject at the first location that you clicked.

Appendix A. Pipette programs

All programs listed below are for the Sutter Instruments P-1000 micropipette puller using Sutter BF120-94-10 capillary glass (1.2mm outer diameter and 0.94mm inner diameter with filament) and Sutter FB255B heating filament (2.5mmx2.5mm box). All Heat values are relative to the glass's Ramp value (aka if Heat=+5 and Ramp=500, the true Heat value is 505).

Pipette geometry is very important for making a pipette that doesn't clog but also is this and small enough to inject cells with minimal damage. When evaluating pipettes, you should not see any visible plume if dye in DIC. The most optimal programs that we found at Human Technopole are in shown in Table 2.

Table 2. Pipette programs with good balance between tip geometry and clogging. The best programs (for me) are the last 2 entries.

Heat	Pull	Vel	Del	Pressure	Notes
+0	50	85	90	450	Infrequent clogging but pipette tip is large
+2	52	87	90	375	Still infrequent clogging with smaller tip
+5	52	87	90	350	Good. Still infrequent clogging with even smaller tip.
+5	52	87	90	325	Best. Still infrequent clogging with even smaller tip and slightly more taper

Some of my further experimentation is listed below when I was striving to find a pipette geometry according to Elena Taverna's suggestion for a long, cylindrical taper with a small tip. I don't recommend using any of these programs, but you may find it useful as a reference for your own experimentation.

Program 30. Heat=510, Pull=70, Vel=80, Delay=90, Pressure=250
Similar morphology to what you described with a long taper. This clogged within 5 minutes of submerging in the media (without injections).

Program 31. Heat=505, Pull=70, Vel=80, Delay=100, Pressure=200
This is very similar to program 30, but it seemed to clog quicker.

Program 32. Heat=515, Pull=60, Vel=70, Delay=80, Pressure=250
Has a slightly shorter taper than the previous programs, but seemed to clog similarly.

Program 33. Heat=515, Pull=60, Vel=55, Delay=100, Pressure=250
Medium taper with a fine tip, but this seemed to clog relatively quickly (within 5 minutes of submerging in media with no injections).

Program 34. Heat=500, Pull=50, Vel=60, Delay=90, Pressure=250
This pipette hardly clogged, but I think it is too short/stubby and has too large of a tip.

Program 35. Heat=500, Pull=45, Vel=65, Delay=105, Pressure=250
This has a larger tip and a shorter taper than 34. This pipette didn't clog at all but is probably inadequate for injections.

Program 36. Heat=510, Pull=55, Vel=70, Delay=110, Pressure=200
I think this has the morphology you described (long and almost cylindrical taper). It has a fine to very fine tip and long taper. This clogged within 5 minutes of submerging in the media (without injections).

Program 37. Heat=515, Pull=65, Vel=60, Delay=110, Pressure=200
I also think this has the morphology you described (long and almost cylindrical taper). Likewise, it has a fine to very fine tip and long taper. Overall, very similar to program 36. This appeared to clog slightly quicker than 36.

Program 38. Heat=505, Pull=55, Vel=65, Delay=95, Pressure=225
This has been the most successful (reasonably shaped) pipette with infrequent clogging. The morphology appears shorter than you described, but it is similar to the shape I use at UMN. It has a fine to medium-fine tip and a medium taper. This pipette did numerous injection planes (6+) without clogging.

Appendix B. Explanation of Configuration application

The configuration application can be opened by running the "configure_application.py" Python file. At Human Technopole, you can run this application by right-clicking on the "configure.ps1" file and clicking "Run with PowerShell" (this "configure.ps1" simply runs the "configure_application.py" file). You can also open the configuration application if the main application is open/running by clicking on "Menu" in the top left and selecting "Open Configuration".

In the configuration application, the user will tell the Autoinjector system which hardware devices to use, software settings for the calibration process, and the directory containing the Micro-Manager application (which controls the camera). For example, the configuration application enables the user to specify the camera to be used (of the potentially multiple cameras connected to the computer), and the COM port of pressure controller communication with the computer. The Autoinjector system will not work without knowing which camera to use or how to communicate with the pressure controller. Descriptions of all the parameters in the configuration app are given in Table 3.

Once the user selects and enters the parameter values in user interface, they must press the "Save Configuration" button to save the values to a file. The Autoinjector system will use this file to initialize itself when the main application is opening. After you save the configuration, you can press "Open Autoinjector" if you want to open the main application. You can also "Open

Autoinjector” without first “Save Configuration” if the configuration file already exists from a previous configuration procedure.

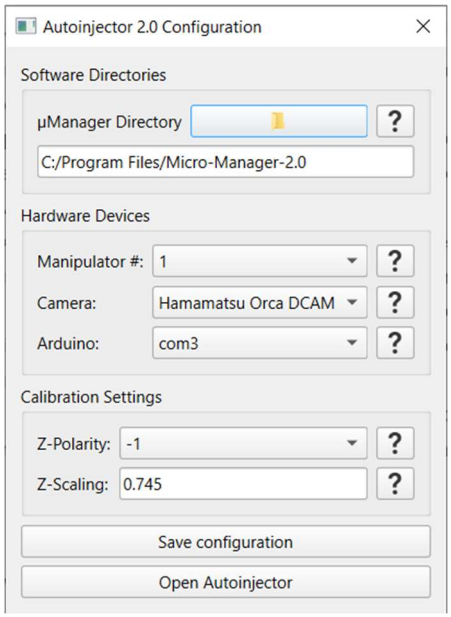


Figure 24. Configuration application where user specifies important operational parameters for Autoinjector 2.0.

Table 3. Description of configuration parameters. These parameters are specified in the configuration application by the user to define how the Autoinjector system works.

Configuration parameter	**Description**
µManager Directory	Specify the file directory location where you installed Micro-Manager (µManager) during the GitHub installation procedure. Micro-Manager is an open-source software that is used for controlling various microscope hardware. The Autoinjector system uses the Micro-Manager to communicate with the camera. Hence, you must tell the Autoinjector system where the Micro-Manager software is installed so it can use Micro-Manager to interface with the camera.
Manipulator Device Number	Specify the index of the Sensapex micromanipulator device that you are using in the Autoinjector system. You can determine the device number by locating the number in the orange circle in the manipulator touch screen unit. If you only have one device, then the device number is probably 1. This tells the Autoinjector which device (of potentially multiple) to use for microinjections.

Camera	Specify the camera that is connected to the Autoinjector microscope. There may potentially be multiple cameras connected to the computer, so this tells the Autoinjector system which one to use. Moreover, different cameras require slightly different initialization procedures, so this also tells the Autoinjector how to initialize the camera.
Arduino	Specify the COM port that is used by pressure controller Arduino to communicate with the user. This tells the Autoinjector system the correct communication address of the pressure controller, so it knows where to send its pressure commands.
Z-Polarity	Specify the “polarity” between the microscope z-axis and manipulator z-axis. The “polarity” is +1 if both axes’ values increase in the same direction and its -1 if they increase in opposite directions. For instance, if you move both axes towards the tissue chamber and the microscope z-height value decreases and the manipulator z-value increases (as is the case at Human Technopole) then the polarity is -1. This polarity is used as a constant value in the microscope-manipulator calibration process so that the user doesn’t need to do an extra z-calibration step.
Z-scaling	Specify the “scaling’ between the microscope z-axis and the manipulator z-axis. The “scaling” is the absolute value ratio of microscope z-axis change to manipulator z-axis change for the same reference distance. For instance, start with the pipette tip in-focus in the microscope. Then move the microscope z-height towards the tissue chamber. Then move the manipulator z-height to bring the pipette back into focus (with the microscope position constant). If the microscope z changed by -750um and the manipulator z changed by +1000, the scaling is |-750/1000| = 0.75. At Human Technopole the value is 0.741 (for an air immersion objective and aqueous solution immersed tissue).

Appendix C. Explanation of Autoinjector 2.0 application

The application is how a user will interact with the Autoinjector 2.0 system and control the microinjection process. It provides various widgets and functionalities that enable the user to modify the behavior of the system.

To separate distinct functionalities of Autoinjector 2.0, the user-interface will display one of three separate windows depending on the current tasks being performed. These three windows include:

- Main user interface window (Figure 25): This is the default window of the user interface. The user can start various procedures (calibration, annotation, injection, etc), control the microscope configuration, and set the microinjection parameters.
- Calibration window (Figure 26): This window is active when the user is conducting a microscope-manipulator calibration.
- Annotation window (Figure 27): This window is active when the user is annotating the microinjection targets.

The following sections will briefly explain every widget in each of these windows.

C.1 Main user interface window

Figure 25. Main user interface window. This is the default window for the Autoinjector 2.0 where the user will control the microinjection system. A piece of agarose is being used as an alternative to tissue.

C.1.1 Menu bar

The menu bar provides access to several more specialized and infrequently used features of the Autoinjector 2.0 system. The menu bar options are described in Table 4.

Table 4. Menu bar options in the Autoinjector application.

Menu option	Description
“Open Configuration”	Closes the Autoinjector application and opens the configuration application where the user can specify important parameters of the Autoinjector system.
“Standard Mode”	This changes the user-interface to standard mode the default interface that you see when you open the application.
“Practice Mode”	This modifies the video display to show a mixture of the live camera feed and an image of tissue. In practice mode, you can practice mock injections with a pipette without risk of damaging tissue or colliding with the pipette with anything. This can be useful when your first learning how to use the system and understanding the principles of robotic microinjection.
“Developer Mode”	This displays several buttons in the main user interface window that allow you to save images during the calibration annotation process. These buttons allow the user to automatically save images from the camera and the coordinates of the calibration positions during the calibration process. They also save the images (and z-stacks) and annotation coordinates during the annotation process. These images and data can be useful for generating data for deep learning training (especially for automatically acquiring data on new tissue to train a tissue detector).

C.1.2 Pipette calibration widgets

These widgets control the manipulator-microscope calibration. With these widgets, the user can define the manipulator diagonal axis angle, start a new calibration procedure, start an update calibration procedure, and save/load existing calibrations. These widgets are described in Table 5.

Table 5. Description of pipette calibration widgets in the main user interface window.

Widget	Description
“Pipette angle” dropdown	Sets how the system determines the manipulator diagonal axis error.

	• "Automatic": Uses the manipulator's internal sensors to measure the angle. This should match the angle displayed in the "Poll Axis Angle" function on the manipulator touch screen. • "Manual": The user must manually set diagonal axis angle. Only choose this option if you believe "Automatic" is inaccurate.
"Pipette angle" entry box	This is where the user would enter an angle value (in degrees) for the manipulator's diagonal axis angle. It also displays the angle when in "Automatic" mode. This is only editable when in "Manual" mode.
"Set" button	Sets the angle to the value in the entry box. This is only editable when in "Manual" mode.
"Save Angle" button	Saves the angle value in the entry box to a file. This angle value can be loaded later (even after closing and reopening the software).
"Load Angle" button	Loads and sets the manipulator diagonal axis angle to the most recently saved angle value. This is only editable when in "Manual" mode.
"Calibration Mode" dropdown	Sets how the system will conduct or update the calibration when the user opens the calibration window. • "Automatic": Gives user opportunity to use computer vision to autofocus the pipette tip and detect the pipette tip while the manipulator automatically moves between calibration positions. • "Semi-Auto": The manipulator will automatically move between calibration positions, but the user must manually click on the pipette tip in the video display (no computer vision). • "Manual": The user must manually move the manipulator between calibration positions and manually click on the pipette tip in the video display.
"Conduct Calibration" button	Opens the Calibration window for conducting a ***new*** calibration procedure.

“Update Calibration” button	Opens the calibration window for conducting an ***update*** calibration procedure.
“Save Calibration” button	Saves the current calibration to a file. This calibration can be loaded later (even after closing and reopening the software).
“Load Calibration” button	Loads and sets the manipulator-microscope calibration to the most recently saved calibration. Loading and updating a calibration will be faster than an entirely new calibration but loading/updating should only be used if you know the calibration should be the same (or similar) to the most recent calibration.

C.1.3 Trajectory annotation widgets

These widgets are used to begin annotating injection targets and removing existing annotations. These widgets are described in Table 6.

Table 6. Description of trajectory annotation widgets in the main user interface window.

Widget	**Description**
“Annotate Target” button	Opens the Annotation window where the user will draw injection target annotations.
“Remove All Annotations” button	Erases any existing target annotations. You will be able to repeatedly run injections on the existing annotations until the annotations are removed (or you open the Annotation window which also removes existing annotations).

C.1.4 Automated microinjection controls widgets

These widgets control the behavior of the Autoinjector 2.0 system during the automated microinjection procedure. With these widgets, the user can modify the injection trajectory like the depth of injections and the spacing between injections, and they can set the compensation pressure for expelling the injection solution from the pipette. Additionally, the user can turn off/on annotation tracking to compensate for tissue movement during injections. These widgets are described in Table 7.

Table 7. Description of automated microinjection control widgets in the main user interface window.

Widget	**Description**

“Approach” entry box	Where the user enters the microinjection approach distance. The approach distance specifies the distance from the annotation that the manipulator will begin the injection trajectory
“Depth” entry box	Where the user enters the microinjection depth. This is the distance past the annotation that the manipulator will penetrate the tissue during injections.
“Spacing” entry box	Where the user enters the microinjection spacing. This is the distance between injection locations in the tissue.
“Speed” entry box	Where the user enters the microinjection speed. This is the manipulator speed as it moves during the injection process. All manipulator moves are conducted with the same speed.
Display boxes next to entry boxes	These boxes display the currently set injection parameters. If these boxes are red, it means that the value in the entry box does not match the currently set parameter value.
“Set Values” button	Sets the microinjection parameters to the values in the entry boxes. The values in the entry boxes are “inactive” until this button is clicked. For instance, if you change depth from 30 to 50, but you don’t click this button then it will inject at a depth of 30.
“Pressure” slider	Sets the constant compensation pressure. Setting to a value above 0 will turn on the pressure, but you will likely need values above 15%-20% to actually create a positive compensation pressure (because of the way the pressure controller works).
“Pressure” display box	Displays the current pressure value. If the box is red, the pressure is not set, so try to drag the pressure slider until this box is grey. This box displays pressure as a percentage, but the actual pressure value should approximately correspond to a percentage of the pressure controllers full-scale-range. For instance, our pressure controller is 0-2psi, so 50% corresponds to ~1psi.
“Purge” button	Sends a high-pressure pulse of 30psi (or your air supply pressure) for 0.5s to the pipette to attempt unclogging.
“Annotation Tracking” on/off	Turns off/on annotation tracking to track the annotations **during the microinjection process**. Annotation tracking is only active while

	injections as it was designed to accommodate for tissue movement caused by the pipette poking this tissue. Notes: • This must be turned on before starting the injection process. • Annotations must be aligned with the tissue before starting the injection process otherwise it will not track properly. • Turing this off during injections will make this feature unavailable until you start a new injection process.

C.1.5 Injection workflow widgets

These widgets display the completion status of the steps that must be finished before beginning the microinjection process. Additionally, these widgets allow the user to start and stop the microinjection process. These widgets are described in Table 8. Description of injection workflow widgets in the main user interface window.

Table 8. Description of injection workflow widgets in the main user interface window.

Widget	**Description**
Status light indicators	These widgets display the completion status of the requisite steps that must be completed before starting microinjection. If the circle is red, that means that the step is incomplete and if it is green then the step is complete. If any indicators are red, you will not be able to start injections.
“Run Trajectory” button	Starts the microinjection process.
“Stop Process” button	Stops the microinjection process immediately without completing all injections.

C.1.6 Microscope video stream widget

This widget displays the video from the microscope camera, so the user can observe the microinjection process. This widget is also interactive in the Calibration and Annotation windows to facilitate completion of the calibration and annotation process.

Table 9. Description of the video stream widget.

Widget	Description
“Video stream” display	Shows a live video feed from the microscope camera. The video display also shows various annotations and overlays to help the user understand and observe the microinjection process (overlays are described in section **C.1.9**). Additionally, the user will interact with this widget during the calibration and annotation processes to mark calibration points and draw/annotate injection targets using the computer mouse.

C.1.7 Zeiss control widgets

These widgets permit computer control of the Zeiss microscope. With these widgets, the user can modify the microscope configuration by changing the focus height, changing the objectives/optovars/reflectors, and adjusting the light sources and light intensities. The widgets are described in Table 10.

Table 10. Description of the Zeiss microscope control widgets.

Widget	Description
“Focus” display box	Displays the microscope’s current focus height.
“-“ button	Decreases the microscope’s focus height by the value indicated in the “Increment” entry box.
“+“ button	Increases the microscope’s focus height by the value indicated in the “Increment” entry box.
“Increment” entry box	Specifies the distance moved when changing the microscope focus height with the “-“/”+” buttons.
“Objective” dropdown	Allows the user to select and change the microscope objective.
“Optovar” dropdown	Allows the user to select and change the microscope optovar.
“Reflector” dropdown	Allows the user to select and change the microscope reflector.
“Lamp” checkbox	Turns off (circle is blank/white) or on (circle is filled/black) the microscope transmitted light lamp.
“385”/”430”/etc checkboxes	Turns off (circle is blank/white) or on (circle is filled/black) the microscope’s excitation LEDs with wavelengths corresponding to the button label (e.g. “385” = 385nm LED excitation).

“Lamp” slider	Adjusts the intensity/power of the microscope’s transmitted light lamp.
“LED” dropdown	Selects the excitation LED whose intensity is modified by the “LED” slider.
“LED” slider	Adjusts the intensity/power of the excitation LED that is specified in the “LED” dropdown.

C.1.8 Manipulator control widgets

These widgets provide utility functions to automatically move the manipulator to the “unload” and ”reload” positions. These widgets handle manipulator movements to facilitate the pipette changing process that is required when the pipette is clogged. The widgets are described in Table 11.

Table 11. Description of the manipulator control widgets in the main user interface window.

Widget	**Description**
“Go to ‘unload’” button	Moves the manipulator to the “unload” position and sets the pressure to 0 so the user can change the pipette. The manipulator moves to the “unload” position by moving its Z- and D-axes both to value of 1000. This raises the pipette and retracts the pipette. The pressure is set to 0, so the pipette is not propelled out of the pipette holder when the user untightens the screw. Example: (X=16000, Y=10000, Z=8000, D=14000) → “Go to ‘unload’” → (X=16000, Y=10000, Z=1000, D=1000)
“Undo previous” button	Goes to the last position of the manipulator before the user most recently clicked “unload” or “undo”. • If user just did “Go to ‘unload’”, the pressure will turn back on to its previous value, and the manipulator will return to the position before “unload” was clicked. However, the Z-axis will be offset by 500um to prevent the pipette from colliding with things if the new pipette is longer than the previous one. Example: (X=16000, Y=10000, Z=8000, D=14000) → “Go to ‘unload’” → (X=16000, Y=10000, Z=1000, D=1000) → “Undo previous” → (X=16000, Y=10000, Z=7500, D=14000)

	• If the user just did “Undo previous” and the clicks “Undo previous” again, then the manipulator will just return to the previous position and not change the pressure. Example: (X=16000, Y=10000, Z=8000, D=14000) → “Go to ‘unload’” → (X=16000, Y=10000, Z=1000, D=1000) → “Undo previous” → (X=16000, Y=10000, Z=7500, D=14000) → “Undo previous” → (X=16000, Y=10000, Z=1000, D=1000)

C.1.9 Display setting widgets

These widgets modify the appearance of the video display. Except for the camera exposure slider, these widgets don’t physically modify the video – they simply change the annotations and overlays that are displayed on the video screen. These widgets allow the user to display or hide things like the system’s prediction of the tip location and the injection annotations. These widgets are described in Table 12.

Table 12. Description of the display setting widgets.

Widget	**Description**
“Camera exposure” slider	Adjusts the exposure value for the camera. Changing the exposure will also affect the camera’s frame rate (frames per second).
“Show tip” on/off	Turns off/on an overlay that shows the system’s prediction of the pipette tip location by drawing a circle depicting the tip location. The location is computed by using the manipulator’s position and the manipulator-microscope calibration to calculate the tip position as pixel coordinates. Hence, this is only visible after calibration is completed. If the circle tracks the pipette tip, the calibration is good. The circle changes size and color depending on the tip being above or below the current focus height. If the circle is red, the system computed the pipette tip as being below the current focus height, and if it is blue, then it is above the current focus height. The size of the circle indicates the distance from the current focus height (pipette tip being father from the current focus height = bigger circle). A small white circle means the system thinks the pipette tip is at the current focus height.

“Show annotation” on/off	Turns off/on an overlay that shows the user’s injection target annotations. The overlay changes color depending on the annotations’ positions relative to the current focus plane. If the annotation is red, the annotation is below the current focus height, and if it is blue, then it is above the current focus height. The thickness of the annotation indicates the distance from the current focus height (thicker means farther from the current focus height). A thin white annotation means the annotation is at the current focus height.
“Show tissue” on/off	Turns off/on a semi-transparent magenta overlay that shows the detected tissue during the automatic annotation process. If the automatic annotation is working properly, the magenta overlay should coincide with the tissue. **This will only appear in the annotation window if conducting automatic annotations!**
“Show edges”	Turns off/on semi-transparent blue and yellow overlays that show the detected tissue edges during the automatic annotation process. If the automatic annotation is working properly, the blue overlay should correspond with the tissue’s apical edge and the yellow overlay should correspond with the tissue’s basal edge. **These will only appear in the annotation window if conducting automatic annotations!**

C.1.10System status widget

The system status widget is described in Table 13.

Table 13. Description of the system status widget.

Widget	**Description**
“System Status” box	Displays the status of the microinjection system and some important information like the number of injections that were completed.

C.2 Calibration window

Figure 26. Calibration window. This is where the user will conduct the calibration processes.

C.2.1 Pipette calibration widgets

The calibration widgets displayed in the calibration window are different than the calibration widgets in the main user interface window. The widgets in the Calibration window facilitate completion of the microscope-manipulator calibration process. These widgets are described in Table 14.

Table 14. Description of pipette calibration widgets for Calibration window.

Widget	**Description**
"Complete Calibration" button	Finishes the calibration process by computing the calibration with the datapoints generated during the process and returns to the main user interface window. The Autoinjector 2.0 will use the completed calibration during the microinjection process. This button should only be clicked after the calibration process is completed.
"Save and Complete Calibration" button	Does the same actions as the "Complete Calibration" button but also saves the calibration to a file. The calibration can be loaded later (even after closing and reopening the software). This button should only be clicked after the calibration process is completed.
"Exit" button	Returns to the main user interface window without computing the calibration, so the system will not be calibrated (unless it was already calibrated when the Calibration window was opened). This button can be clicked at any time.
"Run Auto-Calibration" button	Uses computer vision to autofocus the pipette tip and then move the manipulator to calibration positions while detecting the pipette tip. This is only available when "Automatic" mode is selected before opening the Calibration window.
Process guidance	This text guides the user through the steps on how to complete the calibration process. This text will change depending on the calibration mode that was selected in the main window ("Automatic", "Semi-Auto." or "Manual") and whether the user is conducting a new calibration or updating a calibration.

C.2.2 Microscope video stream, Zeiss control, and system status widgets

These widgets are described previously in sections **C.1.6**, **C.1.7**, and **C.1.10**.

C.3 Annotation Window

Figure 27. Annotation window. This is where the user will conduct the injection target annotation process.

C.3.1 Annotation control widgets

These widgets allow the user to coordinate the annotation process and provide guidance on how to complete injections. With these widgets, the user can decide how the annotation process should be conducted ("Automatic" versus "Manual"), and the user can complete or exit the annotation process. These widgets are described in Table 15.

Table 15. Description of annotation control widgets in Annotation window.

Widget	Description
"Annotation mode" dropdown	Sets how the types of annotations that can be completed • "Automatic": The user can manually draw annotations in the video display, but they can also use the "Automatic Annotation Control" widgets to automatically detect tissue and annotate the injection targets. • "Manual": The user can **only** manually draw annotations in the video display.
"Complete Annotation" button	Finishes the annotation process by keeping the drawn/detected annotations and returning to the main user interface window. The Autoinjector 2.0 will use the drawn/detected annotations during the microinjection process.
"Exit annotation" button	Removes any drawn/detected annotations and returns to the main user interface window. There will be no annotations to be injected.
Process guidance	This text guides the user through the steps on how to complete the annotation process. This text will change depending on the annotation mode that was selected.

C.3.2 Automatic annotation control widgets

These widgets facilitate automatic annotation of injection targets. The user can use these widgets to run single-shot automatic annotation on the current field-of-view. Additionally, the user can setup and run z-stacks to automatically annotate 3D volumes of tissue for injection. These widgets are described in Table 16.

Table 16. Description of automatic annotation control widgets in the Annotation window.

Widget	Description

“Auto-detect Edge” dropdown	Specifies the type of tissue edges that will be automatically annotated. • “Reachable”: Non-biased edge annotation that will annotate any edges that can be “reached” by the pipette. If calibration **is** completed, only edges that the pipette can touch without colliding with other items/tissue will be annotated. For example, if a piece of tissue is detected between the pipette and a detected edge, it will **not** annotate the edge because the pipette would collide with the tissue if it tried to inject that edge. If calibration **is not** completed, this will annotate all detected edges. • “Apical”: Automatic annotation will only annotate “reachable” edges that are classified as apical (their curvature is mostly concave). • “Basal”: Automatic annotation will only annotate “reachable” edges that are classified as basal (their curvature is mostly convex).
“Run Single Auto Annotation” button	Runs a single automatic annotation on the current field-of-view and annotates edges specified by the “Auto-detect Edge” dropdown.
“Set first focus” button	Sets the starting focus height for conducting automatic z-stack annotation.
“Set last focus” button	Sets the finishing focus height for conducting automatic z-stack annotation.
“Number of slices” button	Specifies number of slices/images to use for automatic z-stack annotation. The z-stack must include at least 2 slices (the starting and finishing focus heights).
“Run Z-Stack Auto Annotation” button	Runs a z-stack annotation by focusing to each slice in the z-stack and conducting an automatic annotation to annotate edges specified by the “Auto-detect Edge” dropdown.
“Stop Z-Stack” button	Stops the z-stack annotation process immediately.
“Z-Stack Progress” bar	Displays the progress of the z-stack process. The progress bar will increase with each z-stack slice and reach 100% when the z-stack is completed.

“Z-Stack Results”	Displays the number of slices that were successfully annotated during the z-stack annotation.

C.3.3 Microscope video stream, Zeiss control, display settings, and system status widgets

These widgets are described previously in sections **C.1.6**, **C.1.7**, **C.1.9**, and **C.1.10**.

Appendix D. Calibration mathematical details

A better formatted version of this is available in “calibraiton.pdf”. I wrote this to convert my handwritten notes to something more concrete (and get some practice with typing math), but I have not verified the accuracy of the computations. Even if there are some minor errors the gist should still be there.

v1.0: 2022 - 08 - 12

Note: I will use “position” and “coordinate(s)” interchangeably. Both instances refer to a set of values that define a spatial location relative to a coordinate system. For instance $(10,20)$ *is the position or coordinates of some point in 2D space.*

D.1 Introduction

With the Autoinjector, our goal is to robotically target locations in 3D space for microinjection. To do this, we need to accurately direct the micropipette to these 3D locations and deliver our injection payload. However, this is not a trivial task. When we look at tissue through the microscope and camera, we are visualizing the tissue in a coordinate system (CSYS) defined by the camera and the microscope focus position. For instance, the location of a cell in the coordinate system might be $x = 350$ pixels, $y = 900$ pixels, and $z = 1$ mm. But when we are targetting this cell for microinjection, we need to “tell” the micromanipulator where to go relative to its own coordinate system. The same cell with a position of $(350,900,1)$ in the microscope/camera CSYS might have a position of $x = 14000$ nm, $y = 9560$ nm, $z = 18775$ nm, and $d = 16990$ nm. In this case, d is the manipulator’s injection axis.

So now we have a single location in 3D space defined by two distinct coordinate systems: $p_{external} = (350,900,1)$ is the postion of the location relative to the external coordinate system (the microscope and camera), and $p_{manipulator} = (14000,9560,18775,16990)$ is the position of the location relative to the manipulator’s coordinate system. Now the question is, “Given the location of the micropipette tip as a manipulator postion what is its position relative to the external CSYS? Likewise, what is the manipulator position given an external position?”. We can answer these questions by computing a calibration.

The goal of calibration is to construct a map: a map that tells the robot what the position of the micropipette tip is in the external coordinate system when we know it the manipulator's position. We can then use the "inverse" of this map to tell the robot what its manipulator position should be to reach a location defined by an external position.

D.2 Calibration model

As mentioned we want to define a map, M, that transforms between an manipulator position at some time instance to an external position at the same time instance. Formally:

$$M: p_m(k+1) \rightarrow p_{ex}(k+1)$$

where p is the position, subscripts $_{ex}$ and $_m$ denote the external and manipulator coordinate systems, and $(k+1)$ defines the time instance. In the case of our robotic system where our microscope has a camera and motorized focus mechanism, the external position has 3 entries (i.e. $p_{ex} \in \mathbb{R}^3$). Conversely, our Sensapex micromanipulator has 4 axes (because it has diagonal injection axis), so the manipulator position has 4 entries (i.e. $p_m \in \mathbb{R}^4$).

$$p_{ex} = \begin{bmatrix} x_{ex} \\ y_{ex} \\ z_{ex} \end{bmatrix}$$

$$p_m = \begin{bmatrix} x_m \\ y_m \\ z_m \\ d_m \end{bmatrix}$$

We will assume a very simple discrete model for our robotic system. We assume that the external position of the micropipette tip at some time instance is the tip's previous external position plus the change in the tip's external position.

$$p_{ex}(k+1) = p_{ex}(k) + \big(p_{ex}(k+1) - p_{ex}(k)\big)$$

$$p_{ex}(k+1) = p_{ex}(k) + \Delta p_{ex}(k)$$

However, the system has no explicit knowledge of change in the tip's external position. For instance, the user may observe on the video display that the tip has moved from $x_{ex}(k) = 350$ pixels to $x_{ex}(k+1) = 600$ pixels, but this information isn't known to the computer. The computer has no measurement system to measure the tip position in the camera image (yet). But the system always has access to the manipulator's position, so we want to redefine $\Delta p_{ex}(k)$ in terms of the the change in manipulators positions, $\Delta p_m(k) = p_m(k+1) - p_m(k)$.

We will define a transformation, $\hat{T}$, that transforms between a change in manipulator position to a change in external position. Specifically:

$$\hat{T}: \Delta p_m(k) \rightarrow \Delta p_{ex}(k)$$

$$\Delta p_{ex}(k) = \hat{T} \times \Delta p_m(k)$$

$$\Delta p_{ex}(k) = \hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4} \times \Delta p_m(k)$$

We can assume that $\hat{T}$ is a composition of two separate transformations. The first transformation, $\hat{T}_{m_3 \leftarrow m_4}$, transforms the manipulator's displacement along its d axis to projected displacements along the manipulators x, y, and z axes. The second transformation, $\hat{T}_{ex \leftarrow m_3}$, transforms displacements along the manipulators x, y, and z axes to the external x, y, and z axes.

This composition allows an intutive explanation of how the manipulator's displacements are translated to external displacements. First we account for how moving the manipulator d axis is equivalent to moving its other axes. For instance, if we assume the d axis is coplanar with the manipulator's x - z plane, then displacing the manipulator d axis is equivalent to displacing the manipulator's x and z axes by some amount. After this, then we can account for how displacing the manipulator's x, y, and z axes relates to displacements in the external x, y, and z axes.

Therefore, we can update our initial equation for the external position as

$$p_{ex}(k+1) = p_{ex}(k) + \hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4} \times \Delta p_m(k)$$

When we want to know the external position of the micropipette tip, we simply compute the difference between our manipulator positions (which is possible because we always have access to its position), we multiply it by our transformation matrices, and finally add this to our intial external position. Simple! To do this though, we need to know the equation parameters: namely the initial external position, $p_{ex}(k)$, and the transformation matrices, $\hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4}$.

D.3 Defining model parameters

For our intial system, we are going to make some assumptions that will vastly simplify our parameter estimation for $p_{ex}(k)$, $\hat{T}_{ex \leftarrow m_3}$, and $\hat{T}_{m_3 \leftarrow m_4}$. These assumptions will simplify the structure of the transformation matrices and reduce the number of parameters we will

Assumptions

1. The manipulator's d- and y-axes are perpendicular
   - ($\Delta d_m \perp \Delta y_m$)
   - Explanation: This implies that some displacement in the manipulator's d-axis could be replicated by a combination of displacements in the x- and d-axes. Aka. Moving the injection axis "looks" like a combination of going forwards/backwards and up/down.
2. The manipulator's x-, y-, and z- axes are perpendicular to each other.
   - ($\Delta z_m \perp \Delta x_m$ and $\Delta z_m \perp \Delta y_m$ and $\Delta x_m \perp \Delta y_m$)

- Explanation: Maybe this assumption should actually be that none of the axes (not including d) are a linear combination of each other? This means that with moving just x, y, and z axes allows you to acheive any location in 3D space.

3. The angle between the manipulator's x-axis and d-axis is known. (Wtih assumptions (1) and (2), this also defines the angle between the manipulator's z-axis and d-axis)
   - $cos(\theta) = \frac{\Delta x_m}{\Delta d_m}$
   - $sin(\theta) = \frac{\Delta z_m}{\Delta d_m}$
   - Explanation: These equalities are predicated on perpendicuarity. If we modify (2) to be linearly independent (rather than perpendicular) then these equalties down hold. Anyways, this means that by knowing the angle, we can transform the d-axis displacement into an equivalant combination of x- and z-displacements.
4. The external x-, y-, and z- axes are all perpendicular to each other.
   - ($\Delta z_{ex} \perp \Delta x_{ex}$ and $\Delta z_{ex} \perp \Delta y_{ex}$ and $\Delta x_{ex} \perp \Delta y_{ex}$)
   - Explanation: The camera plane is square and is perpendicular to the focus axis.
5. The manipulator's z-axis is parallel to the external z-axis. (With assumption (4) this also implies that the manipulator x-y plane is parallel with the external x-y plane)
   - ($\Delta z_m \parallel \Delta z_{ex} \rightarrow \Delta z_{ex} = \alpha_z \Delta z_m$)
   - Explanation: The camera plane is square and is perpendicular to the focus axis.
6. The manipulator z-axis units are nanometers and the external z-axis units are micrometers.
   - ($\alpha_z = \pm \frac{1}{1000} \frac{\mu\text{m}}{\text{nm}}$)
   - Explanation: The scaling between the z-axes is already known (doesn't need to be determined).

D.3.1 Structure of matrices

With these assumptions we can now define the structure of our transformation matrices. We will uses the $'$ notation to define the pseudo-axis displacements (the projected axis displacements as a result of a d-axis change).

Recall that the $\hat{T}_{m_3 \leftarrow m_4}$ matrix transforms the manipulator's displacement along its d axis to projected displacements along the manipulators x, y, and z axes.

With assumptions (2) and (3) we have the feollowing equation for a displacement along the manipulator x-axis.

$$\Delta x_m{}' = 1 \times \Delta x_m + 0 \times \Delta y_m + 0 \times \Delta z_m + \cos\theta \times \Delta d_m$$

With assumptions (1) and (2) we have the following equation for a displacement along the manipulator y-axis.

$$\Delta y_m{}' = 0 \times \Delta x_m + 1 \times \Delta y_m + 0 \times \Delta z_m + 0 \times \Delta d_m$$

With assumptions (2) and (3) we have the feollowing equation for a displacement along the manipulator x-axis.

$$\Delta z_m{}' = 0 \times \Delta x_m + 0 \times \Delta y_m + 1 \times \Delta z_m + \sin\theta \times \Delta d_m$$

Putting these equations into matrix form, we define the structure of $\hat{T}_{m_3 \leftarrow m_4}$ matrix.

$$\begin{bmatrix} \Delta x_m{}' \\ \Delta y_m{}' \\ \Delta z_m{}' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cos(\theta) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sin(\theta) \end{bmatrix} \begin{bmatrix} \Delta x_m \\ \Delta y_m \\ \Delta z_m \\ \Delta d_m \end{bmatrix}$$

$$\hat{T}_{m_3 \leftarrow m_4} = \begin{bmatrix} 1 & 0 & 0 & \cos(\theta) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sin(\theta) \end{bmatrix}$$

Recall that the $\hat{T}_{ex \leftarrow m_3}$ matrix transforms displacements along the manipulators x, y, and z axes to the external x, y, and z axes.

With assumptions (2) and (3) we have the feollowing equation for a displacement along the manipulator x-axis.

$$\Delta x_{ex} = \alpha_{1,1} \times \Delta x_m{}' + \alpha_{1,2} \times \Delta y_m{}' + 0 \times \Delta z_m{}'$$

With assumptions (1) and (2) we have the following equation for a displacement along the manipulator y-axis.

$$\Delta y_{ex} = \alpha_{2,1} \times \Delta x_m{}' + \alpha_{2,2} \times \Delta y_m{}' + 0 \times \Delta z_m{}'$$

With assumptions (2) and (3) we have the feollowing equation for a displacement along the manipulator x-axis.

$$\Delta z_{ex} = 0 \times \Delta x_m{}' + 0 \times \Delta y_m{}' \pm \alpha_z \times \Delta z_m{}'$$

Putting these equations into matrix form, we define the structure of the $\hat{T}_{m_3 \leftarrow m_4}$ matrix.

$$\begin{bmatrix} \Delta x_{ex} \\ \Delta y_{ex} \\ \Delta z_{ex} \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 \\ 0 & 0 & \pm\alpha_z \end{bmatrix} \begin{bmatrix} \Delta x_m{}' \\ \Delta y_m{}' \\ \Delta z_m{}' \end{bmatrix}$$

$$\hat{T}_{ex \leftarrow m_3} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 \\ 0 & 0 & \pm\alpha_z \end{bmatrix}$$

D.3.2 Estimating model parameters

During the calibration process, the user “clicks” on the in focus pipette tip in the video display which causes the system to not the clicked pixel location, the focus height location, and the

manipulator location. This allows us to associate an external position (pixel location and focus height location) with the manipulator position. This gives us two matrices: P_{ex} which is a 'list' of external positions (p_{ex}) and P_m which is a 'list' of manipulator positions (p_m). With these matrices we are able to completely define all the parameters of our model.

$$P_{ex} = \begin{bmatrix} p_{ex,1}^T \\ p_{ex,2}^T \\ \vdots \\ p_{ex,n}^T \end{bmatrix} = \begin{bmatrix} x_{ex,1} & y_{ex,1} & z_{ex,1} \\ x_{ex,2} & y_{ex,2} & z_{ex,2} \\ & & \vdots \\ x_{ex,n} & y_{ex,n} & z_{ex,n} \end{bmatrix}$$

$$P_m = \begin{bmatrix} p_{m,1}^T \\ p_{m,2}^T \\ \vdots \\ p_{m,n}^T \end{bmatrix} = \begin{bmatrix} x_{m,1} & y_{m,1} & z_{m,1} & d_{m,1} \\ x_{m,2} & y_{m,2} & z_{m,2} & d_{m,2} \\ & & & \vdots \\ x_{m,n} & y_{m,n} & z_{m,n} & d_{m,n} \end{bmatrix}$$

D.3.2.1 $\hat{T}_{m_3 \leftarrow m_4}$ parameters

For $\hat{T}_{m_3 \leftarrow m_4}$, this matrix is already completely defined. Above, we showed the structure of this matrix and it can be seen that there is a single variable in this matrix (θ). But we said in assumption (3) that we know this angle θ, so no additional work is needed to define this matrix. The manipulator has a function that returns the angle, so all we need to do is 'ask' the manipulator.

D.3.2.2 $\hat{T}_{ex \leftarrow m_3}$ parameters

For $\hat{T}_{ex \leftarrow m_3}$, we have 5 variables shown above ($\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{2,1}$, $\alpha_{2,2}$, and α_z). We can eliminate the need to esitmate α_z because we assumed its value in assumptions (5) and (6). (We still need to define the directionality between the axes (+/-) but we leave that to be set by the user. This directionality can be saved and reloaded, so it doesn't need to be set every time.) This leaves four values to be determined ($\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{2,1}$, and $\alpha_{2,2}$). We will determine these values by a least squares regression.

For a stationary manipulator d-axis we have the following relationship (same as the equations above that defined the $\hat{T}_{ex \leftarrow m_3}$ matrix while omitting 0 entries):

$$\Delta x_{ex} = \alpha_{1,1} \times \Delta x_m{}' + \alpha_{1,2} \times \Delta y_m{}'$$

$$\Delta y_{ex} = \alpha_{2,1} \times \Delta x_m{}' + \alpha_{2,2} \times \Delta y_m{}'$$

If we have multiple instances in of changes in position (like we do in between entries in the P_{ex} and P_m matrices) then we can rewrite this as a matrix form

$$\begin{bmatrix} \Delta x_{ex,1} & \Delta y_{ex,1} \\ \Delta x_{ex,2} & \Delta y_{ex,2} \\ \vdots & \vdots \\ \Delta x_{ex,n} & \Delta y_{ex,n} \end{bmatrix} = \begin{bmatrix} \Delta x_{m,1} & \Delta y_{m,1} \\ \Delta x_{m,2} & \Delta y_{m,2} \\ \vdots & \vdots \\ \Delta x_{m,n} & \Delta y_{m,n} \end{bmatrix} \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} \\ \alpha_{1,2} & \alpha_{2,2} \end{bmatrix}$$

This series of equations can be solved via least squares for the α terms as long as you have two non singular changes in position (aka three distinct positions in P_{ex} and P_m tha don't lie on a single line)

D.3.2.3 $p_{ex}(k)$ parameters

We simply choose the last calibration position in P_{ex} to be the intial/reference position.

$$p_{ex}(k) = \begin{bmatrix} x_{ex,n} \\ y_{ex,n} \\ z_{ex,n} \end{bmatrix}$$

D.4 Putting it all together

We have a calibration model that defines where the location of the micropipette tip is in the external coordinate system when we know the tip's position in the manipulator coordinate system and we know where the tip started (the initial/reference position) external coordinate system and manipulator coordinate system. This calibration model states that the "new" tip position is the external coordinate system is the "old" tip positin in the external coordinate system plus its change in position (as transformed to the external coordinate system from the manipulator coordinate system).

$$p_{ex}(k+1) = p_{ex}(k) + \hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4} \times \Delta p_m(k)$$

We know the "old" position in the external coordinate system ($p_{ex}(k)$ is known), the "old" position in the manipulator coordinate system, and the "new" position in the manipulator system (so $\Delta p_m(k)$ is known)

We made some assumptions to define $\hat{T}_{m_3 \leftarrow m_4}$ in terms of the pipette angle, θ, which can be queried from the manipulator (so θ is known and therefore $\hat{T}_{m_3 \leftarrow m_4}$ is known).

$$\hat{T}_{m_3 \leftarrow m_4} = \begin{bmatrix} 1 & 0 & 0 & \cos(\theta) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sin(\theta) \end{bmatrix}$$

We made some assumptions, to define the $\hat{T}_{ex \leftarrow m_3}$ matrix interms of 5 parameters. We assumed we knew the z-scaling, and we can use least squares regrssion on a list of non-singular external and manipulator positions to estimate the rest.

$$\hat{T}_{ex \leftarrow m_3} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 \\ 0 & 0 & \pm\alpha_z \end{bmatrix}$$

Altogether we have the following equation by expanding the terms and multiplying the matrices:

$$\begin{bmatrix} x_{ex}(k+1) \\ y_{ex}(k+1) \\ z_{ex}(k+1) \end{bmatrix} = \begin{bmatrix} x_{ex}(k) \\ y_{ex}(k) \\ z_{ex}(k) \end{bmatrix} + \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 & \alpha_{1,1} \times cos(\theta) \\ \alpha_{2,1} & \alpha_{2,2} & 0 & \alpha_{2,1} \times cos(\theta) \\ 0 & 0 & \pm\alpha_z & \pm\alpha_z \times sin(\theta) \end{bmatrix} \left(\begin{bmatrix} x_m(k+1) \\ y_m(k+1) \\ z_m(k+1) \\ d_m(k+1) \end{bmatrix} - \begin{bmatrix} x_m(k) \\ y_m(k) \\ z_m(k) \\ d_m(k) \end{bmatrix} \right)$$

D.5 Going the inverse (external to manipulator position)

What if we want the manipulator position given a the external position? For instance, we click on a cell we want to target and get its external coordinates, but how do we move the manipulator to that position?

We can rearrange the equation above to get a useful relationship:

$$\Delta p_{ex}(k) = \hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4} \times \Delta p_m(k)$$

However, you can see above that $\hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4}$ is not square so it certainly can't be inverted to solve for $\Delta p_m(k)$. But we can do a little trick and say that we are going to hold one of the manipulator axes constant, so it can't move. Say that we aren't going to allow the injection axis to move so that $\Delta d_m = 0$. Then we can eliminate the "d" column in the $\hat{T}_{ex \leftarrow m_3} \times \hat{T}_{m_3 \leftarrow m_4}$ matrix, so that our our new equation looks like:

$$\Delta p_{ex}(k) = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 \\ 0 & 0 & \pm\alpha_z \end{bmatrix} \times \left(\begin{bmatrix} x_m(k+1) \\ y_m(k+1) \\ z_m(k+1) \end{bmatrix} - \begin{bmatrix} x_m(k) \\ y_m(k) \\ z_m(k) \end{bmatrix} \right)$$

Now we can invert the matrix (assuming the modified version isn't singular) to solve for our change in manipulator position (for $\Delta d_m = 0$).

Appendix E. Automatic calibration with pipette tip detection

Pipette tip detection GitHub repository is located at:

https://github.com/obria006/HT_yolov5_pipette_detection

Autoinjector 2.0 uses neural network pipette tip detection to automate the microscope-manipulator calibration procedure. By using the tip detection neural network, the system can automatically detect the pipette tip location in the camera and use these location coordinates when computing the calibration. Moreover, the tip detection neural network also classifies the tip as being in-focus or out-of-focus which can be used to autofocus the pipette tip at the beginning of the calibration procedure. The combination of autofocusing the tip and detecting the tip location means that the whole calibration procedure is fully automatic: the user doesn't need to

bring the tip fully into focus before calibration or click on the tip during the calibration process. Overall, this makes the calibration process faster and easier for the user.

E.1 YOLOv5 neural network for pipette tip detection

To enable neural network tip detection, Autoinjector 2.0 uses a YOLOv5 object detection neural network that was trained on images of the pipette. The training data consisted of the images containing the pipette and the annotated 3D location of the pipette in the image. The annotated 3D locations were comprised of the x- and y-pixel locations of the tip and the displacement of the tip from the in-focus plane in um. For example, an annotation of (500, 900, 23.3) means the tip is located at x=500pixel, y=900pixel, and z=23.3μm (or 23.3μm above the current focus height).

The Autoinjector 2.0 system's automation was leveraged to automatically generate the training data. A script was written (in the "Pipette Annotator" branch of the Autoinjector 2.0 GitHub) to send the pipette tip to random (x, y) locations (uniform random) in the camera FOV with random z-displacements (gaussian random) from the in-focus plane. When the manipulator reached the random position, the system took a picture and recorded the camera pixel coordinates and the z-displacement of the pipette tip (which it knew because microscope-manipulator calibration was completed beforehand). Additionally, the light intensity was randomly varied (gaussian random) to produce a more diverse training dataset. This automatic data generation was used for all combinations of objectives (10x, 20x, 40x) and optovars (1x, 1.4x, 1.6x). If I recall correctly, this training data resulted in 500 images for the (10x, 1x) and 100 images each for all other (objective, optovar) combinations. The (10x, 1x) was prioritized because this is the most used microinjection configuration at Human Technopole.

Because I was using YOLOv5 object detection for pipette tip detection[12], it expects training data in the form of bounding boxes around the desired object and a class for the object. Therefore I converted the annotated (x,y) pixel locations into a bounding box by padding uniformly around the centroid. I can't remember how large I made the bounding boxes around the pixel location centroid, but it should be in the GitHub code. Moreover, I had to convert the numerical z-

[12] For some time, I tried playing around with making a neural network regression network that could estimate the z-displacement of the pipette tip (rather than object detection that would classify the z-displacement). I made some progress with the ResNet-50 modified with a regression head, and it worked well on (10x, 1x) data but it didn't generalize well to the other magnifications (data not available). Its accuracy decreased for (10x, 1x) when training and inferring on all other magnifications. Ultimately, I was running out of time, so I had to revert to using the YOLOv5 with z-displacement classification rather than regression.

displacement to a class which I with the following class schedule {z<-8um: below focus, -8<z<8: in-focus, z>8: above focus} (I think this is the cutoff, should also be in the GitHub). Likewise, the output of the YOLOv5 neural network is a bounding box and a z-displacement classification. The system takes the centroid of the detected bounding box as the (x,y) location of the pipette tip, and it uses the classification (above-, below-, or in-focus) as the z location of the pipette tip.

The tip localization results are shown in Figure 28[13]. As shown, YOLOv5 accurately classifies the z-position and accurately localizes the (x,y) position. When the neural network predicts the pipette tip as being below-, in-, or above-focus this mirrors the actual z-displacement relative to the current focus plane. Also, the approximate 2um average XY prediction error is sufficient for our application especially when the pixels at (10x, 1x) are 1.1-1.3 um large.

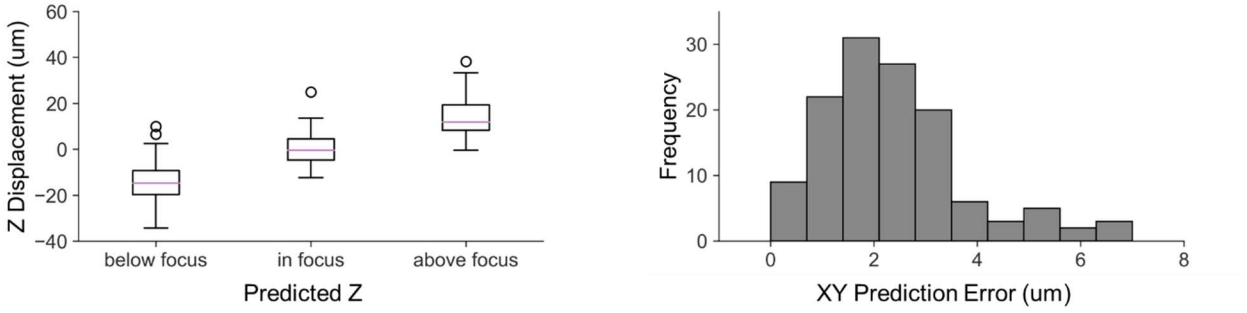


Figure 28. YOLOv5 pipette tip detection accuracy evaluated on test data including images taken on all objectives and optovars combinations.

E.2 Implementation of pipette tip detection for automatic calibration

The implementation of the tip detection neural network for automatic calibration is shown in Figure 29[14]. When the user clicks “Run Auto-Calibration”, the system starts with autofocusing the pipette. This is accomplished by detecting the pipette tip and assessing the classification of the z-displacement. If the tip is classified as above-focus, then it is commanded to move down, and if it is classified as below below-focus then it is commanded to move up. It is not shown in the schematic, but if it starts flip-flopping between above- and below-focus it will reduce the distance that the tip is moved so that the tip starts coming into focus. After it is autofocused then the pipette is moved to pre-computed positions. At these positions, the pipette tip location is detected and stored. When the pipette has gone to all precomputed positions, then the calibration is computed using the detected tip locations.

[13] Taken from my November group meeting presentation.
[14] Taken from my November group meeting presentation.

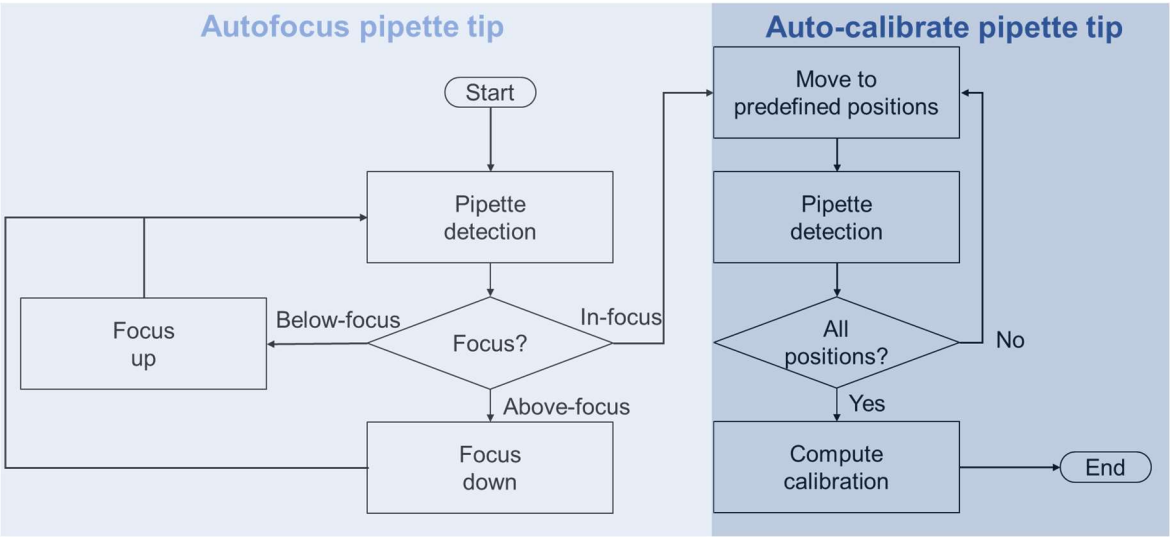


Figure 29. Pipette detection for auto focusing and autocalibration process.

Appendix F. Automatic annotation with tissue detection

Tissue detection GitHub repository is located at: https://github.com/obria006/unet_segmentation

Autoinjector 2.0 uses neural network tissue segmentation[15] to automate the annotation procedure. With the tissue segmentation neural network, the system can automatically classify all pixels in an image as belonging to the tissue or to the background (this pixelwise classification is called segmentation). The result of this segmentation is an image region that the system thinks is the tissue and an image region that the system thinks is the background. The Autoinjector system can extract the edges/boundaries of the tissue region as potential candidates for microinjection. Further post-processing with computer vision can analyze these edges shapes to determine if they are apical or basal edges of the tissue. Finally, these edges can be used as annotations to automatically complete the target annotation process. This makes the annotation process faster because the user doesn't need to manually draw each annotation along the edge of the tissue. It also makes the annotation process potentially more consistent with the tissue since the user cannot exactly trace the edge of the tissue.

F.1U-Net neural network for tissue segmentation

For automatic neural network tissue segmentation, Autoinjector 2.0 uses a U-Net neural network that was trained on images of embryonic mouse tissue. The training data consisted of the image of the embryonic mouse tissue and a separate, ground-truth labeled image where

[15] In the main text of the document, I use the word tissue "detection". Here in the appendix, I'm using the word tissue "segmentation" as it is more technically accurate description the computer vision process.

each pixel is labeled as being background (0) or tissue (1). This ground-truth labeled image was semi-automatically annotated using MATLAB's Image Labeler. In the image labeler, I use the "Superpixel Tool" to divide the image into distinct regions and then specify which regions were tissue or background. The resulting training image is a binary labeled image with pixel intensities indicating whether that pixel was background or tissue.

The tissue images consisted of a piece of embryonic mouse tissue at multiple different focus heights. Since the tissue is uneven, these multiple focus heights mean that the different parts of the tissue are in-focus in images. To generate the training images, I took 3 different images of each piece of tissue corresponding to 3 different focus heights: one image where most of the tissue is in-focus and the other two images are above and below this "in-focus" focus height. When splitting these images for training, testing, and validation, I kept images of the same piece of tissue in the same dataset. For example, all the images of "tissue piece 1" were put in the training dataset while all images of "tissue piece 2" were put in the testing dataset. I separated these tissue images by dataset because even though each tissue image is unique, the "above focus" image of a piece of tissue looks like the "in-focus" image of that piece of tissue. Hence, I didn't want to include images in the testing and validation datasets that are similar to images that neural network was trained on. This is so I can get a more accurate sense of the neural network's ability to generalize to other images when I evaluate the testing/validation data. In total there were approximately 120 images across 40 pieces of tissue that were used for training/testing/validation.

After the neural network makes the tissue detection, I do some minor post-processing of the detected regions. First, I eliminate any detected tissue or background regions whose sizes are in the 5th percentile of the respective tissue or background region sizes in the training dataset. For example, if the tissue detection results in a small detected "background" whose sizes is small (like if it is surrounded by detected "tissue"), then this is converted to tissue. Moreover, I do median filtering on the image to smooth out the detected tissue edges.

The tissue segmentation results are shown in Figure 30 and Figure 31[16]. Despite some minor segmentation errors shown in the bottom row of images in Figure 30, the segmentation neural network accurately labels the tissue. The segmentation metrics in Figure 31 all approach 1 which is optimal.

[16] Figures taken from September group meeting presentation.

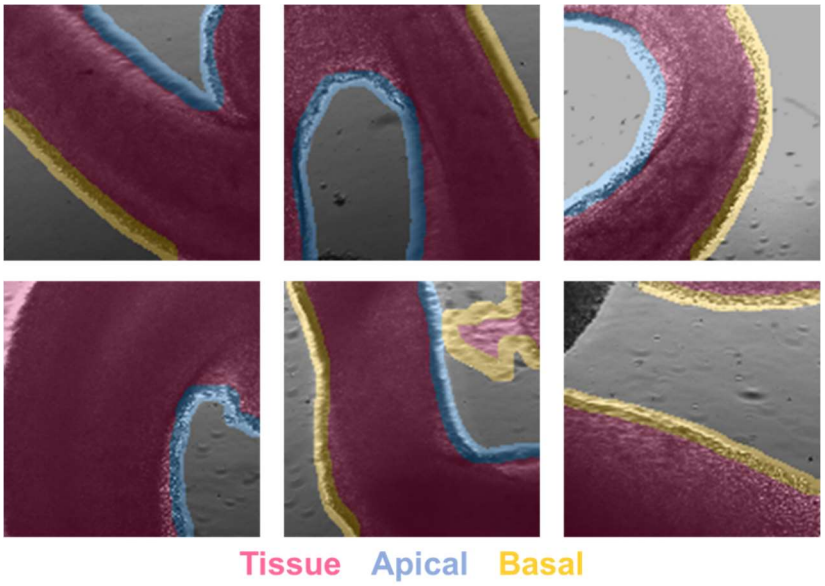


Figure 30. Qualitative tissue segmentation results. The top 3 images are accurate segmentations. The bottom 3 images have some segmentation errors resulting from the post-processing (the left and right images mis-classify the small background and tissue regions) and the neural network detection itself (neural network thinks the textured petri dish is tissue).

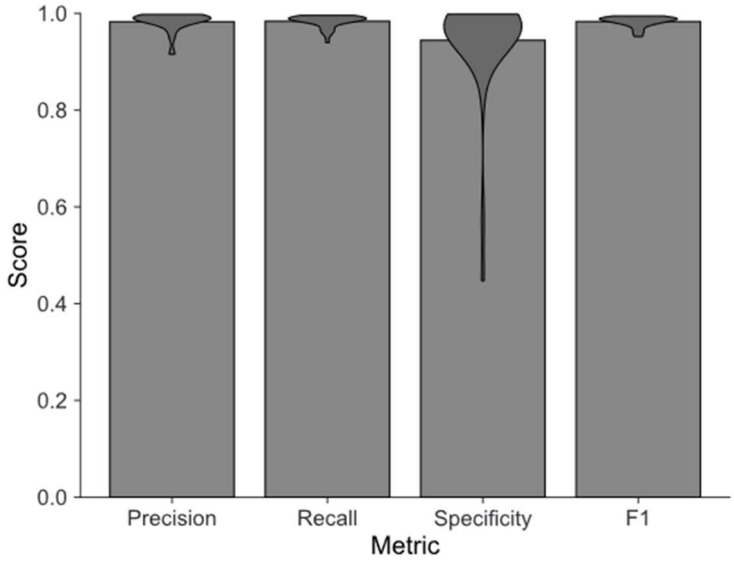


Figure 31. Quantitative segmentation results on test data.

F.2Computer vision shape analysis for edge classification

Different cell populations exist in at different locations within the embryonic mouse tissue (and organoids). Apical progenitors are located near the tissue's apical surface while neurons are located near the tissues basal surface. Thus, it is important to automatically classify the edge as

apical/basal, so the Autoinjector 2.0 correctly targets the user's desired cell population. In embryonic mouse tissue (and in organoids), the apical surface has concave curvature (tissue outside the "cave") while the basal surface (tissue inside the "cave") as shown in the top three pictures in Figure 30.

Autoinjector uses this stereotyped morphology of apical=concave and basal=convex to classify the tissue edges as apical or basal. After tissue segmentation, the system extracts the edges/boundaries of the tissue region. Then it performs a shape analysis on each edge to determine if it is apical or basal. This shape analysis works by creating a "convex hull" around the detected edge and determining the area of intersection of that convex hull with the detected tissue. If the ratio of areas (intersection area/convex hull area) is greater than 0.5 then it is classified convex (or basal) otherwise its concave (or apical)[17]. The computer vision algorithm is slightly more complex than this because it also accounts for edges belonging to the same piece of tissue. If a piece of tissue has two detected edges, then one edge is likely basal and one edge is likely apical. In this case, the edge with the larger area ratio is considered basal and the other edge is apical.

Edge classification results are shown above in Figure 30 and below in Figure 32[18]. Like tissue segmentation, the edge classification is accurate at correctly identifying edges.

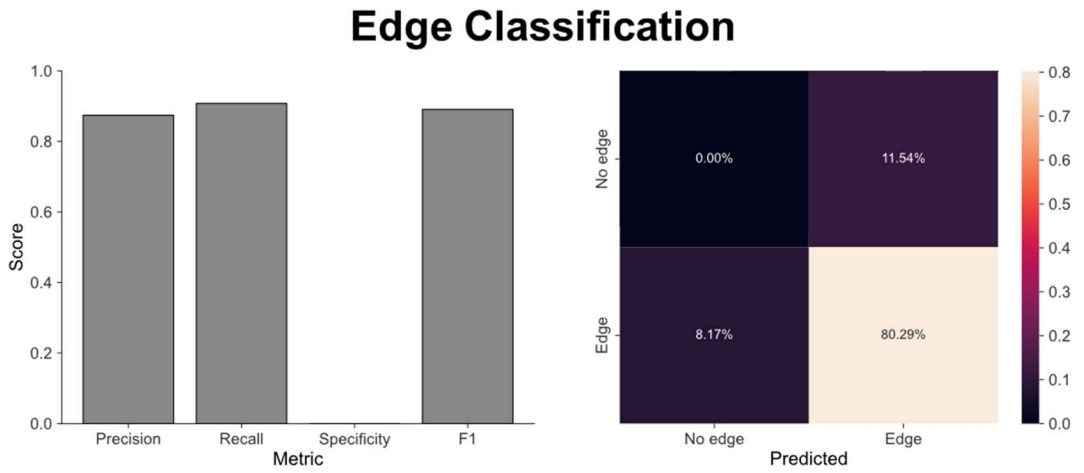


Figure 32. Quantitative edge classification results showing classification metrics and confusion matrix.

[17] I'm sorry there is no figure for this but I'm writing this on my last day…
[18] From September group meeting presentation.

F.3Implementation for automatic annotations

When the user conducts automatic annotation (whether with the single annotation or the z-stack), the system takes the current image and passes it into the U-Net neural network and edge classifier to get the edge annotations as shown Figure 33. Once the Autoinjector 2.0 receives the edge annotations it does some further post-processing to refine the annotations. This post-processing includes identifying if the pipette tip can “reach” the detected edges or if there is detected tissue in the way (**3.2.3** Reachable vs Basal vs Apical edge detection). It also trims edges so that they don’t go to the edge of the FOV and ignores very short edges.

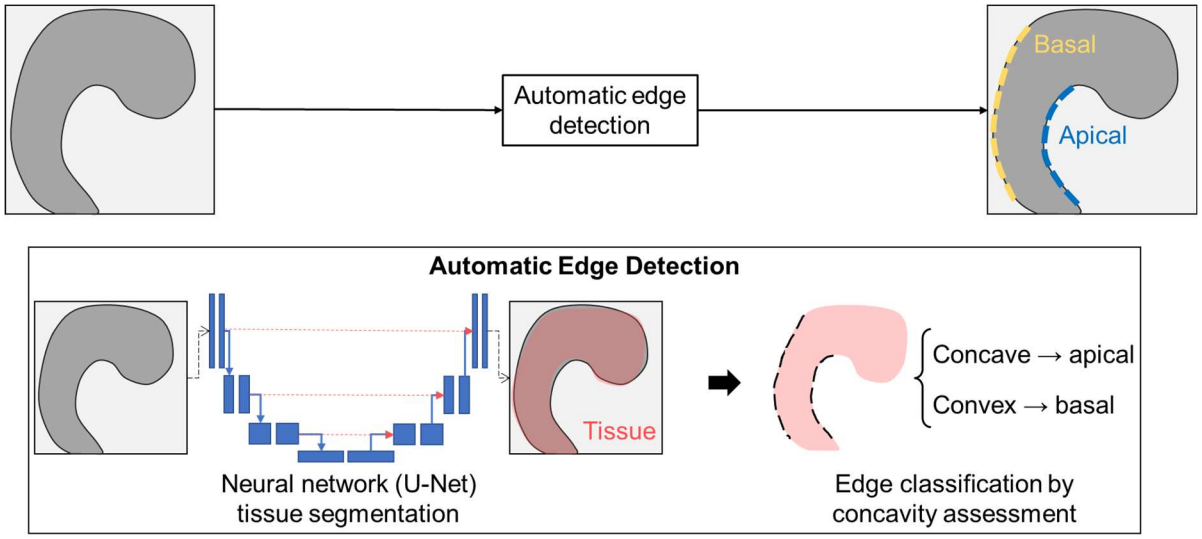


Figure 33. Schematic for automatic annotation with neural network and edge classification.

Appendix G. Annotation tracking to compensate for tissue movement

During microinjections the tissue may move as it is being penetrated by the pipette. This can cause the microinjection accuracy to decrease because the tissue will not be aligned with the drawn annotation, so microinjections will miss the tissue (because all microinjection positions are relative to the drawn annotation). Autoinjector 2.0 implements annotation tracking to track the tissue movement and update the annotation positions to align with the displaced tissue. This enables the Autoinjector 2.0 system to automatically compensate for tissue displacement and facilitate accurate microinjections without the user needing to manually compensate for the tissue movement.

G.1 Optical flow and Kabsch algorithm for multiple annotation tracking

Autoinjector 2.0 uses the optical flow algorithm to track tissue displacement[19]. It employs the OpenCV-Python implementation of the optical flow algorithm solved with Lukas-Kanade method to conduct single tissue tracking for a single annotation. During microinjections, the Autoinjector 2.0 first focuses on the annotation before moving the pipette to inject the tissue. When the Autoinjector 2.0 focusses to the tissue, the system generates a list of pixel coordinates on the annotation that are uniformly separated by 15 pixels (or it generates a list of pixel coordinates of 10 evenly spaced pixels if the annotation is short). This list of pixel coordinates is used as the initial set of pixels for optical flow tracking. Because it uses annotations to generate the set of pixels for tracking, this tracking algorithm works for both manually drawn and automatically drawn annotations. Once the tracker is initialized, then the system starts injecting the annotation.

After the tracker is initialized and the system is injecting, the system calculates the optical flow with every new frame available from the camera. This updates the position of each of the tracked pixels according to the tissue movement and governed by the optical flow equations. The optical flow tracker works best when the area around the annotations is highly textured because of the way the optical flow algorithm works: if the pixel intensity of the pixel that it is tracking is similar to the surrounding pixels, it will be harder to track the pixel as it moves because it looks like everything around it.

The optical flow algorithm only tracks the set of pixels corresponding to a single annotation at any given time. This is because other annotations may be out-of-focus, so that annotation's set of tracking pixels will also be out-of-focus making them un-trackable. However, we still want to update the positions of non-tracked annotations according to the movement of the tissue for the annotation that is currently being tracked. For this, we assume the tissue is rigid such that the ways in which current annotation is being displaced applies to all annotations. This is a valid assumption because this tissue behaves relatively rigidly: as the pipette pokes/prods the tissue, it mostly just translates and rotates without being stretched or shrunk.

With this rigid body assumption, it simplifies how we can compute tissue displacements. During tracking of the tissue for the current annotation, the system compares the current position of the tracked set of pixels to the initial positions of those pixels. These current and initial positions can

[19] I'm sorry I don't have any figures for this. I'm writing this on my last day and I developed the annotation tracking feature in my last 3ish weeks at Human Technopole.

then be used in the Kabsch algorithm to compute the rigid body rotation and translation of the annotation. These rotations and translations are then applied to all annotations even if they are not currently being tracked.

Once the system finishes tissue tracking for the current annotation, it focusses on the next annotation (which has been updated by the computed rotations and translations). The system reinitializes the tracker on this annotation's updated set of pixels and continues the tracking process. This process is repeated all of the drawn annotations.

I unfortunately don't have any quantitative data on the tracking performance because I developed this feature in my last 3 weeks at Human Technopole. Qualitatively, I can say this feature works best for long annotations in tissue that is highly textured. The long annotations means that there are lots of tracking points, so even if the system inaccurately tracks a couple pixels, on average it is still accurately tracking most pixels. And as mentioned before, having textured tissue makes it easier to track the pixels because the tracked pixels look different than their neighbors.

Appendix H. Pressure Controller Details

One of the main limitations of microinjection is the throughput limitations dictated by needing to change microinjection pipettes. This limitation is exacerbated when the microinjection pipette clogs with aggregated injection solution or miscellaneous debris which requires frequent changing of the pipettes. Autoinjector 2.0 aims to mitigate this issue by adding an "unclogging" feature that sends a high-pressure pulse to the microinjection pipette to dislodge the clog.

The Autoinjector 2.0 pressure controller is a modified version of the pressure controller from Autoinjector 1.0 to include functionality for high-pressure unclogging. In Autoinjector 1.0, the air supply passes through a mechanical regulator and an electronic regulator to reduce the pressure from air supply pressures (30+ psi) to injection pressures (0-2psi). Then, the air passes through a solenoid valve that acts as a gate to turn off/on pressure at the pipette tip. With Autoinjector, the highest achievable pressure at the pipette tip was limited to 2psi by the electronic regulator which is insufficient to unclog the pipette. Autoinjector 2.0 permits high-pressure unclogging by allowing an alternate pathway for the high-pressure air supply to bypass the regulator and pressure controller. This alternate high-pressure air pathway is gated by two solenoid valves which allows the user to select between 0-2psi for injections or 30psi for unclogging. A simplified schematic comparing pressure control for Autoinjector 1.0 and

Autoinjector 2.0 is shown in Figure 34. The wiring and pneumatic diagrams for the Autoinjector 2.0 pressure controller are shown in Figure 35 and Figure 36.

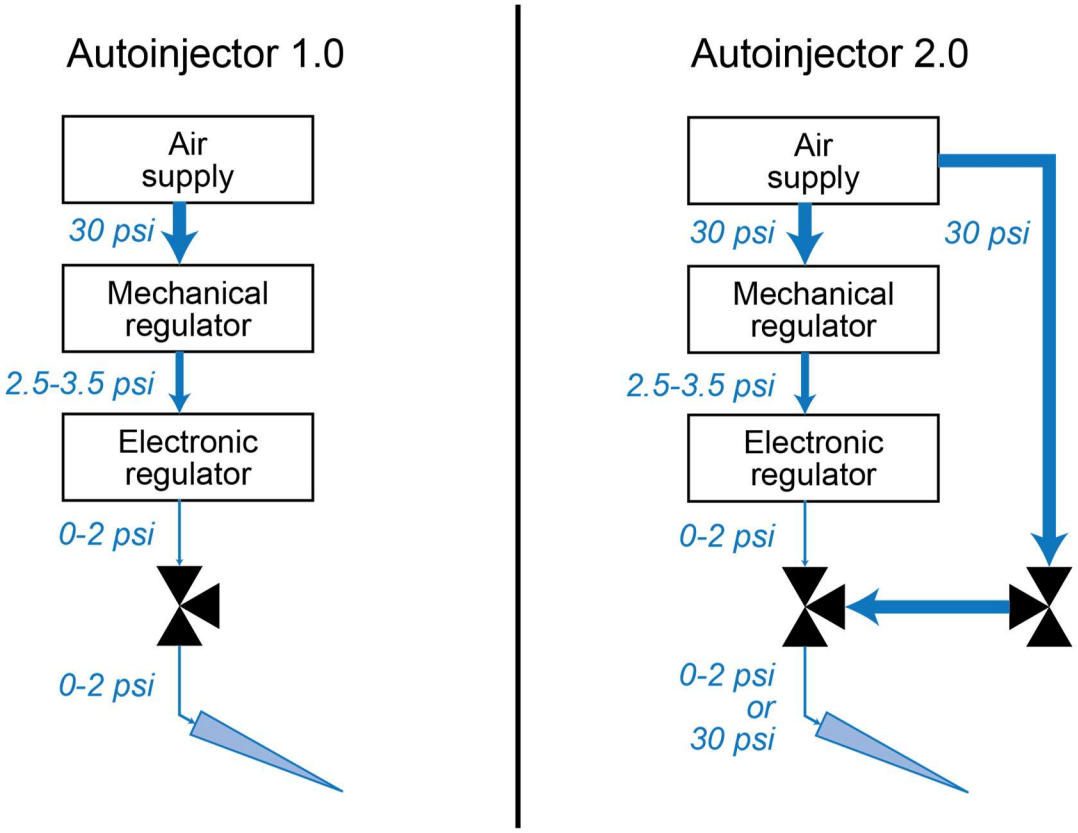


Figure 34. Pressure control scheme comparison between Autoinjector 1.0 and 2.0. Autoinjector 2.0 permits high-pressure unclogging by including a high-pressure pathway that bypasses the regulator and controller before being gated by solenoid valves.

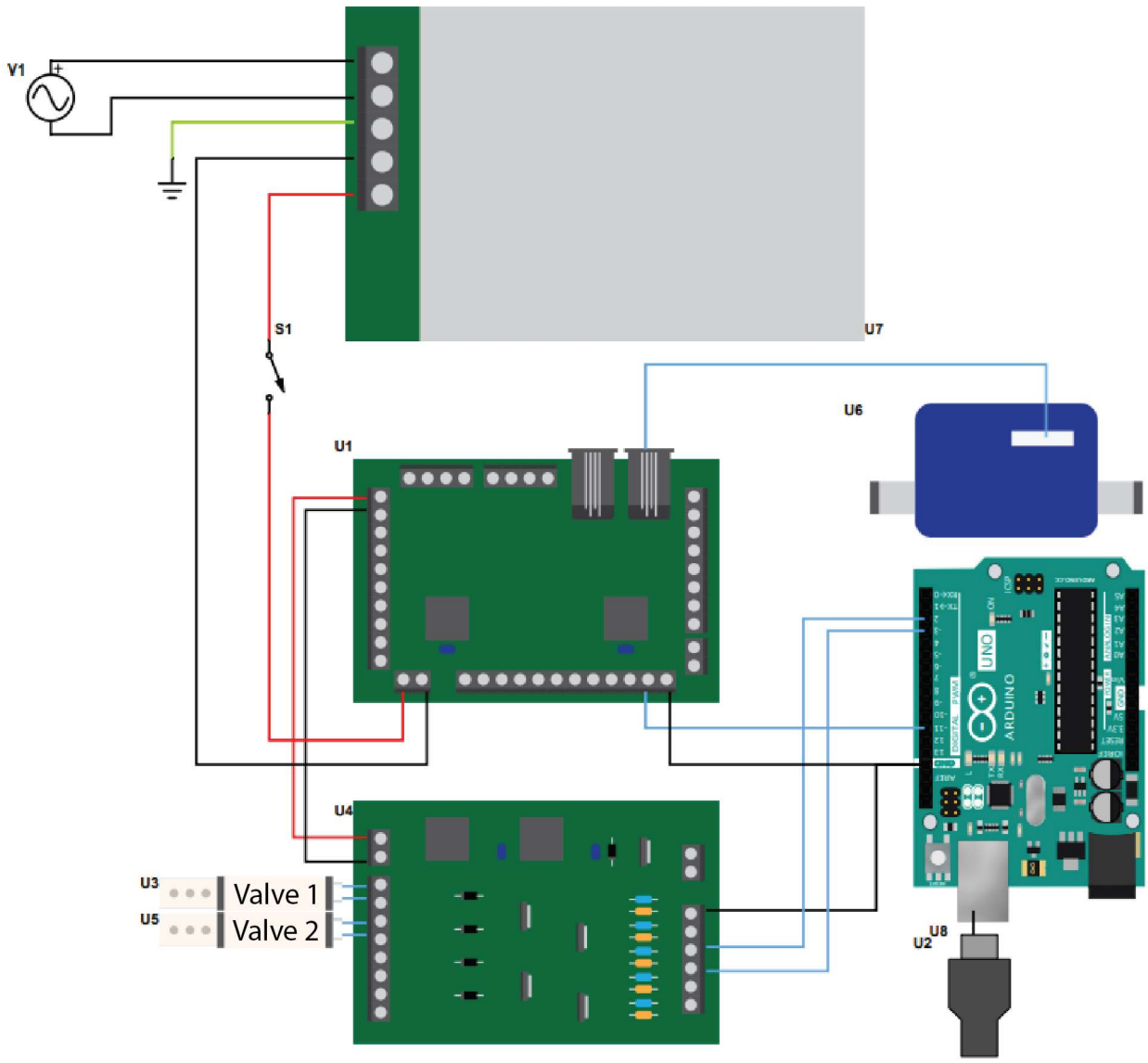


Figure 35. Wiring schematic for Autoinjector 2.0 pressure control system.

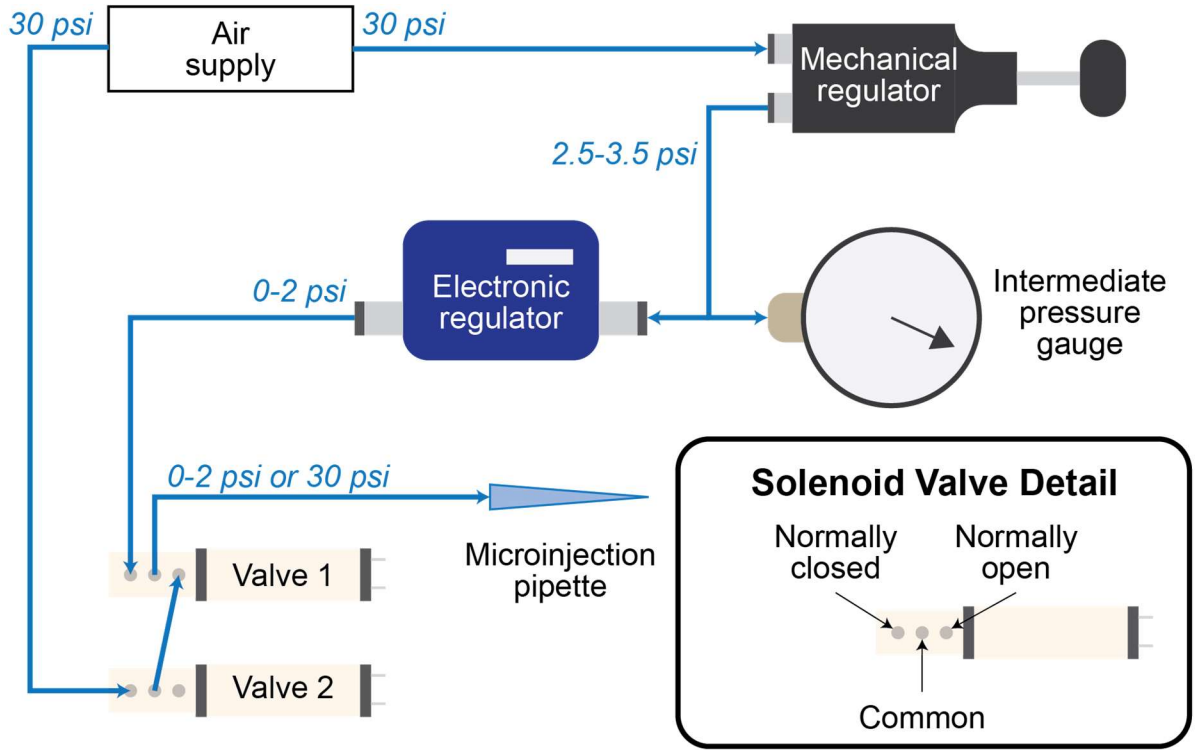


Figure 36. Pneumatic diagram for Autoinjector 2.0 pressure control system.

Appendix I. Potential future improvements

Below I've detailed some aspects that I would have liked to improve to make the system more robust and have a greater feature set.

- Change U-Net segmentation network to be 3D. Current U-Net is 2D (single image), but U-Nets can be 3D. This may be a better option when doing z-stack annotations rather than doing a stack of 2D annotations. In the 2D annotations, there can be sharp discontinuities in the segmentation along the z-axis which maybe wouldn't happen with 3D segmentation.
- Penalize sharp discontinuities/non-smooth boundaries for tissue. The tissue shape is generally smooth and does not have sharp corners (typically). Potentially you could modify the U-Net segmentation loss function to penalize these boundaries.
- U-Net to automatically segment edge rather than computer vision shape analysis. You could modify the segmentation network to automatically segment and classify the edge boundaries as "apical" and "basal" rather than inferring them from their shape with the computer vision algorithm (as discussed in **F.2**).

- Pipette detection to be robust for different imaging modalities (DIC, Ph1, etc). Pipette detection was trained in a single imaging mode (transmitted light), so it likely won't work well with DIC or phase contrast. Generating training data for these modalities and training on them could improve performance when injecting in these different modalities.
- Pipette detection to be refined to give quantitative estimate of the z-position. Currently the system uses the classification (above-, below-, in-focus) as the z-coordinate of the pipette. But you could change models to a regression model that specifically predicts the z-location as in Machine Learning-Based Pipette Positional Correction for Automatic Patch Clamp In Vitro. This would simplify (and maybe make more accurate) the autofocusing algorithm.
- Improve the object tracking with a Kalman Filter. You could use the optical flow algorithm as the measurement system and assume a dynamic system with a stationary tissue position (the dynamical matrix would be identity: aka all pixels retain their original position). Also maintain the same rigid body tissue assumption (or maybe a slightly plastic one with small deformations). Then you would Kalman filter the optical flow positions with the dynamical information to generate an enhanced position estimate?
- Additionally, you could use tissue detection to update the ground truth annotations. Rather than assuming the updated rotated/translated annotation is the new position, you could run a U-Net segmentation to generate an updated position of the tissue for tracking. Additionally, you could feed these updated rotated/translated coordinates into the U-Net (as a prior potentially but I don't know if this is possible) to selectively prioritize segmentation in that region where the updated annotation was shifted to.
- Z-calibration function to be able to account for different z-scaling. Right now, the z-scaling is specific to the specific immersion medias of the objective and tissue (according to Abbe's sine condition and refractive index mismatch – idk about this but it was information from the microscope/optics specialist). This scaling is subject to change if you change the media of immersion, so z-calibration could be used to automatically compensate for this.