MACHINE LEARNING AND SIMULATION FOR HIGH-THROUGHPUT SINGLE-CELL MEASUREMENT

A Dissertation
Presented to
The Academic Faculty

By

Nathan Malta

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Computing
School of Computer Science

Georgia Institute of Technology

December 2023

MACHINE LEARNING AND SIMULATION FOR HIGH-THROUGHPUT SINGLE-CELL MEASUREMENT

Thesis committee:

Dr. Craig Forest
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Danfei Xu
School of Interactive Computing
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Date approved: December 1, 2023

If in physics there's something you don't understand you can always hide behind the uncharted depths of nature. You can always blame God. You didn't make it so complex yourself. But if your program doesn't work, there is no one to hide behind. You cannot hide behind an obstinate nature. A zero is a zero, a one is a one.

If it doesn't work, you've messed up.

Edsger W. Dijkstra

To everyone who builds and maintains open source software,

especially under the MIT and Apache Licenses

ACKNOWLEDGMENTS

I am incredibly grateful to my advisor, Dr. Craig Forest, for his invitation into the lab and constant support of all my work. His feedback and encouragement was instrumental in building this thesis. I owe an immense debt of gratitude to my committee members, Dr. Danfei Xu and Dr. Seth Hutchinson, for their scholarly input, meticulous feedback, and unwavering commitment to pushing me towards excellence. I also want to express my appreciation to my colleagues in the lab for their valuable insights and continuous motivation in my pursuits.

I would also like to thank my parents, whose support and sacrifices paved the way for my academic endeavors. Finally, I want to express my sincere thanks to my friends for their unwavering support, encouragement, and occasional study breaks, and for supplying me with laughter and much-needed moments of relaxation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The following work concerns increasing throughput and reducing manual labor required when conducting biological experiments. As a prerequisite for a wide range of experiments in biology, cells must be cultured. During culturing, important parameters regarding the health of these cells must be either approximated by eye or manually counted in microscope images. In the first chapter of this work, I introduce a novel approach to detecting cells in microscope images and extracting quantitative, biologically relevant parameters to aid researchers.

The remainder of this work considers patch clamping, an important technique for characterizing the electrical changes of individual, living cells. In this technique, a high-precision robotic manipulator is driven inside of a cell, so that electrical recordings can be taken. This technique is currently being used to better understand a wide array of ailments, including Alzheimer's disease and macular degeneration. Unfortunately, the technique has been plagued by low-throughput. To improve automation and throughput I create new high-accuracy calibration routines and an automatic patch clamping protocol.

Finally, I attempt to tackle the high barrier to entry for learning about patch clamping. A patch clamping rig cost many tens or hundreds of thousands of dollars due to the specialized equipment needed. This severely limits the number of people familiar with the technique. To address this, I create a software simulation, a video game, that walks a user through the highlights of the technique. This software is open source and free to download. Preliminary user trials suggest that users develop a better understanding of the technique in just minutes of playing.

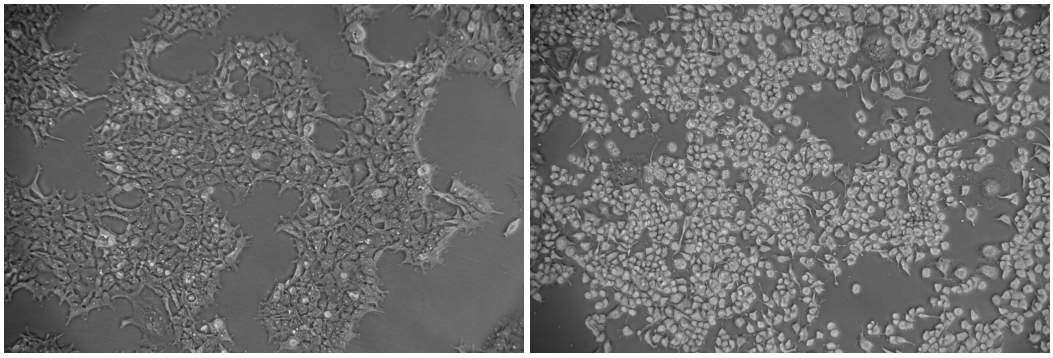
CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 An Introduction to Single-Cell Experiments

Single cell experimentation is an interesting and powerful area of biology. The ability to observe characteristics of individual cells has become an important tool for researchers studying a wide array of conditions. For example, by assessing changes in voltage potential of a few neurons individually, firing patterns and connectivity can be deduced. Recent efforts use this idea to study the progression Alzheimer's disease in mice [1]. Further, by analyzing and modeling the electrical properties of the eye, researchers can gain a deeper understanding of conditions like age-related macular degeneration [2].

It is important to consider the size of cells under experimentation. Bacteria, for example, are considerably smaller than their eukaryotic counterparts (0.5 to 2 micrometers in diameter) and sometimes enter experiments as unintentional contaminants [3]. The remainder of this work concerns mammalian cells which are much larger, generally around 20 micrometers in diameter, or about the width of a human hair. In particular, our experiments consider so-called "immortalized cell lines" or cells that have been modified in a lab to divide indefinitely, similar to tumorous cells, and unlike healthy mammalian cells found in nature [3]. These cells are generally grown floating in a solution ("suspended cells"), after which they can they can be stuck to a thin piece of glass known as a cover slip ("adherent cells") for viewing under a microscope or for further experimentation [3]. In particular, our lab cultures the Human Embryonic Kidney 293 (HEK 293) and Neuro-2a (N2a) cell lines. Although these cell lines are both derived from mammalian cells, as seen in Figure 1.1, these cells have very different shapes and patterns of growth.

When culturing cells, it is important to assess various parameters to determine cell



(a) HEK 293 (b) N2a

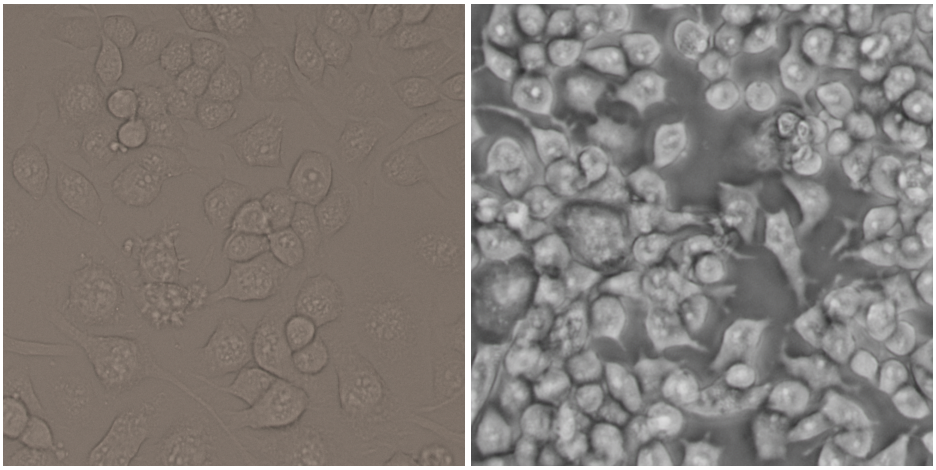
Figure 1.1: Images of the two main cell lines used in this work.

health, such as the density of cell growth ("confluency"), the shape of cells ("cell morphology"), and average number of adjacent cells. As seen in Figure 1.1, these can easily number in the hundreds per image. This can make a simple task like counting the number of cells incredibly time intensive. This is a ripe area for automation.

1.2 An Introduction to Microscopy

Microscopes are important tools for studying living organisms. A few common varieties are in use, each with a set of strengths and weaknesses. In our lab, the two main varieties in use are known as Brightfield and Phase Contrast microscopes. In Brightfield microscopes, light passes up through a thin sample and is enlarged by an objective lens [4]. Brightfield microscopy is very simple, although images can appear flat with lower contrast between features, as shown in figure Figure 1.2a.

Alternatively, in Phase Contrast microscopes, light is strategically blocked and refracted to create a sense of depth in the sample [4]. This makes images taken under Phase Contrast have a more 3-dimensional appearance and greater contrast between features, when compared to the flatter images produced by Brightfield. This is seen in Figure 1.2b. Unfortunately, samples imaged with Phase Contrast can sometimes have a "halo effect" where light from a bright object bleeds over the edge slightly towards its darker surroundings. [5]. This effect subtle but present in Figure 1.2b.



(a) Brightfield (b) Phase Contrast

Figure 1.2: Crops from microscope images of the same cell type (N2a) taken under (a) Brightfield and (b) Phase Contrast microscopy.

1.3 Patch Clamping

Patch clamping is a technique that allows experimenters to observe the electrical properties of individual cells. This technique has a wide array of use cases, from assessing the connectivity of neurons in brain tissue [1] to studying the progression of macular degeneration [2]. At a high level, patch clamping measures voltage changes across the cell membrane by placing an electrode inside the cell and recording changes in voltage potential between this electrode and a wire placed in the surrounding solution.

Initially, the only limiting factor for electrically charged ion flow is the small opening at the tip of the micropipette as seen in Figure 1.3a. Depending on pipette sharpness, at this stage, the electrical resistance between the internal electrode and the external solution is 4 to 8 Megaohm. A small amount of positive pressure is used here to prevent any debris in the external solution from clogging the pipette. Next, using a high-precision manipulator and microscope image, an experimenter will guide the pipette such that it is just adjacent to the cell as seen in Figure 1.3b. At this stage a small bump in resistance will be seen, as part of the pipette tip is blocked by the cell, reducing ion flow.

Subsequently, slight negative pressure is exposed to the cell, pulling a section of the

cell membrane within the pipette and forming a seal, as in Figure 1.3c. Because the pipette is almost entirely blocked at this stage, the resistance spikes to over a Gigaohm. Hence, this stage is called a "Gigaseal". Finally, a few bursts of negative pressure are applied to rupture the cell membrane as displayed in Figure 1.3d. At this stage, we now have an electrode inside the cell, and can thus measure the cell's electrical response to different stimuli.

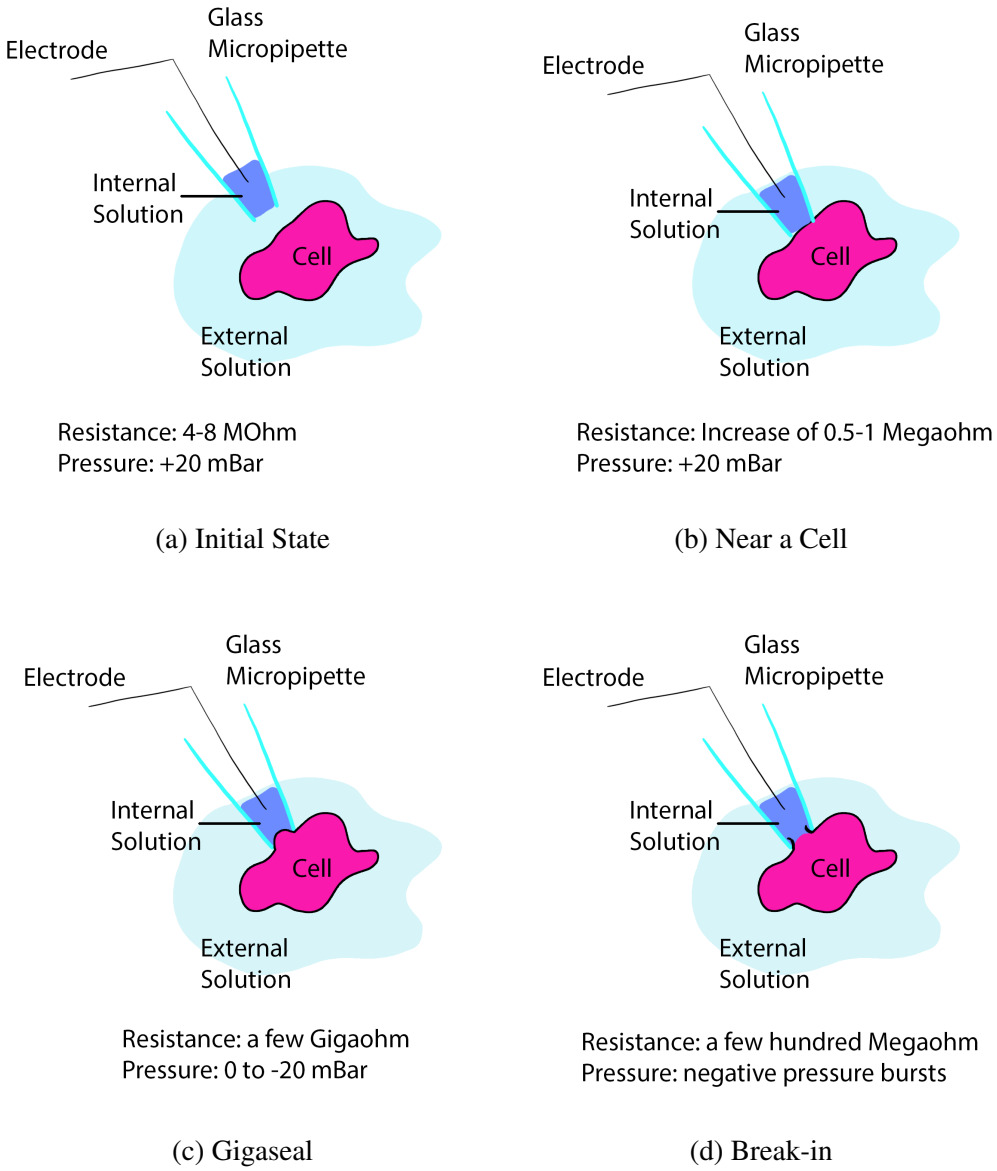


Figure 1.3: A diagram showing the various stages of patch clamping a cell.

CHAPTER 2
SAMCELL: A NOVEL APPROACH FOR GENERALIZED CELL SEGMENTATION

2.1 Preface

Before conducting patch clamping experiments, a supply of cells is required, generally obtained through cell culturing. As mentioned in chapter 1, it is necessary, albeit laborious, to obtain certain metrics like growth density from microscope images. In this chapter, I attempt to tackle this problem by producing a method of automatically obtaining biologically relevant parameters from microscope images of cells.

The following chapter is reproduced from the preprint "SAMCell: Generalized Label-Free Biological Cell Segmentation with Segment Anything." The manuscript is currently under peer-review by BMC Bioinformatics.

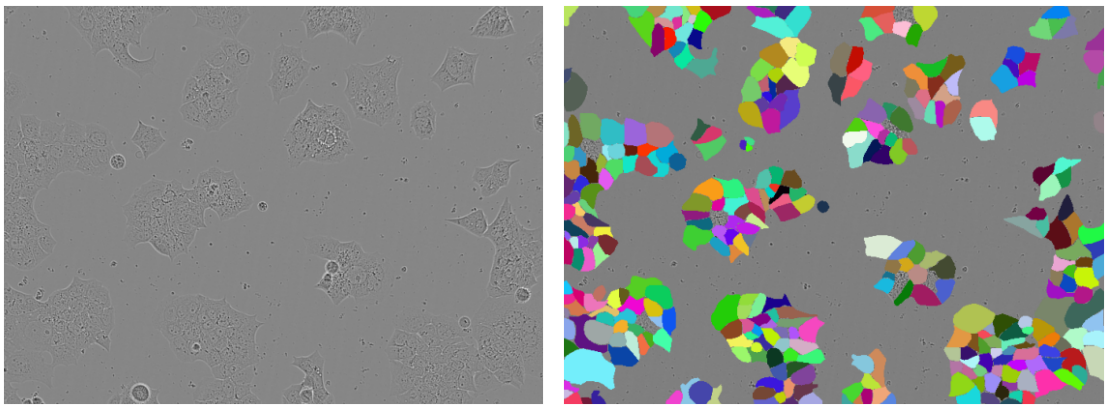
2.2 Background

Cell culture is widely used for biological research applications. In conducting cell culture, microscopy is used to assess cell attributes like confluency (density of cell growth), morphology (cell shape), and count - metrics which relate to cell health and allow a researcher to assay viability visually throughout the cells' growth period [6]. These parameters, typically obtained manually by a skilled biologist, are important to the success of biological experiments. Confluency, for example, is often used to gauge when cells are ready to be passaged, differentiated, or otherwise manipulated [3] and requires experience to assess accurately by eye.

Automating this process is an open area of research. Automatically segmenting cells is a difficult problem. Data annotation is especially laborious as cells routinely number in

the hundreds per image. Further, due to differences in imaging parameters like contrast and field brightness, high variance exists between images from different microscopes even of the same imaging technique (e.g., Phase-Contrast, Brightfield). Finally, cells are often tightly clumped together and have difficult to discern edges as illustrated in Figure 2.1. An ideal model would accurately predict cell boundaries for a wide range of images and cell types to reduce the need for manual data annotation. If accurate cell boundaries could be produced, relevant parameters to cell health like average number of nearest cell neighbors, average cell aspect ratio, and average cell size could be automatically calculated and reported. Reliably automating the segmentation of microscopy images across cell types and imaging parameters has not yet been achieved [7].

Driven by the needs of cell culturing, we are concerned with finding the boundaries of entire, unlabeled cells in this work. Unlike other approaches [8], we do not explore segmenting the nucleus or other regions as separate from the remainder of the cell.



(a) LIVECell Image (b) Annotation Overlayed

Figure 2.1: (a) An image from the LIVECell Test Set containing clumped cells with hard to distinguish edges, (b) overlayed ground-truth masks

Interestingly, cell segmentation poses new challenges not seen in standard, multi-class segmentation like the previous SAM fine-tuning works. For example, cells are often packed tightly in microscope images, with weak or even ambiguous edge features. As such, care must be taken not to merge adjacent cells in predicted masks. One approach, employed

by the introductory U-Net paper, is to segment the image into two categories, “cell” and “background”, with a minimum 1-pixel border of category background between adjacent cells. Then, connected components of category “cell” are extracted from the prediction as individual cells. To avoid clumped cells being merged by the model, a pixel-wise weighted loss function with a higher weight near cell boundaries is used [9]. Unfortunately, due to the constraint that predicted cell masks cannot be in contact with each other, this approach restricts the model’s ability to accurately predict cell mask when dealing with clumped cell masks. Further, the light-weight U-Net model can under-fit larger datasets and has a reduced potential to benefit from pre-training compared to larger models.

Subsequent work expands on this idea, introducing a 3rd “cell border” category and experimenting with various loss function weightings. After classification, these edge pixels are assigned to the nearest cell [10]. Although this additional category eliminates the need for a background border, “cell border” category pixels can become difficult to assign to the correct cell when cells are not circular or contain fine features, like long offshoots. This approach still has limited benefit from pre-training and like the base U-Net has the potential to under-fit large datasets. From initial experiments with Segment Anything, we observe that for cells with weak edge features, a three-category classification with a weighted loss function is insufficient to prevent the merging of clustered cells.

Another prominent cell segmentation work, Cellpose, uses a modified U-Net architecture to estimate a specialized vector field, computed from a simulated diffusion process [11]. The authors define a diffusion process that produces a vector field where, for each cell, gradient vectors point away from the cell’s center in all directions. Cell masks are then computed by finding the fixed points of this vector field, locations where the vectors from neighboring cells intersect. Unfortunately, these vector fields are expensive to compute. Further, because Cellpose calculates and predicts this vector field independently for the x and y components, rotation in data augmentation becomes non-trivial. Finally, being based on U-Net, the model may under-fit larger datasets and has a reduced advantage from

pre-training.

In April 2023, Meta released Segment Anything Model (SAM) [12], a general model for image segmentation. SAM was trained on a diverse dataset of 11 million everyday images containing, in total, more than 1 billion segmentation masks. Due to this extensive dataset, Segment Anything is a great generalist: showing strong performance even on datasets not seen during model training. Given a point or bounding box in an image (the "prompt"), SAM can produce a reasonable segmentation boundary. Alternatively, SAM can generate masks automatically, by uniformly sampling points around the image, and keeping the most confident segmentations. SAM was trained on 11 million everyday images containing, in total, more than 1 billion segmentation masks. Thanks to this extensive dataset, SAM can produce a reasonable boundary for a wide set of objects.

Because of SAM's strong performance on natural images, there have been attempts to fine-tune it for the biomedical domain. MedSAM [13], for example, fine-tunes SAM using more than a million images collected from various medical imaging modalities, like X-Rays, Computed Tomography (CT) Scans, and ultrasounds. These authors preserve SAM's prompting capability, training MedSAM to determine masks from a bounding box. In training, the authors completely update the weights in the image encoder and mask decoder, while freezing the prompt encoder. MedSAM delivers strong results, exceeding baselines in a variety of segmentation tasks and offering impressive zero-shot performance on unseen datasets. Preserving the prompt does allow a user to have greater control over the model's segmentation. This prompt based approach is less useful for cell segmentation, however, because of the sheer number of cells in microscopy images. Drawing a bounding box or selecting a prompt point for a large number of cells would become incredibly laborious and limit the model's usefulness to biologists.

SAMed [14], a concurrent work to MedSAM, fine-tunes SAM to segment individual organs in CT scans. Interestingly, SAMed does not require a prompt and, unlike SAM, can segment images into distinct categories, like liver, stomach and pancreas. Further, SAMed

is trained on the comparably smaller scale Synapse multi-organ CT dataset, which consists of just a few thousand images. SAMed uses Low Rank Adaptation [15] to fine-tune SAM's image encoder, while retraining all parameters in the mask decoder. SAMed's ability to segment images without a prompt is a desirable characteristic for cell segmentation. However, because SAMed categorizes images into a discrete number of categories, we find that it still is prone to combining adjacent cells, like the U-Net approaches [9, 10].

Hoping to bring Segment Anything's generalization to the field of cell segmentation, we were motivated to adapt SAM to the new domain. We present SAMCell, a fine-tuned approach based on SAM for predicting cell boundaries in microscope images. Unlike the default SAM, we observe success in cases even when cells are densely packed or boundaries are soft. SAMCell inherits Segment Anything's main advantages: a high parameter, Vision Transformer [16] based architecture and extensive pretraining. These advantages improve generalization and allow our method to better fit a large dataset, compared to existing approaches. As such, we find that our method exceeds the performance of existing approaches on both cells similar to those seen in training (test-set) and on completely novel cell lines from images taken by other microscopes (zero-shot).

2.3 Methods

2.3.1 Datasets

We evaluate SAMCell for two main use cases. First, the case where a sufficiently large dataset is available to train SAMCell. After training, we can observe performance on the test set, a portion of the dataset not seen during training. In this case, the training images and test images for evaluation are similar. We also evaluate the "zero-shot" case, where no annotated dataset is available. In this case, we train the model on a wide range of cell images, with the aim of generalization. Then, we evaluate on images dissimilar to what the model has seen during training. We select datasets to evaluate both these "test set" and "zero shot" use cases.

Large-Scale Datasets for Test-Set Evaluation

For model training and test set evaluation we select two existing large-scale datasets of cell images. First, we choose the LIVECell dataset [17] which consists of more than 5000 phase-contrast images, across 8 cell types. The dataset consists of cells with a range of confluency and morphology. Further, many images have low contrast. Together, these two characteristics produce a difficult segmentation task which we feel is useful in evaluating model performance. Unfortunately, all images in this dataset were produced with the same phase-contrast microscope. Even among phase-contrast microscopy, different microscopes can produce dissimilar pictures because of variations in imaging parameters like contrast, brightness, and resolution. As such, we feel that only including images from a single microscope limits LIVECell's coverage of the cell segmentation task.

To complement LIVECell, we also employ the Cytoplasm dataset gathered by the authors of the Cellpose model [11]. This dataset consists of microscopy images scraped from the internet which were subsequently annotated. Because of the nature of web scraping, these images came from a variety of different individual microscopes. Further, a variety of microscopy techniques are included in this dataset, including bright-field images, membrane-labeled cells, and fluorescent labeled proteins. This dataset contains around 600 images – a smaller-scale dataset than LIVECell. Despite this, we feel the Cytoplasm dataset is better suited for training generalist models because of the wider range of images included. Some of the images in the Cytoplasm dataset are 3-channel, with separate colors showing the nucleus and membrane using fluorescent labeling. Because we are specifically concerned with label-free, whole-cell segmentation in this work, we convert all images in this dataset to grayscale for training and evaluation. Because images in the LIVECell dataset are of differnt sizes, we also resize all images to 512x512, preserving aspect ratio by adding a border as needed. A majority of images are near this size and close to square. This resizing allows us to more easily train our model and baselines by avoiding jagged arrays.

Small-Scale Datasets for Zero-Shot Evaluation

We produce two novel, small-scale datasets. Each contains 5 images of varied confluency and microscope contrast annotated by an expert biologist co-author. These datasets contain images of the Human Embryonic Kidney (HEK) 293 and Neuro2a (N2a) cell lines respectively. We name these datasets after our lab, the Precision Biosystems Laboratory (PBL), and the cell lines they contain, yielding *PBL-HEK* and *PBL-N2a* respectively.

Although these datasets are of small scale, we feel they are sufficient for zero-shot evaluation as each image contains approximately 300 cells. These datasets are entirely used for testing, to evaluate how SAMCell and our baselines generalize to unseen images from different Phase-Contrast microscopes. We evaluate zero-shot performance using SAMCell and baseline models trained on the Cellpose Cytoplasm dataset, because of the higher diversity in microscopy techniques and cells imaged.

2.3.2 Evaluation Metrics

For evaluation, we employ a common cell segmentation metric used by existing approaches like StarDist [18] and CellPose [11]. We first choose a threshold value τ between 0 and 1. This threshold sets the minimum Intersection over Union (IoU) between a predicted cell I_{pred} and a cell's ground truth mask I_{true} which is considered a True Positive as follows:

$$IoU = \frac{I_{pred} \cap I_{true}}{I_{pred} \cup I_{true}}$$

Any detections whose IoU falls below τ are considered false positives as they do not match any cells in the annotation closely enough. Likewise, any ground truth annotations without a matching prediction are considered False Negatives. Given these True Positives (TP), False Positives (FP), and False Negatives (FN) we can compute the Average Precision (AP) for a given τ:

$$AP_\tau = \frac{TP_\tau}{TP_\tau + FP_\tau + FN_\tau}$$

By evaluating AP over a range of τ values, we can numerically measure each model's performance and compare it to other models.

2.3.3 The Default Segment Anything Model

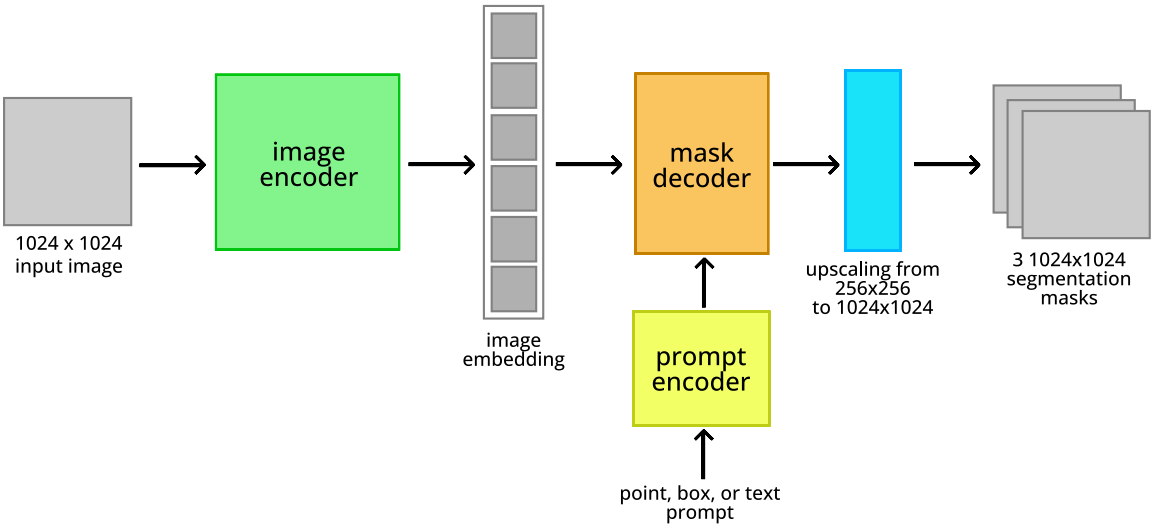


Figure 2.2: The architecture of Segment Anything Model.

As mentioned, Segment Anything Model is a state of the art generalist model for image segmentation. A diagram of SAM's architecture is displayed in Figure 2.2. SAM consists of a large image encoder, based on ViT [16], that converts a 1024x1024 image into a condensed embedding vector. Optionally, an input mask can be added to this embedding, as an additional input to the model. The image embedding is then supplied to a lightweight mask decoder along with an encoded prompt (or a default prompt embedding if no prompt is supplied). Subsequently, this mask decoder generates three sets of 256x256 binary masks, each accompanied by a "score" value denoting the model's confidence for each mask. Finally, these masks are upscaled bilinearly to match the input dimension. Multiple masks are supplied to resolve ambiguity if there are multiple reasonable segmentations for a certain prompt. Meta offers pretrained variants of Segment Anything in 3 sizes, corresponding to the size of the image encoder: Base, Large, and Huge.

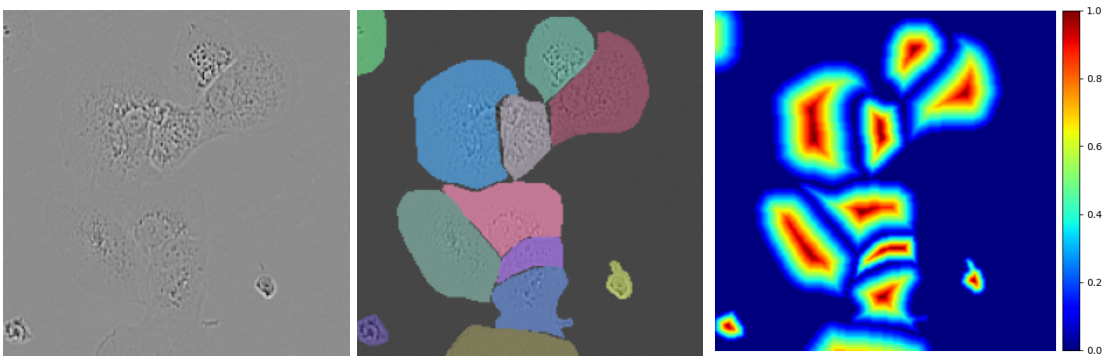
Unfortunately, when employing the existing fine-tuning approaches seen in MedSAM

[13] or SAMed [14], difficulties arise with issues specific to the domain of cell segmentation. Using standard 2 (cell, background) or 3 (cell, cell border, background) category segmentation with the Segment Anything architecture, we observe poor results when cells have low contrast boundaries as is seen in Figure 2.3a. We find performance to be poor even when employing pixel-wise weighted loss functions [9, 10] which have been used to tackle this problem in the past. When these these common characteristics arise, adjacent cells are predicted as being merged together along weak edges. Additionally, these subpar segmentation results have a detrimental impact on the accuracy of insights derived from segmentation. For instance, they can lead to an underestimation of the cell count and an inflation of the average cell size when cells are merged.

2.3.4 Segmentation as Regression

We formulate cell segmentation as a regression problem, drawing inspiration from CellPose [11], to address difficulties with soft edges. Rather than predicting segmentation masks directly like U-Net [9], MedSAM [13], or SAMed [14], we predict a real-valued distance map, describing the euclidean distance from each pixel to its cell's boundary (or 0 if the pixel is not part of a cell). We normalize the map such that each pixel is a real number between 0 and 1 by dividing the distance map by the max distance within each cell. An example of an annotation and its associated distance map is shown in Figures Figure 2.3b and Figure 2.3c respectively. Existing functions for efficiently producing these distance maps are present in common Python packages like SciPy [19] and OpenCV [20]. By predicting these distance maps instead of discrete categories, this problem of merged cells can be avoided.

We compute a distance map using each annotation mask in the training set before training time. Because these images are invariant or almost invariant to all our data augmentation strategies, we can avoid recomputing these maps in every training step.



(a) Original Image Patch (b) Annotation Overlayed (c) Computed Distance Map

Figure 2.3: (a) A cropped microscope image from the test set, (b) overlayed ground-truth cell masks, and (c) a distance map computed from these cell masks (right).

2.3.5 Architecture

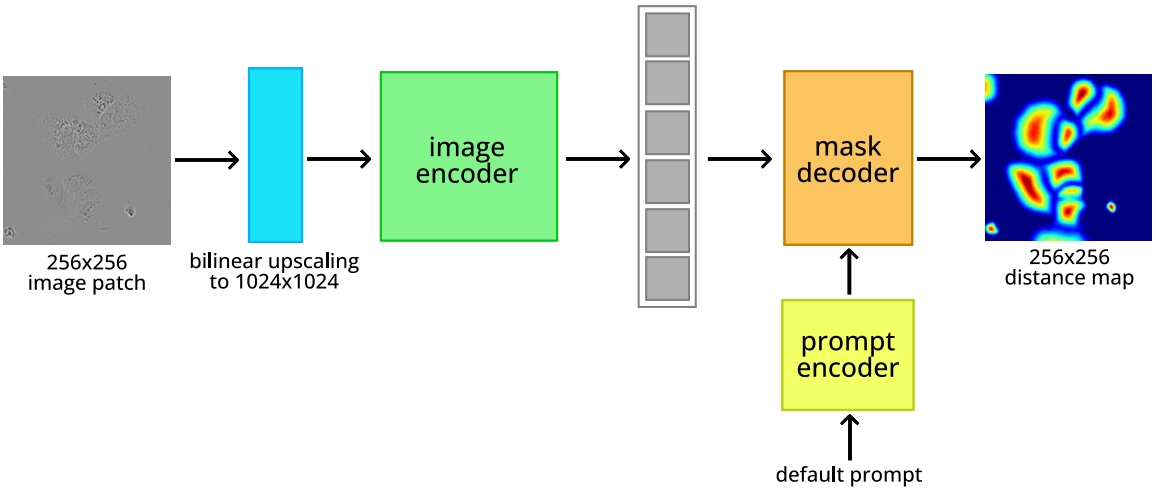


Figure 2.4: The architecture of SAMCell.

An architecture diagram for SAMCell is shown in Figure 2.4. Because of VRAM constraints, concerns with inference time, and SAM's modest performance improvements with larger models, we choose to fine-tune SAM-base.

Pre-Processing:

We find that applying Contrast Limited Adaptive Histogram Equalization (CLAHE) [21] is a beneficial preprocessing step. This approach eliminates any nonuniformities in bright-

ness across the microscope field, and increases the visibility of hard to distinguish edges. Subsequently, following SAM [12], we normalize all input images to zero-mean and unit-variance.

Sliding Window Approach:

To inference SAMCell, we first separate an input image into a series of 256x256 patches using a sliding window approach, like in U-Net [9]. We use an overlap of 32 pixels on either side of these patches, to avoid poor classification of areas partially in frame. To convert each 256x256 patch to the 1024x1024 input size of the image encoder, we use bilinear upsampling. Then, we pass each image through our fine-tuned SAM, producing a distance map prediction. We apply the sigmoid activation function to the mask decoder output such that the output (predicted distance map) is in the correct range: $[0, 1]$.

Eliminating Prompt:

Manually supplied prompts for densely packed cells are difficult and laborious to produce, as cells in culture often number in the hundreds per image. As such we forgo SAM's default prompting capability. To accomplish this, we employ an approach similar to that used by SAMed [14]. We freeze the prompt encoder during fine-tuning and always input SAM's default prompt. Because the prompt embedding is static over fine-tuning, the mask decoder learns to predict the distance map from the image embedding exclusively.

Finetuning Methodology:

Drawing inspiration from previous SAM fine-tuning approaches [14, 13], we fine-tune all parameters in SAM's lightweight mask decoder. Unlike SAMed, in testing we find Low Rank Adaptation to be detrimental to performance. As such, we fine-tune all parameters in the image encoder as well. Because we aim to inference without user-supplied prompts, we freeze the prompt encoder

Post-Processing:

Post-processing must occur to convert a predicted distance map into predicted cell masks. An outline of this process is shown in Figure 2.5. First, via thresholding the predicted distance map, we create two binary images. The first image, we denote "binary mask", tells whether a certain pixel belongs to a cell or background. Empirically, we find that distance map > 0.05 yields a good binary mask. The second image, we denote "cell centers", must contain exactly one connected component per cell, someplace in the body of the cell. We find that distance map > 0.5 produces good masks for this. Fortunately, we find that these threshold values are not sensitive to the image being inferenced or the training set used. As such, we use these same threshold values for all experiments.

We apply the Watershed algorithm [22], a classical approach to boundary detection, to recover cell masks from the distance map. The idea of this algorithm is inspired by how water floods a 3D region: first collecting at the lowest points, then rising to higher regions. As the water rises higher, previously separate pools of water join together.

Initially, sets of "marker points" in the image are given, with each set of points corresponding to a single cell. In SAMCell, we use the boundary of each connected component in the "cell center" binary mask as a set of marker points.

We negate the distance map so that the centers of cells are the low in value and the edges of cells are higher. A synthetic flooding process then starts from each set of these marker points. Just as water floods starting at low elevation, moving up in a controlled manner, the Watershed algorithm simulates this flooding process by iteratively raising the water levels around the marker points.

At each iteration, the algorithm identifies the lowest neighboring pixel to the current flooding front, representing the point where water would naturally flow next. It then raises the water level at that pixel by one unit. This process continues until the water from different marker points merges at a common boundary - the segmentation boundary of the

cell. To prevent classifying background pixels as part of a cell, we only run the above algorithm for regions where the "binary mask" acquired from thresholding denotes that a cell is present. As with producing the distance map, existing high-speed implementations of this algorithm exist in common libraries like OpenCV [20] and scikit-learn [23].

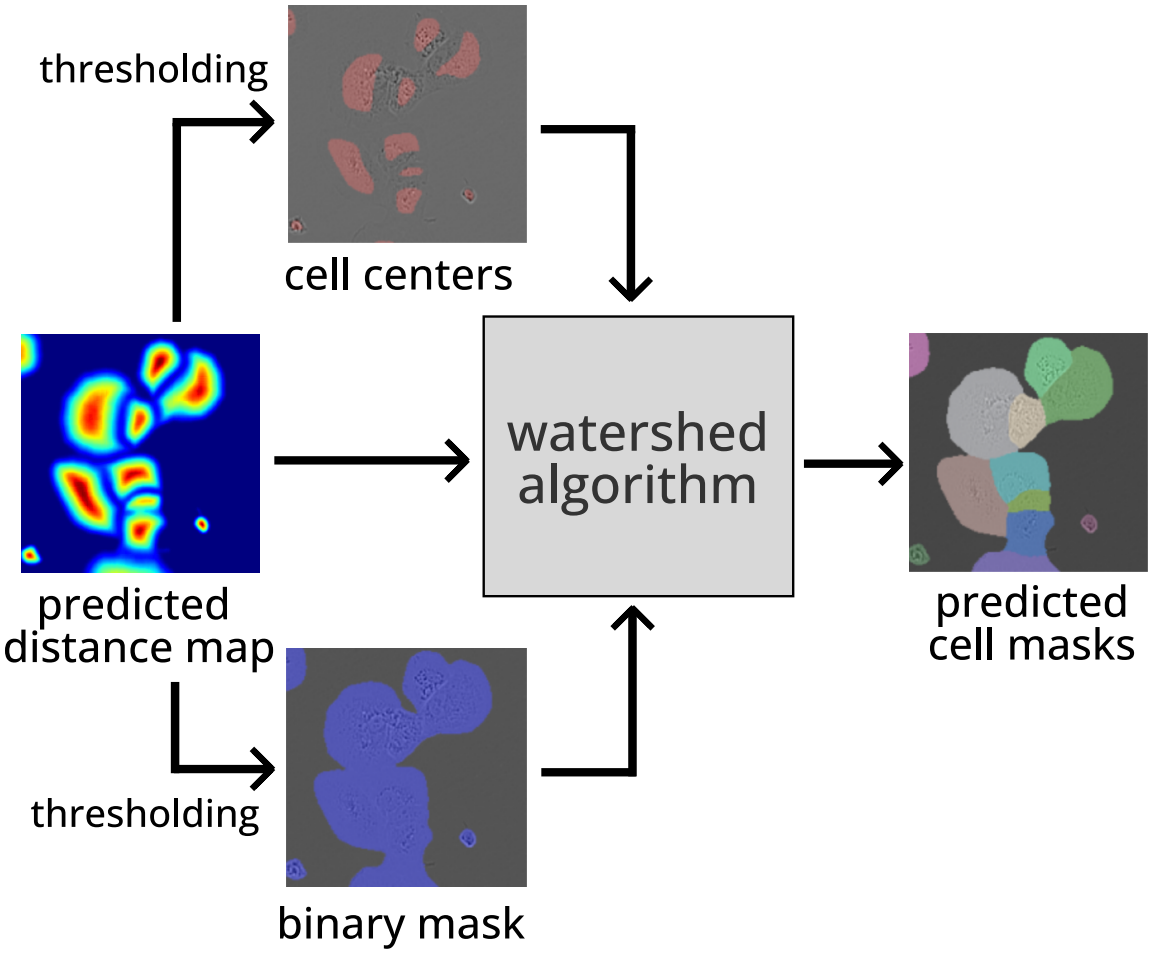


Figure 2.5: The post-processing procedure used to recover a set of distinct cell masks from a SAMCell distance map prediction.

2.3.6 Data Augmentation

To encourage generalization, we employ aggressive data augmentation during training as follows:

- random horizontal mirroring

- random rotation between -180° and 180°
- random rescaling between 80% and 120% of the original image dimensions (preserving aspect ratio)
- random brightness adjustment between 95% and 105% of original
- randomly invert the image

Once data augmentation has been applied to an image, a random 256x256 patch of the image and associated augmented distance map is used for fine-tuning. As such, every epoch, SAMCell is trained on one unique, augmented 256x256 patch from every image.

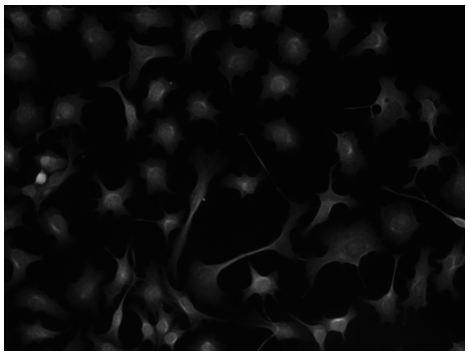
2.3.7 Training Protocol

We train SAMCell for 40 epochs on an Nvidia RTX 4090 GPU, requiring about 8 hours of training time. We start with the pretrained SAM-base model and employ the AdamW [24] optimizer with an initial learning rate of 0.0001, and a weight decay of 0.1. We set β_1 to 0.9 and β_2 to 0.999. Following SAMed [14], we employ a learning rate warm-up [25, 26] with a period of 250 and a linear decrease in learning rate to 0 over the training period. Due to GPU VRAM limitations, we use a batch size of 2 for training. Because we pose the task as a regression to a distance map, we train with L2 loss.

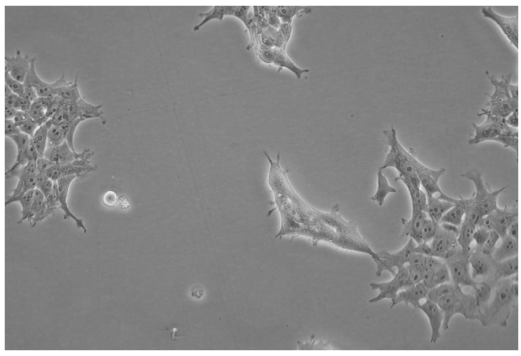
2.4 Results and Discussion

2.4.1 Datasets

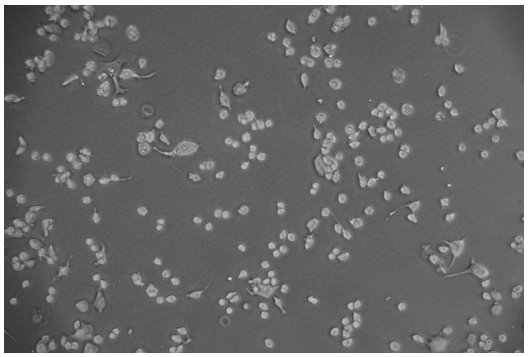
We aim to evaluate both test-set and zero-shot performance of our model. As shown in Figure 2.6, the zero-shot datasets we create differ in cell type, microscope contrast, and field brightness, compared to a sample image from the Cytoplasm training dataset. Thus, we argue these datasets act as a realistic evaluation of model generalization.



(a) Cytoplasm Dataset



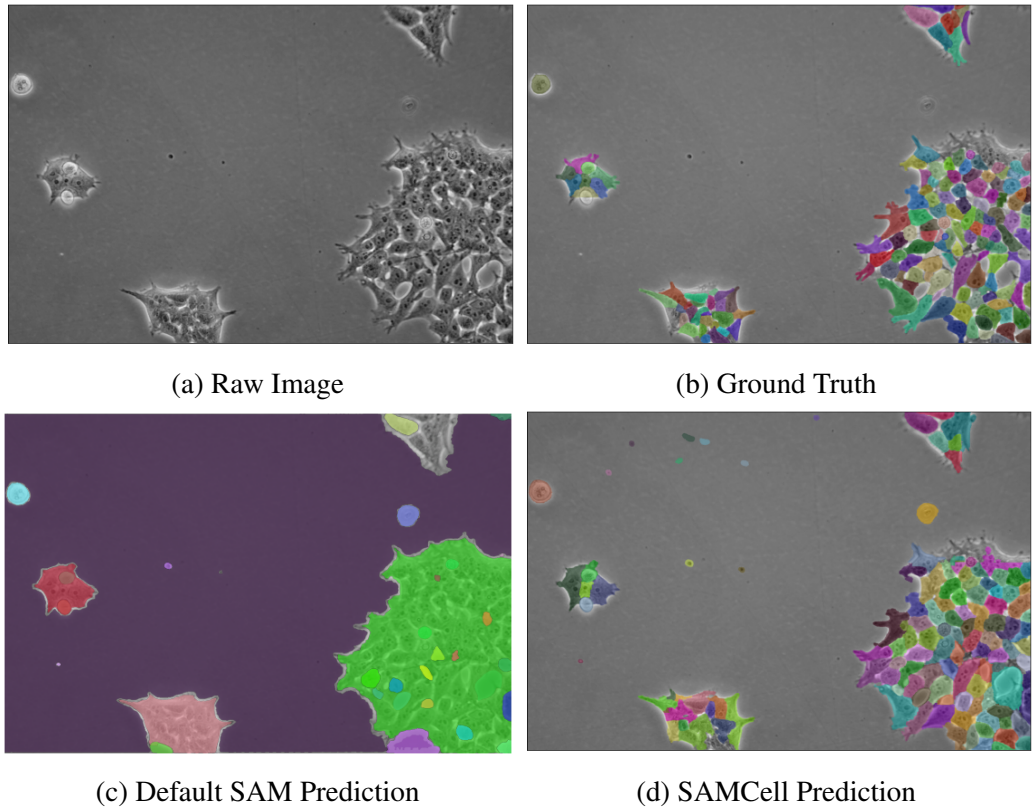
(b) PBL-HEK Dataset (Ours)



(c) PBL-N2a Dataset (Ours)

Figure 2.6: (a) Example image from the Cytoplasm dataset used for training and example images from each of our two zero-shot datasets, (b) PBL-HEK and (c) PBL-N2a.

2.4.2 Comparison to Default SAM



(a) Raw Image (b) Ground Truth

(c) Default SAM Prediction (d) SAMCell Prediction

Figure 2.7: (a) A single Phase Contrast image of HEK cells, (b) annotated manually by an expert, (c) automatically annotated by Meta's SAM-huge model, and (d) automatically annotated by our SAMCell model. The above image is from our PBL-HEK zero-shot dataset.

Although the off-the-shelf Segment Anything Model demonstrates impressive performance on a variety of tasks, we observe some common failure modes that can limit its usefulness to microscopy. We illustrate these downfalls in Figure 2.7. When attempting to segment densely packed cells in "automatic mask generation" mode (sampling prompt points in a uniform grid), SAM can often segment large clumps, rather than individual cells. Further, we find that SAM is prone to segmenting empty portions of the cover slip. Although this shortcoming can be overcome by a biologist prompting SAM for every cell, this introduces a manual, tedious process.

These failures likely stem from SAM's training set, which, although large in scale, is

focused on everyday images like segmenting food in a supermarket or people in a sporting event [12]. Cell segmentation is a unique task: cells routinely have irregular shapes and are often clumped together with difficult to discern boundaries. As such, SAM does not have a strong prior for the appearance of cells because of its more generic dataset. This leads to SAM predicting boundaries in images that often do not correspond to individual cells.

Our fine-tuning of Segment Anything effectively resolves this shortcoming. By training SAM with a large-scale dataset of microscopy images, the model learns a prior for the appearance of cells. This, along with our other modifications, results in the significantly enhanced performance presented in Figure 2.7d.

2.4.3 Baseline Methods

In addition to the default Segment Anything Model, we select 3 existing cell segmentation models as baselines for comparison:

U-Net [9]: U-Net is a common image segmentation architecture consisting of a convolutional neural network arranged in a characteristic "U" shape, which allows both local and global features to be combined into the segmentation result. In line with the original paper, we use a two-category model (comprising background and cell categories) with a pixel-wise weighted loss function to encourage the precise delineation of cell borders. Our model is structured with four up/down sampling blocks, each comprising two convolutional layers with a 3x3 kernel size. The segmentation result is obtained by first producing a binary mask with a simple threshold of 0.5, and then extracting connected components. We train for 50 epochs on each dataset using the Adam optimizer and a learning rate of 0.0001.

Stardist [18]: Stardist predicts cell shape as series of line segments radiating from the cell's center point. These line segments are evenly spaced around the center and variable in length. The ends of these line segments are connected to form the boundary of a cell. This approach leverages a modified U-Net architecture to predict the component line segments

for segmentation. We train this model for 50 epochs on each dataset, with the default 32 line segments per cell.

Cellpose [11]: Cellpose attempts to predict specialized x and y gradients from an input image. Then, these gradients are combined to produce a smooth increase in value from cell center to the edges. The watershed algorithm is used to recover cell boundaries from these gradients. Like Stardist, a modified U-Net architecture is used as a backbone for predictions of, in this case, image gradients. Pretrained models from the Cellpose authors exist for both the LIVECell and Cytoplasm datasets. As such, we simply use these weights without modification[1].

2.4.4 Test-Set Performance

Table 2.1: A performance comparison between our fine-tuned model, SAMCell and 3 existing cell segmentation baselines.[2]

Threshold τ	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90
Dataset: *Livecell*									
U-Net	0.2983	0.2640	0.2300	0.1901	0.1444	0.0964	0.0542	0.0234	0.0056
Stardist	0.6162	0.5670	0.5119	0.4471	0.3722	0.2852	0.1881	0.0924	0.0224
Cellpose	0.6094	0.5688	0.5278	0.4807	0.4230	0.3491	0.2558	0.1512	0.0554
SAMCell (ours)	**0.6243**	**0.5805**	**0.5363**	**0.4865**	**0.4266**	**0.3516**	**0.2594**	**0.1559**	**0.0572**
Dataset: *Cellpose Cytoplasm*									
U-Net	0.2965	0.2680	0.2423	0.2058	0.1703	0.1327	0.0939	0.0534	0.0183
Stardist	0.5101	0.4679	0.4145	0.3500	0.2818	0.2027	0.1236	0.0533	0.0109
Cellpose	0.5014	0.4668	0.4275	0.3803	0.3253	0.2594	0.1927	**0.1231**	**0.0536**
SAMCell (ours)	**0.5553**	**0.5067**	**0.4524**	**0.3983**	**0.3366**	**0.2698**	**0.1996**	**0.1231**	0.0478

As shown in Table 2.1, SAMCell demonstrates strong test-set performance, surpassing baseline performance on both the LIVECell and Cellpose Cytoplasm test sets. We hypothesize this success can be attributed to two key factors: Firstly, SAMCell inherits SAM's image encoder which is pretrained on a diverse dataset of 11 million diverse, everyday-life images. This pretraining provides SAMCell with a strong prior for objects in general,

[1]Cellpose pretrained models are available at https://cellpose.readthedocs.io/en/latest/models.html

[2]Because of our modifications to the Cellpose Cytoplasm dataset - conversion to grayscale and resizing to 512x512 - as mentioned in Section subsubsection 2.3.1, our AP metrics are lower than those cited in Cellpose [11] for all baselines.

helping the model to better detect features that contribute to a cell boundary while ignoring extraneous features like field brightness or microscope contrast.

Secondly, SAMCell boasts a larger parameter count compared to our U-Net-backed baselines. This allows our model to better fit large training sets, like LIVECell and Cytoplasm, without underfitting. Further, we speculate this higher parameter count better allows the model to better capture complex features, like low contrast cell borders or tightly packed clumps of cells.

2.4.5 Zero-Shot Performance

Table 2.2: A zero-shot performance comparison between SAMCell and baselines.

Threshold τ	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90
Dataset: *PBL-HEK*									
U-Net	0.0629	0.0479	0.0328	0.0182	0.0109	0.0052	0.0023	0.0004	0.0001
Stardist	0.1533	0.1417	0.1166	0.0935	0.0659	0.0345	0.0155	0.0048	0.0004
Cellpose	0.1792	0.1553	0.1317	0.1084	0.0883	0.0599	0.0332	0.0116	**0.0032**
SAMCell (ours)	**0.2830**	**0.2487**	**0.2116**	**0.1766**	**0.1391**	**0.0921**	**0.0495**	**0.0163**	0.0016
Dataset: *PBL-N2a*									
U-Net	0.4082	0.3670	0.3125	0.2530	0.1808	0.1079	0.0493	0.0135	0.001
Stardist	0.7127	0.6704	0.5908	0.4941	0.3573	0.2391	0.1280	0.041	0.0051
Cellpose	0.7165	0.6863	0.6455	0.5839	0.4829	0.3586	0.2103	0.0824	0.0124
SAMCell (ours)	**0.7608**	**0.7263**	**0.6762**	**0.5963**	**0.4911**	**0.3636**	**0.2358**	**0.1038**	**0.0229**

As with test-set evaluations, we notice impressive zero-shot capabilities with SAMCell seen in Table 2.2. We observe our method besting baseline approaches for a majority of values of τ. For both the PBL-HEK and PBL-N2a datasets, we observe a very strong lead over baselines for lower values of τ. However, as τ increases, the extent of this advantage diminishes. We postulate this occurs because of Segment Anything pretraining which is inherited by SAMCell. SAM pretraining provides SAMCell with prior knowledge of boundaries in non-microscopy images. This information is helpful in detecting the boundaries of cells. However, as τ increases, forcing a tighter boundary for a valid detection, the value of this prior knowledge diminishes compared to information gained from training from the Cytoplasm dataset. This results in a competitive or slightly superior model than

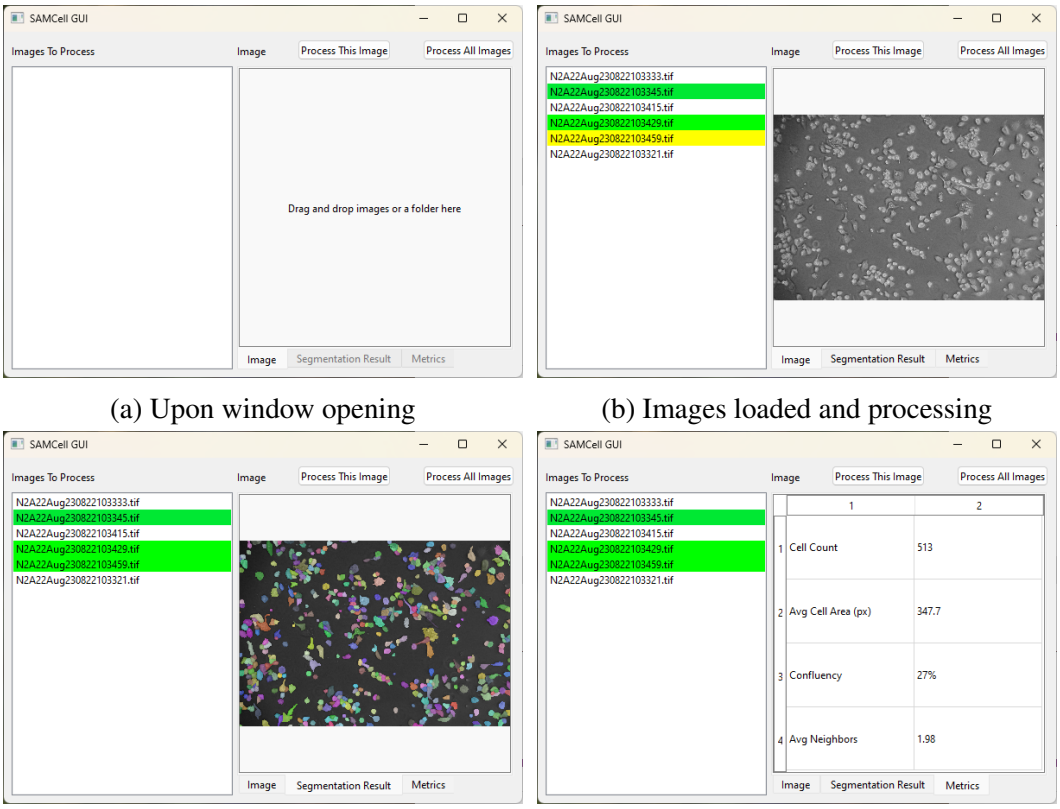
Cellpose at higher τ values.

Across all approaches, we observe much higher mAP values for PBL-N2a compared to PBL-HEK. We attribute this to differences in appearance between the two cell lines. Neuro-2a (N2a) cells contain a circular morphology with high contrast borders, as shown in Figure 2.6c. Further, these cells are less likely to grow in tightly packed clumps. Conversely, Human Embryonic Kidney (HEK) 293 cells are prone to grow in more densely packed regions and are less circular in form, as seen in Figure 2.6b. We feel these differences result in PBL-HEK producing a more difficult segmentation task, thus suppressing mAP metrics.

2.4.6 Limitations

The main strength of the Segment Anything Model architecture is its large, ViT based image encoder, but this increases computational requirements compared to other lighter weight baselines. Because of the model's size, it can capture complex features and achieve high performance on large and diverse datasets - in both the original training set [12], and the microscopy datasets used in this work. Unfortunately, running inference on a larger model is more computationally expensive. With a high-end consumer GPU, the Nvidia RTX 4090, we find that running SAMCell on a single image takes about 5 seconds. We find that under the same conditions, our baselines inference in well under a second. This is likely because our baselines are backed by the much slimmer U-Net architecture which requires reduced computational resources to run.

We find this difference is exacerbated when a GPU is not available as would be the case on lower-end computers. We find that running SAMCell on CPU (an AMD Ryzen 7 7700X) takes approximately 2 minutes and 20 seconds per image. We observe that our lighter-weight baselines are able to inference in a just few seconds under these conditions. As such, we recommend running SAMCell on a higher-end computer with a GPU to mitigate this limitation.



(a) Upon window opening

(b) Images loaded and processing

(c) Segmentation masks displayed

(d) Predicted metrics displayed

Figure 2.8: A demonstration of SAMCell's interface throughout the various stages of use.

2.4.7 User Interface

To make SAMCell more accessible to users without a machine learning background, we create a user-friendly front-end for SAMCell. Using this, users can click and drag microscopy images into a straightforward window and see results from SAMCell's automatic segmentation. The workflow of the user interface is shown in Figure 2.8. The landing page invites a user to drag and drop microscopy images as shown in Figure 2.8a. Then, as shown in Figure 2.8b, users can select and view images to process with SAMCell. Images in the list that have been processed are denoted green, and images being currently processed are highlighted in yellow. After an image is processed, segmentation masks can be visualized as seen in Figure 2.8c. Finally, we supply a few relevant metrics extracted from the segmentation result to aid biologists culturing cells: number of cells detected, average cell area, confluency, and number of neighbors for each cell. A table with this information is presented to the user, seen in Figure 2.8d. As mentioned in Section subsection 2.4.6, we recommend a computer with a GPU to use SAMCell, to ensure images are processed quickly.

2.5 Conclusion

We present SAMCell, a cell segmentation model based on the Segment Anything Model architecture. Using our model, we observe state of the art performance exceeding powerful and commonly cited baselines like Cellpose and Stardist. We evaluate our model for two main use cases: when a training dataset for a similar task is available (test-set performance) and when no training dataset is available for the specific task (zero-shot performance). In both of these circumstances, our approach exceeds existing models in quantitative evaluation. To make SAMCell more accessible, we present a user-friendly interface to make this powerful model available to biological researchers without a machine learning background.

We supply training and evaluation code[3] for SAMCell, as well as our user interface[4] on Github. Additionally, we provide the trained weights for SAMCell, as well as our custom datasets (PBL-HEK and PBL-N2a).[5]

[3]https://github.com/NathanMalta/SAMCell
[4]https://github.com/NathanMalta/SAMCell-GUI
[5]https://github.com/NathanMalta/SAMCell/releases/tag/v1

CHAPTER 3
AUTOMATING PATCH CLAMPING

3.1 Introduction

Because of the high precision and prerequisite knowledge required, manual patch-clamping has historically been plagued by low throughput. Recent advances in automation [27, 28], however, promise significant improvements in the speed of experiments while reducing required required labor of scientists.

This section explores the equipment setup, software, and protocols utilized to streamline and improve throughput of patch clamping experiments via automation. The aim is to provide a comprehensive overview of the automated processes involved in obtaining single cell recordings, contributing to the advancement of efficient and reliable experimental procedures in the field of neuroscience research. We attempt to create a solution that is open-source, able to be quickly adapted to different experimental setups, and easy to connect to deep learning approaches to encourage additional automation.

3.2 Related Works

There is serious interest in the neuroscience community surrounding increasing the experimental throughput behind patch-clamping. As such, prior work for patch clamping automation exists:

3.2.1 Patcherbot [27]

One of the earlier approaches to patch clamping automation created by our lab is the patcherbot. The patcherbot allows for automated gigaseal and break-ins given a user selected point on a cell. Further, it allows for automated pipette cleaning, allowing for easy

reuse. Unfortunately, the patcherbot is uses hard coded transform matrices between coordinates systems (e.g. stage and manipulator). This means that any change in manipulator setup means that the system must be manually calibrated by the experimenter, taking a significant time investment. Further, the software is written in LabView, which limits its use to other labs as the language is closed source and requires a subscription[1]. Further, integration of machine learning approaches to further automation is difficult in this language.

3.2.2 Holypipette [29]

Holypipette is an open source python effort to automate patch clamping developed at the Vision Institute in Paris, France. The software has a clean structure, with separation between low level tasks like sending serial commands to a manipulator, and higher level tasks like the steps to patch a cell. Further, being written in Python it is very easy to integrate with machine learning to further automation. Unfortunately, the manipulators, camera, and pressure control system used by our lab are not supported by the default holypipette software suite. Further, testing of existing calibration methods in the software did not yield sufficient accuracy for our patching experiments. As such, although this is a fantastic starting point for automation in our lab, significant modifications are required to run experiments in our lab with this software suite.

3.3 Experimentation Setup: The "Rig"

The following is an overview of the instrumentation on our specific patch clamping rig:

3.3.1 Motorized Axes

Multiple degrees of freedom are available under software control, for both changing the portion of the sample in microscope's view, as well as manipulating cells. The specifics of this setup is as follows:

[1] https://www.ni.com/en/landing/subscription-software.html

Stage

With our microscope, only a small portion of the sample can be viewed at one time. To facilitate automation, it is necessary to automatically move and bring into focus different portions of the sample. As such, we use a motorized stage from Scientifica[2] which allows for automated 3 axis control of the microscope stage over a simple USB serial interface. With this system, the microscope can be moved relative to the sample, allowing a different area in the field-of-view. Thus, a larger number of candidate cells on a certain cover slip can be selected for experimentation.

Micromanipulator

To run patch-clamping experiments, it is necessary for high precision, 3 degree of freedom control of the micromanipulator. In our lab's rig, we use the Sensapex UMP 3 axis micromanipulator[3] because of its incredible endpoint accuracy (100 nanometer rated precision). This endpoint accuracy is paramount because of the small size of cells under experimentation: HEK-293 cells measure just 20 micrometers on average, around half the width of a human hair. Additionally, this manipulator can be easily interfaced with using a C++ SDK[4] or python package[5].

CellSorter

After finding a cell with a desired trait via patch-clamping, some experiments call for the genomic sequencing of this cell. As such, it is necessary to extract specific cells from the cover slip with slight negative pressure and place them into individual tubes for containment.

To achieve this goal, we integrate the CellSorter[6] into our rig. The CellSorter has a

[2]https://www.scientifica.uk.com/products/scientifica-xy-stage
[3]https://shop.sensapex.com/product/ump-3/
[4]http://dist.sensapex.com/misc/um-sdk/latest/
[5]https://pypi.org/project/sensapex/
[6]https://www.cellsorter-scientific.com/

large diameter glass pipette on a movable, vertical axis with a connected pressure system. Using this device, we can extract and separate single cells for analysis after running patch clamping experiments. Like the microscope stage, this manipulator can be moved with a simple USB serial interface.

3.3.2 Camera

Compared to manipulators, microscope camera requirements are not as strict. We select the PCO Panda microscope camera for our rig[7] because of its reasonable 2048x2048 resolution, 16 bit sensor and ease of communication with external software - the manufacturer supplies a python package for capturing frames and changing camera parameters[8].

3.3.3 Electrophysiology

Our rig contains a few instruments for studying electrical properties of cells, as follows:

Amplifier

Patch clamping requires stimulating cells on the order of millivolts, and reading currents on the order of picoamps. To convert from the range required for cellular experimentation to waveforms that can be produced by standard analog to digital converters, we use the Multiclamp 700B amplifier[9]. This instrument essentially acts as a non-inverting op amp circuit, with gains that can be adjusted over a simple software interface.

Analog to Digital Converter / DAQ

To produce analog waveforms for cellular stimulation and read electrical responses, we employ the National Instruments myDAQ. This instrument can reproduce an electrical signal discretized in time and voltage. Most experiments require basic 50% duty cycle square

[7] https://www.excelitas.com/product/pcopanda-42-scmos-camera

[8] https://pypi.org/project/pco/

[9] https://www.moleculardevices.com/sites/default/files/en/assets/data-sheets/dd/cns/multiclamp-700b-microelectrode-amplifier.pdf

wave stimulation, so arbitrary waveform generation is more than sufficient. The myDAQ can also read electrical signals with good precision, using its built in 16 bit analog-to-digital converters (ADCs).

Light for Cellular Stimulation

Certain kinds of experiments require light stimulation of the cells. For example, one project in the lab concerns channelrhodopsins, a specialized protein within the cell membrane which changes the electric potential across the membrane when exposed to a particular wavelength of light. To facilitate these kinds of experiments, we employ the Lumencor Spectra X light engine. This device can emit light in 1 of 6 wavelengths within the visible spectrum. This light is guided through an optical fiber and directed toward the sample. The produced wavelength and power of light emitted can be controlled over a simple serial interface.

3.4 Overview of Software Architecture

We opt to use Holypipette [29] as a base for this automation work due to its simplicity, polymorphic design, and release as free and open source software. Further, this software is written in python, thus making it easy to integrate machine learning approaches to further automation. Holypipette's software separates class files into a few subgroups to encourage modularity and to separate higher and lower level control. This makes it particularly appealing to extend for use in our lab. This class hierarchy is illustrated in Figure 3.1 and is described below:

Devices

Device classes, as shown in the left most column of Figure 3.1, provide a uniform method of interacting with devices that achieve the same goal, even if the lower level communication differs between devices. For example, being motorized axes, the Scientifica Stage and

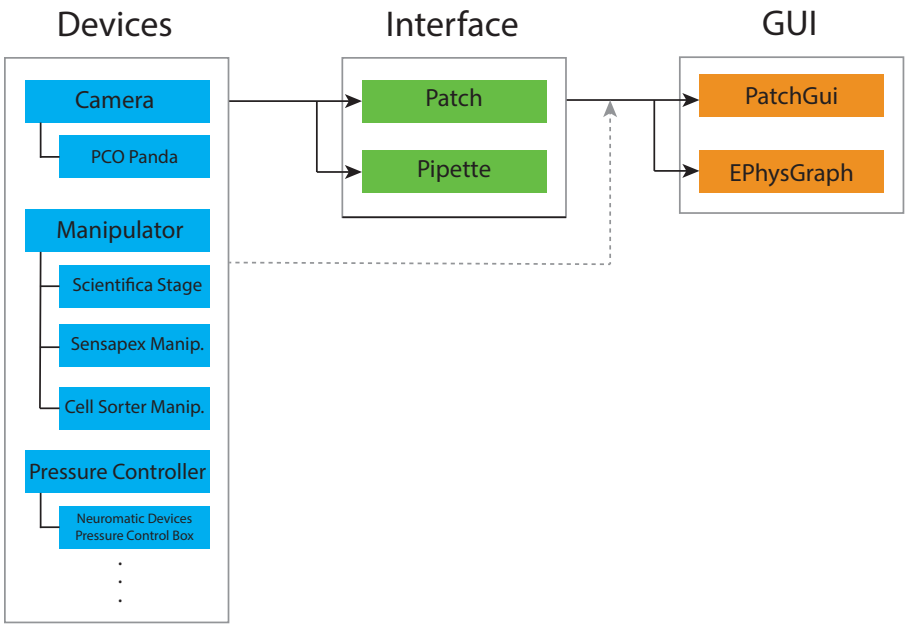


Figure 3.1: A visualization of the software hierarchy used by Holypipette.

Sensapex manipulator have similar functionality. Both can be commanded to a setpoint, or have their maximum speed adjusted. Unfortunately, as each device is built by a different manufacturer, the low level commands used to command these devices is vastly different. For example, Scientifica uses a proprietary serial protocol over USB and Sensapex uses a network protocol. Dealing with these differences in higher level control classes would be poor software design. Instead, holypipette tackles these issues via a parent "Manipulator" class which is extended by all motorized axes. This creates a uniform interface for interacting with all manipulators regardless of their underlying communication protocols.

Interfaces

Interface classes provide higher-level automation, separate from low level control. These are shown in the green center column of Figure 3.1. These classes like "patch.py" or "pipette.py" are passed some subset of the device classes in their constructor and provide code for sequential tasks that can be performed automatically. For example, calibrating the stage, patching a cell, and running a current stimulation protocol all require a series of

commands to run properly. These routines are held in the interface classes.

GUI

The actual graphical user interface (GUI) that experimenters can interact with is contained in so-called "GUI" classes. These classes, like "PatchGui" or "EPhysGraph" draw graphs, microscope displays, and buttons for the experimenter to interact with. Routines from interface classes are called from these classes. For example, when an experimenter clicks on "Patch Cell", a method from the patch interface class handles moving the pipette to the correct location, checking for resistance changes, etc. GUI classes can also sometimes interact with device classes directly, for example when pulling frames from the camera. This is illustrated as the gray dashed line in Figure 3.1

3.5 Calibration Procedures

To achieve good endpoint accuracy, a linear calibration matrix relating pixel locations in the image to encoder positions in the manipulator and stage must be constructed.

3.5.1 Stage Calibration

To facilitate automation, it is necessary to center the microscope field-of-view on a particular portion of the sample. Unfortunately, a region of interest on the sample is known in pixel coordinates of the image, while centering this area requires a setpoint in micrometers as read by the motorized axis encoders.

To create a transformation matrix between these two coordinate frames, we develop a simple calibration procedure. First, the stage is moved diagonally 500 microns, or about one frame width. Over this movement, images from the microscope and associated encoder positions are recorded.

The total vector of movement, in pixels, can then be calculated for each frame. We employ Lucas–Kanade optical flow [30], a computationally efficient and widely used ap-

proach for recovering motion from a series of images. A visualization of the results are shown in Figure 3.2. From the left to right, the stage moves diagonally up and to the right. The cumulative motion vector from optical flow is overlayed as 3 blue lines. Notice how the upper endpoint of the lines stays in the same pixel location in the image, while the lower endpoint stays in the same image-relative position.

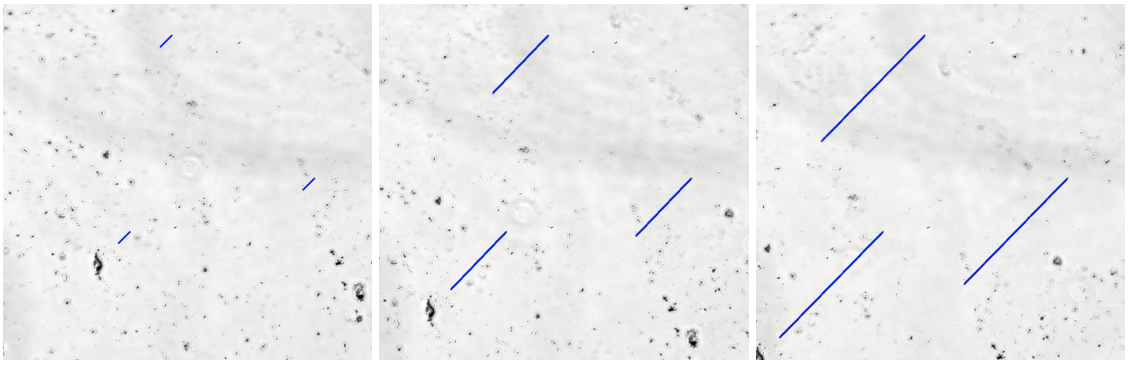


Figure 3.2: Visualization of Optical Flow for Stage Calibration.

Now, that a list of vector movements in pixel and encoder coordinates is known, a homogeneous transformation matrix can be produced two convert between these two coordinate systems, via RANSAC [31] (or, alternatively, a least squares approach). Existing methods to produce this matrix are present in common Python libraries, like OpenCV [20]. These methods attempt to minimize the error in the following equation:

$$\begin{pmatrix} x_{microns} \\ y_{microns} \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{23} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{pixels} \\ y_{pixels} \\ 1 \end{pmatrix}$$

Finally, now that the homogeneous transformation matrix has determined, an experimenter can move the stage to any arbitrary pixel location, allowing for further automation.

3.5.2 Micromanipulator Calibration

The appearance of the pipette tip can vary depending on surrounding conditions like background, lighting, and focus, as well as sharpness of the pipette itself. As such, finding the tip is a difficult problem for classical Computer Vision. As such, I opt to use YoloV8 [32], an anchor-free object detection method created by Ultralytics.

To train a deep learning approach like this a dataset must be curated. Unfortunately, because of differences in imaging parameters (e.g. contrast, zoom level, etc.) between microscopes, no suitable dataset was readily available online. To avoid manually annotating hundreds of pipettes, I devised a synthetic data generation approach, shown in figure Figure 3.3. I start with around 50 so-called "background images", images of cells in varying focus levels with no pipette in frame. Because I am only moving a the microscope stage, these images are very quick to capture. Then, I took 7 images of pipettes in an otherwise empty microscope field. The pipette tip was then manually cropped out and the tip location was annotated.

Next, slight data augmentation is applied to each image set. For background images, a random crop and blurring is applied in an attempt to force the model to ignore erroneous detail in the microscope field. For the set of pipette images, a random affine transform is applied, consisting of a stretch and rotation. Finally, the pipette image is overlayed on the background image to create a synthetic image of a pipette in a microscope field. Note that because the tip location is known before the affine transformation and overlay, using simple trigonometry, the pipette location in the synthetic dataset is also known, thus eliminating the need to annotate data. This procedure was repeated to create dataset of a few hundred images. This dataset was used to train YoloV8-nano, with default hyperparameters.

After training, this model is capable of locating the tip of a pipette in a range of conditions outside its synthetic dataset, an example of which is shown in Figure 3.4. Now, all that is needed for calibration is a list of coordinates in both image pixels and manipulator encoder microns. Unlike the stage, the pipette moves in 3D space and is susceptible to

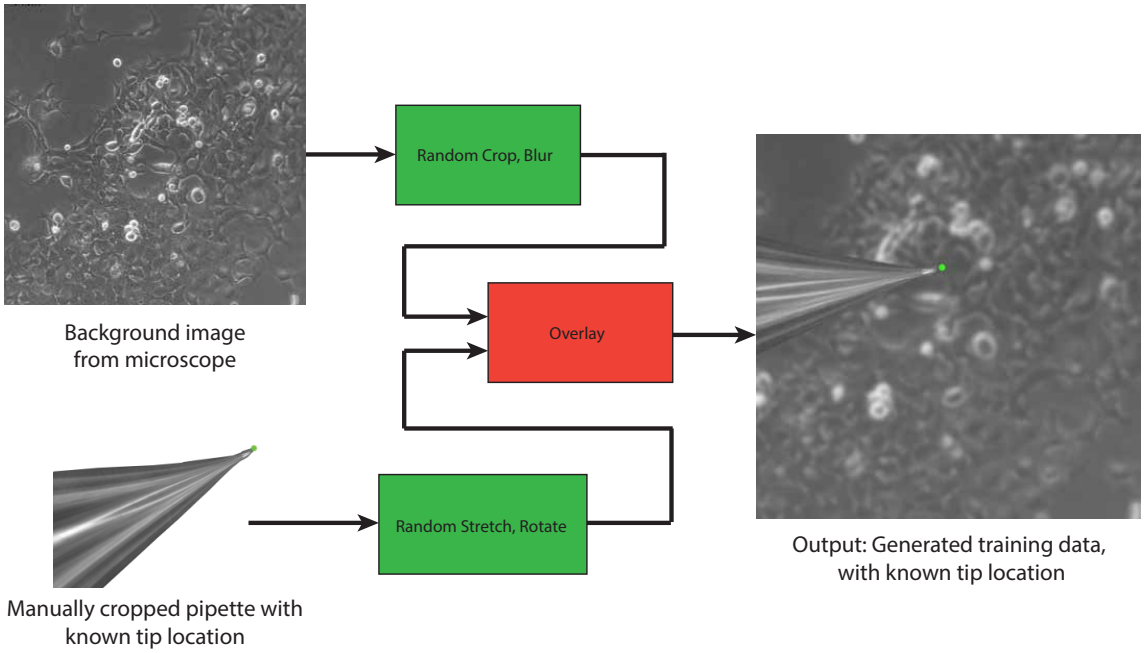


Figure 3.3: The data generation approach for creating a synthetic pipette tip detection dataset.

crashes if a bad movement command is executed, which could delay experiments. To avoid this, I opt to ask the experimenter to move the pipette into a variety of locations in all three dimensions, clicking a "Add Pipette Cal Point" whenever a new location is reached and the pipette is in focus. Whenever this button is pressed, a new pipette position in both coordinate frames is recorded. After around 20 positions are recorded (determined empirically), "Finish Pipette Cal" is pressed and a new transformation matrix is produced. As with stage calibration, RANSAC [31] or Least Squares can be used to solve the following equation:

$$\begin{pmatrix} x_{manip-microns} \\ y_{manip-microns} \\ z_{manip-microns} \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{pixels} \\ y_{pixels} \\ z_{stage-microns} \\ 1 \end{pmatrix}$$

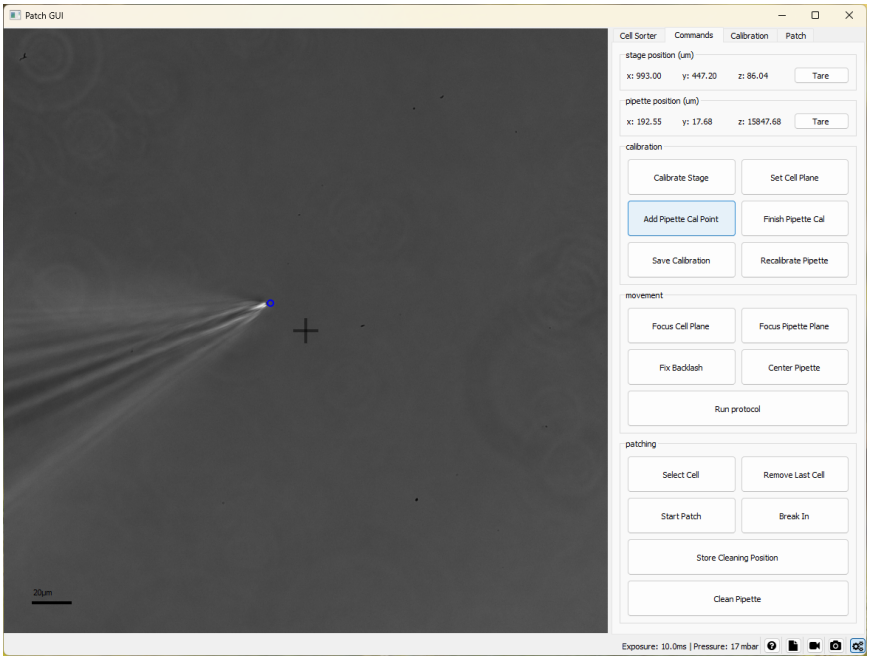


Figure 3.4: Finetuned YoloV8 detecting a pipette tip.

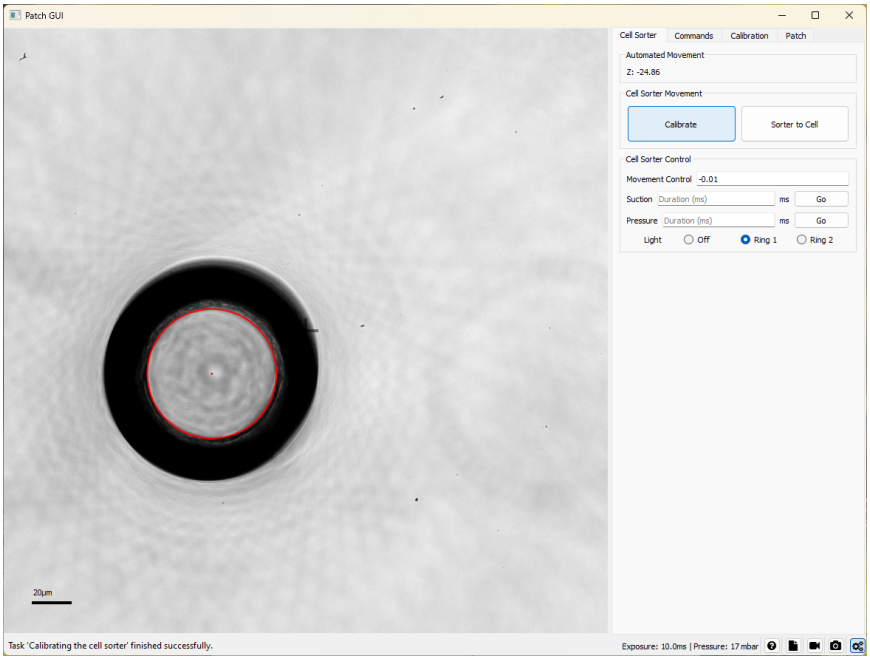


Figure 3.5: Hough Transform detecting the CellSorter pipette in a micrsocope image.

3.5.3 CellSorter Calibration

As seen in Figure 3.5, the CellSorter's large pipette appears in the image as a near perfect circle. Fortunately, circle detection in images is a simple and well studied problem in Computer Vision. Classical approaches like the Hough transform [33] are both accurate and computationally efficient. For calibration, we ask the experimenter to move the CellSorter to the desired Z-location manually, focus the CellSorter, and click the "Calibrate" button. Because, the CellSorter is rigidly fixed to the stage, we only need a vector offset (in pixels) from the center of the image to the CellSorter pipette. Then, when a user requests the CellSorter to focus on a particular cell, we can simply center the stage on the cell's location in pixels plus the offset to the CellSorter.

3.6 Patch Clamping Procedure

After calibration, all manipulators can be moved with high accuracy to any point on the sample. We leave automatically determining the exact point in the image to patch to future automation work. Instead, we invite an experimenter to click on the point on the sample to patch. A green dot then appears indicating a selected cell. When the "Start Patch" button is pressed, the stage is moved to center the selected point in the image. Then, the pipette is moved 30 micrometers (greater than a diameter of both HEK293 and N2a cells) above the sample and a initial pipette resistance reading is recorded. Next, the pipette is repeatedly lowered in steps of 0.5 micrometers until a bump in resistance of 15% is detected. This is associated with approaching the cell. A slight negative pressure of -10 millibar is exposed to the pipette in an attempt to pull in a portion of the cell membrane and produce a gigaseal. Finally, a few bursts of negative pressure are delivered to break-into the cell. Control is then yielded back to the user to run additional experiments.

CHAPTER 4

PATCH CLAMPING VIDEOGAME

4.1 Preface

The following is a detailed description of work presented as a poster at the 2023 Society for Neuroscience (SfN) conference. The original poster can be found at the following link: https://github.com/NathanMalta/holypipette/blob/videogame/media/videoGameSfnPoster.pdf This project is free and open source, and available to download at the following link: https://github.com/NathanMalta/holypipette/tree/videogame

4.2 Introduction and Motivation

Patch clamping is a well-established technique in electrophysiology for investigating the electrical properties of neurons [1] and other cells [27]. Unfortunately, the equipment required for patch clamping is often large, expensive, and limited in availability. For example, our lab uses a high precision micromanipulator built by the company Sensapex. This manipulator and supporting equipment costs more than $9000 at the time of writing[1]. Other necessary components like an amplifier to accurately measure electrical readings[2], a mechanized microscope stage[3], and a digital pressure control system[4] each cost thousands of dollars. Further, culturing and maintaining cells for experiments requires both significant background knowledge and a dedicated cell culturing space with associated equipment.

This high cost and prerequisite knowledge makes time on physical patch-clamping rigs scarce. Because of this, teaching students about the technique is difficult, as any time being

[1] https://shop.sensapex.com/product/system-of-one-in-vitro-micromanipulator-for-patch-clamp/

[2] https://www.moleculardevices.com/products/axon-patch-clamp-system/amplifiers/axon-instruments-patch-clamp-amplifiers

[3] https://www.scientifica.uk.com/products/scientifica-universal-motorised-stage

[4] https://neuromaticdevices.com/products/

used for training could instead be used for real experiments. These limitations limit the number of people that can be exposed to this important technique. Something like allowing a large undergraduate class to learn the process of the technique, observe its pitfalls like crashing the manipulator, and take electrical readings of cells is far from practical with real equipment.

4.3 Method

To make learning about patch-clamping more accessible, we introduce a user-friendly software simulation, a video game, that emulates the highlights of working with a real patching rig. Compared to real patching, our video game has a significantly lower barrier to entry, as no expensive equipment or prerequisite expertise is required.

4.3.1 Video Game Overview

The software offers a virtual environment that replicates key aspects of patch clamping procedures. As shown in Figure 4.1, the software contains a simulated microscope and micro-manipulator which the user can control to run experiments, using either keyboard controls or and Xbox controller. Additionally, a simulated current response, pressure readout, and pipette resistance are displayed, just like in a real experiment setting. A column of buttons populates the right hand side of the program, allowing for additional tasks: cleaning or replacing the pipette and changing the pressure setpoint.

As shown in Figure 4.2 users can establish gigaseals, break into cells, and view simulated readouts. For example, note the quick growth of resistance to a gigaohm in the resistance graph (bottom left) of Figure 4.2a. In addition, note the quick bursts of negative pressure (middle graph) and spiked 1st order current response in Figure 4.2b.

Further, the software replicates various failure modes encountered during real experiments, like pipette tip clogging, breakage, and variations in resistance between different tips. For example, when a user drives a pipette too low, the tip shatters and it takes on a

blunted appearance, just like in real life. Broken pipettes have a larger tip diameter, and thus allow more ions to flow from the pipette's electrode into the surrounding solution, resulting in a lower resistance reading. This resistance drop is also emulated in the simulation. A broken pipette tip and the associated drop in resistance are presented in Figure 4.3.

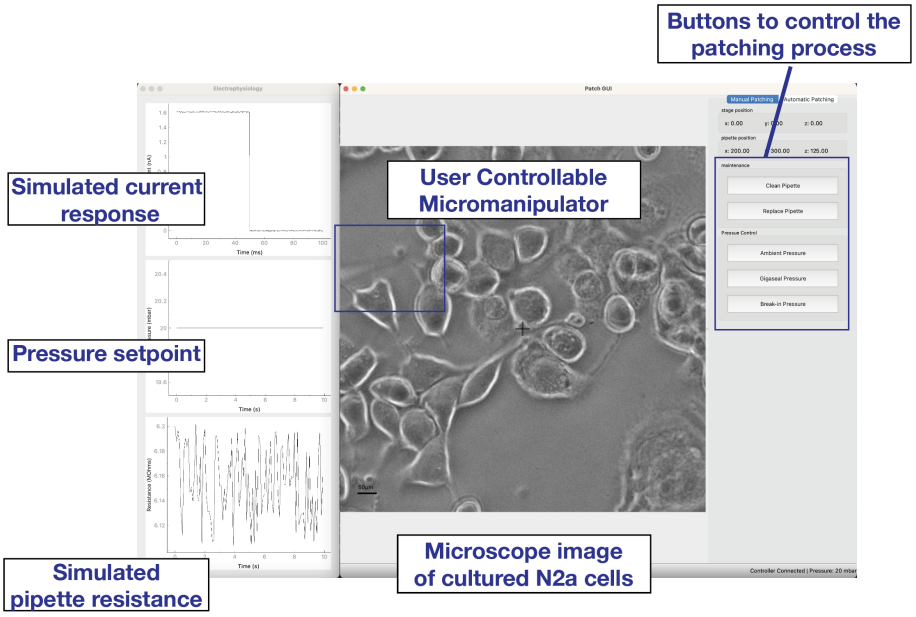
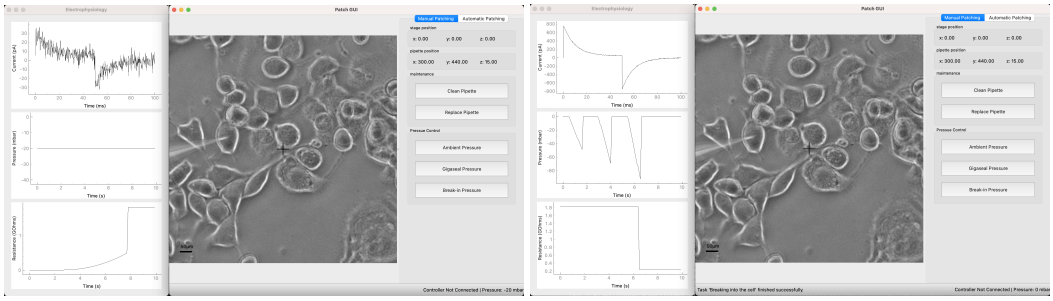


Figure 4.1: An annotated version of the videogame main window.



(a) Gigaseal (b) Break-in

Figure 4.2: The patch-clamping videogame when the user has (a) obtained a gigaseal on a cell and (b) broken into a cell.

4.3.2 Patching Modes

We recreate the two main ways that experimenters interact with a patch clamping setup with two different modes of the software: Automatic and Manual Patching. In the main

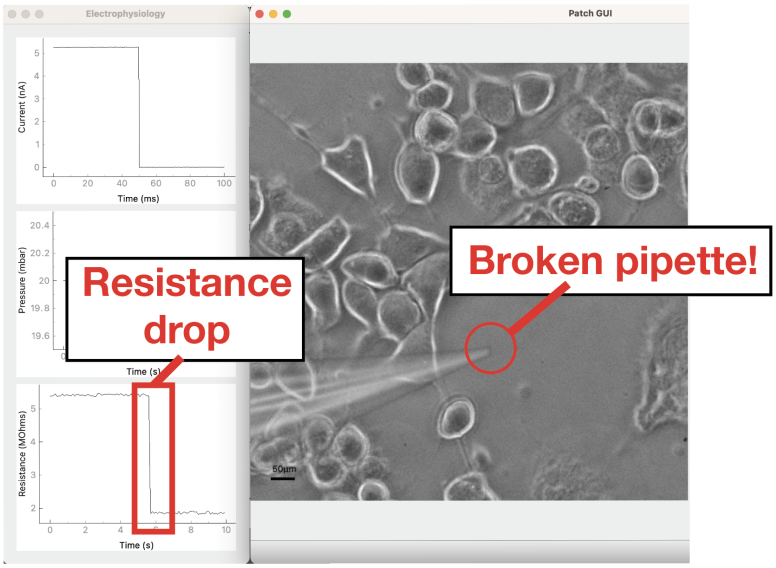
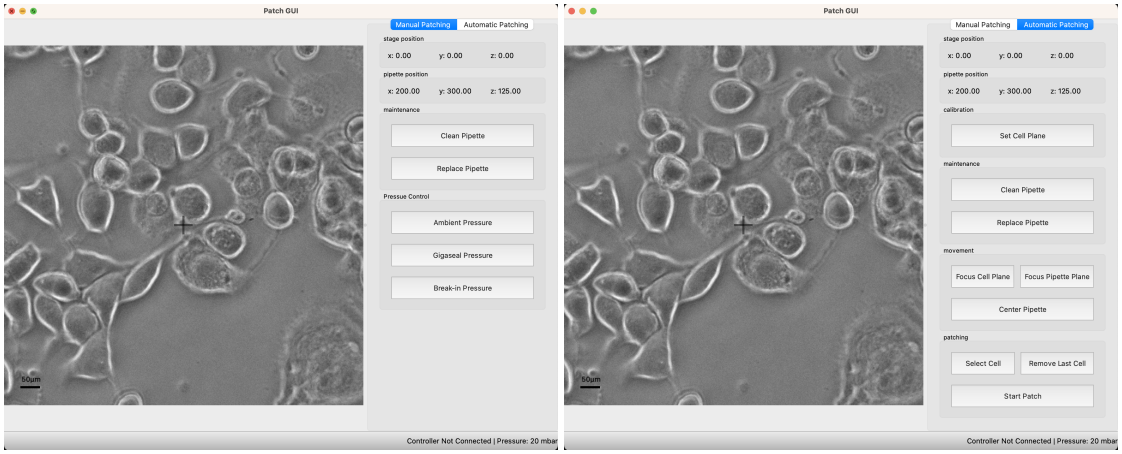


Figure 4.3: The videogame when a user has broken a pipette.

window, we present a tab with controls for each mode, as shown in Figure 4.4.



(a) The Manual Patching Tab (b) The Automated Patching Tab

Figure 4.4: The difference in control buttons between the Manual Patching and Automatic Patching program modes.

Manual Patching

Traditionally, a skilled scientist would use a some sort of controller to position the micromanipulator to a desired location. With the "Manual Patching" mode, we attempt to recreate this technique. When using this mode, the user will manually position the manip-

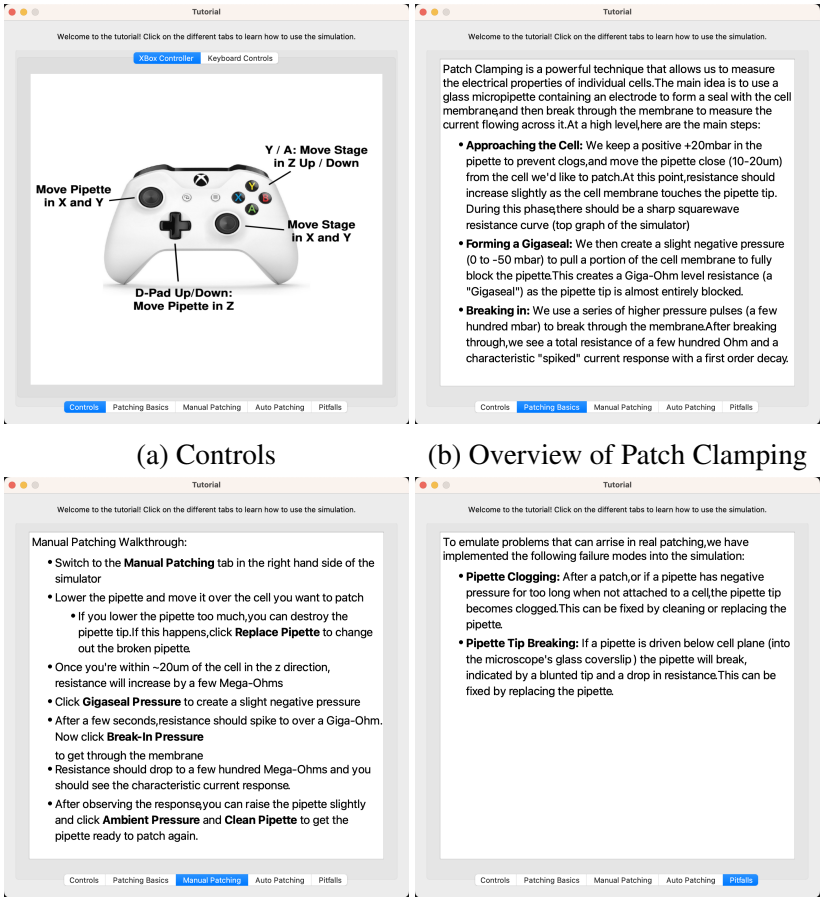
ulator with either their keyboard or an Xbox controller. In this mode, the user must detect changes in pipette resistance as the pipette approaches a cell. Then, the user manually adjusts pressure settings using the buttons on the left to obtain a gigaseal and break-in. The interface when using this mode is shown in Figure 4.4a.

Automatic Patching

Recently, many works attempt to improve throughput of patch clamping via automated positioning of the manipulator both in chapter 3 of this work, and in other works [27, 28]. We implement a simplified variety of this approach into the simulation with an "Automated Patching" mode. The user will simply press the "Select Cell" button and click on a cell in the image. A green circle will then appear indicating the desired cell to patch. Then, the user can click the "Start Patch" button. The software will automatically move the manipulator to the selected cell and begin patching, without any manual movement of either the stage or microscope. The buttons associated with automated patching are shown in Figure 4.4b.

4.3.3 Tutorial Window

To easily instruct users during trials and encourage use of the videogame outside our lab, we produce a tutorial window. This window covers everything related to both theoretical knowledge of patch clamping, and properly running experiments in simulation. The first view (Figure 4.5a) describes how to control the pipette and stage using either an Xbox controller or keyboard. The highlights of Patch Clamping are then described in the next tab, "Patching Basics" (Figure 4.5b). The two subsequent tabs "Manual Patching" (Figure 4.5c) and "Auto Patching" hold step-by-step instructions of running experiments under both modes of the simulator. Finally, the "Pitfalls" section warns about issues a user might experience when performing patch clamping - notably crashing the pipette and the pipette becoming clogged after experiments.



(a) Controls (b) Overview of Patch Clamping

(c) Steps to Patch a Cell (d) Pitfalls of Patch Clamping

Figure 4.5: An overview of the information presented in the tutorial window.

4.4 User Trials

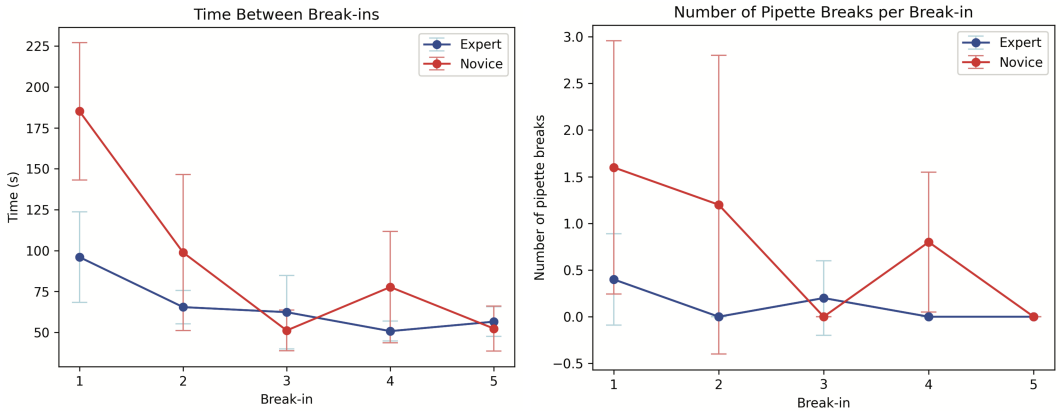
As a preliminary evaluation of our software as a teaching tool, we conducted small scale trials with both “experts” - users who use the technique professionally - and “novices” - users that have limited or no familiarity with the technique in the Manual Patching mode.

4.4.1 Experimental Protocol

We conducted 10 total user trials, split between 5 experts and 5 novices. We provided all users with two minutes to read through the tutorial to familiarize themselves with patch-clamping and the controls of the video game. Then, users are instructed to obtain break-ins of 5 different cells as quickly as possible. Each user spent around 10 minutes using the software.

The software automatically logged important events and timestamps over the course of the study. Logged events included resistance increases during cell approach, obtaining gigaseals, breaking into cells, as well as any accidental crashes that broke the pipette.

4.4.2 Results and Discussion



(a) Time to Break-in (b) Break-in

Figure 4.6: Quantified telemetry from preliminary user trials.

From telemetry data, improvements become apparent after using the software. As

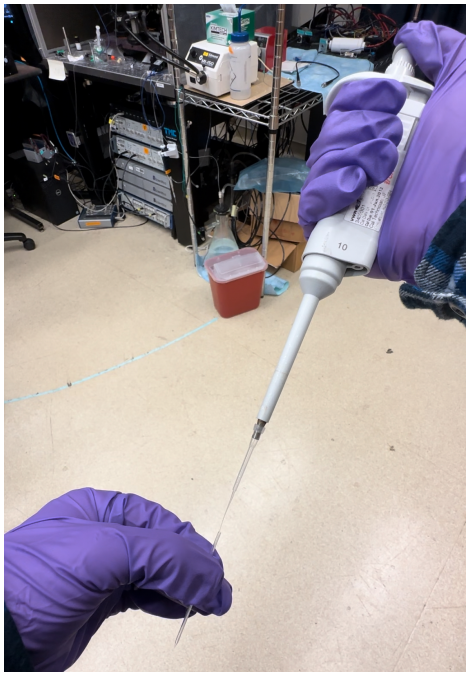
shown in Figure 4.6a, there is a clear reduction in time between break-ins for both groups, alluding to increased familiarity with the software and technique. This is especially apparent for novice users whose time approximately halved between first and fifth trials. A similar trend can be seen for pipette breaks in Figure 4.6b. Both groups broke fewer pipettes on average in later trials, with a more significant improvement observed for novice users. These outcomes highlight the software's potential to streamline training, emphasizing the videogame's practical importance.

4.5 Limitations

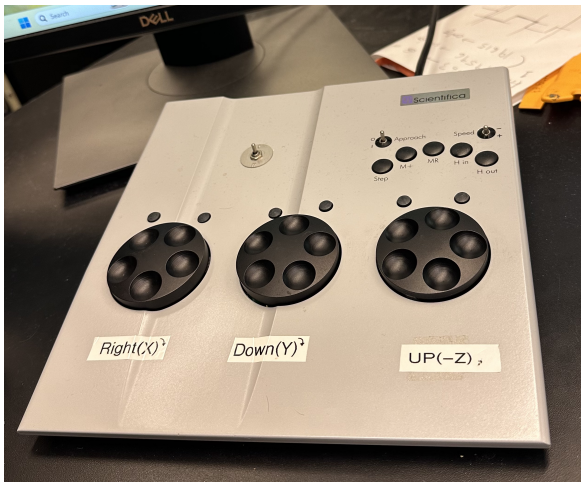
Some aspects of Patch-Clamping are quite difficult to reproduce in software. For example, before being controlled with a manipulator, glass micropipettes must be filled with a solution which allows for ion transfer from the electrode to the cell. Inserting this solution takes a bit of practice and a steady hand. This process is shown in Figure 4.7a.

In addition, some manipulators, like those manufactured by Scientifica and Sensapex, come with proprietary control devices, like the one shown in Figure 4.7b. Getting used to a particular control setup is part of the learning curve of patch clamping. Although any controller can be theoretically connected to the software simulation, interfacing with these proprietary controllers is difficult due to the large variety in production and lack of SDKs to communicate with them.

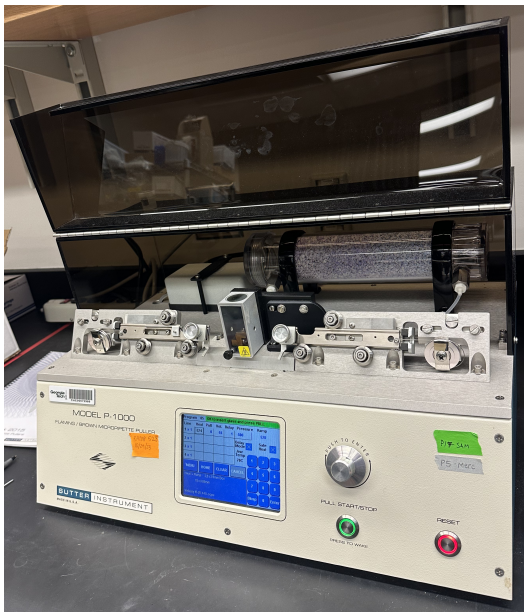
Finally, glass micropipettes must be manufactured prior to use. In my lab, the Precision Biosystems Laboratory (PBL) this manufacturing takes place in house using a specialized machine called a "pipette puller", shown in Figure 4.7c. In this process, a delicate glass tube must be inserted into this machine and a special heating and tensioning protocol is run to form a sharp micropipette. Improper usage could damage the heating element in the machine or result in a micropipette not suitable for experiments. These real-world steps are very difficult to emulate in software and could be overwhelming for novices. As such, the videogame abstracts this process away in a simple "Replace Pipette" button.



(a) Filling micropipettes with solution



(b) Proprietary Manipulator Controllers



(c) Pulling glass micropipettes

Figure 4.7: Examples of Patch Clamping aspects that are difficult to emulate with just software.

4.6 Future Work

After talking to potential users both in our lab and at the Society for Neuroscience conference, a few useful improvements became apparent. In terms of software functionality, it would be very useful to have multiple different cell types available to run experiments on. Notably, simulating microscope behavior looking at brain slices would be particularly useful. Neurons are difficult to find by eye in images . Additionally, as a microscope changes focal planes, different neurons come in and out of view. The ability to train scientists in this area would be a particularly useful addition to the simulator.

In addition to software improvements, we also hope to run additional trials to both benchmark effectiveness and provide more users with familiarity of patch-clamping. Firstly, we aim to provide the video game to an undergraduate class as a homework assignment. This would allow us to gauge the game's impact on students' understanding and application of patch-clamping techniques in a controlled educational setting. Moreover, conducting a broader study among diverse participants will offer a clearer picture of how proficiency develops over extended usage, aiding in refining the software and enhancing its educational value.

4.7 Conclusion

We introduce a software simulation which serves as an accessible and free alternative to a real rig. Unlike a real experimentation setup, our solution be deployed to a large group of people like an undergraduate class, thus empowering students to explore and gain familiarity with patch clamping and bridging the gap between theoretical knowledge and practical experience. From preliminary trials with our software, we see rapid improvement, particularly among inexperienced users.

Although there are some aspects of patch clamping that are difficult to replicate with a software solution, our videogame allows users to experience the high level tasks that

experimenters must perform, without any of the barriers to entry that come with real experimentation. In the future, we aim to continue development of the software, adding new features and conducting larger scale trials to validate the software's efficacy and expose more users to the technique.

REFERENCES

[1] C. R. Landry *et al.*, "Electrophysiological and morphological characterization of single neurons in intact human brain organoids," *Journal of Neuroscience Methods*, vol. 394, p. 109 898, 2023.

[2] C. F. Lewallen *et al.*, "A biologically validated mathematical model for decoding epithelial apical, baso-lateral, and paracellular electrical properties," *American Journal of Physiology-Cell Physiology*, vol. 0, no. 0, null, 0, PMID: 37899750. eprint: https://doi.org/10.1152/ajpcell.00200.2023.

[3] *Cell Culture Basics Handbook*. Waltham, MA: Thermo Fisher Scientific, 2020.

[4] D. Lawlor, *Introduction to Light Microscopy Tips and Tricks for Beginners /*, 1st ed. 2019. Cham: Springer International Publishing, 2019, ISBN: 3-030-05393-8.

[5] *A guide to phase contrast — principles, applications and setup*.

[6] A. Verma, M. Verma, and A. Singh, "Animal tissue culture principles and applications," in *Animal Biotechnology*, Elsevier, 2020, pp. 269–293.

[7] D. Jindal and M. Singh, "Counting of cells," in *Animal Cell Culture: Principles and Practice*. Cham: Springer International Publishing, 2023, pp. 131–145, ISBN: 978-3-031-19485-6.

[8] M. Weigert and U. Schmidt, "Nuclei instance segmentation and classification in histopathology images with stardist," in *2022 IEEE International Symposium on Biomedical Imaging Challenges (ISBIC)*, 2022, pp. 1–4.

[9] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[10] F. A. Guerrero-Pena, P. D. M. Fernandez, T. I. Ren, M. Yui, E. Rothenberg, and A. Cunha, "Multiclass weighted loss for instance segmentation of cluttered cells," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, Oct. 2018.

[11] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: A generalist algorithm for cellular segmentation," *Nature Methods*, vol. 18, no. 1, pp. 100–106, Jan. 2021.

[12] A. Kirillov *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[13] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, *Segment anything in medical images*, 2023. arXiv: 2304.12306 [eess.IV].

[14] K. Zhang and D. Liu, *Customized segment anything model for medical image segmentation*, 2023. arXiv: 2304.13785 [cs.CV].

[15] E. J. Hu *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

[16] A. Dosovitskiy *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV].

[17] C. Edlund *et al.*, "Livecell—a large-scale dataset for label-free live cell segmentation," *Nature Methods*, vol. 18, no. 9, pp. 1038–1045, Sep. 2021.

[18] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, 2018, pp. 265–273.

[19] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[21] S. Pizer, R. Johnston, J. Ericksen, B. Yankaskas, and K. Muller, "Contrast-limited adaptive histogram equalization: Speed and effectiveness," in *[1990] Proceedings of the First Conference on Visualization in Biomedical Computing*, 1990, pp. 337–345.

[22] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," vol. 132, Jan. 1979.

[23] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG].

[25] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[26] R. Xiong *et al.*, *On layer normalization in the transformer architecture*, 2020. arXiv: 2002.04745 [cs.LG].

[27] I. Kolb *et al.*, "PatcherBot: A single-cell electrophysiology robot for adherent cells and brain slices," *J Neural Eng*, vol. 16, no. 4, p. 046 003, Apr. 2019.

[28] S. Stoelzle-Feix, "State-of-the-art automated patch clamp: Heat activation, action potentials, and high throughput in ion channel screening," *Methods Mol Biol*, vol. 1183, pp. 65–80, 2014.

[29] M. Stimberg and R. Brette, *holypipette: A Python package for automated patch-clamp experiments*, version v0.2.1, May 2023.

[30] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81, Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[31] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[32] G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics*, version 8.0.0, Jan. 2023.

[33] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.