2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Computer-Assisted Patch Clamping

Mahdi Azizian, *Student Member, IEEE,* Rajni Patel, *Fellow, IEEE,* Cezar Gavrilovici and Michael Poulter

Abstract—**Patch clamping is an electrophysiological technique that permits the measurement of ion channel activity in many different kinds of cells. Placement of the patch clamp electrodes using micromanipulators is a time consuming and complicated task due to the lack of depth perception of microscope optics and the constrained physical environment. In order to simplify this process, a software platform has been created that permits the user to easily perform not only single electrode recordings but multiple ones. The software platform provides capabilities for automatic positioning of micropipettes in specified locations on the image plane, autofocusing on selected objects, detecting visible micropipettes using image processing techniques, haptic-enabled master slave control of micromanipulators for accurate positioning of electrodes while generating virtual forces to prevent collision between micropipettes, as well as several other novel features which help the user to perform patch clamping more efficiently. The system does not require any changes in the hardware, and uses a fully software-based approach.**

Index Terms—**Patch Clamp Electrophysiology, Microrobot, Micromanipulator, Collision Avoidance, Haptic, Microscope Image Processing**

I. Introduction

Patch clamp electrophysiology is a technique that permits the study of single or multiple ion channels in cells. The technique can be applied to a wide variety of cells, but is especially useful in the study of excitable cells such as neurons and cardiomyocytes [1].

To do patch clamp recordings glass micropipettes having an inner tip diameter of about $0.2\mu m$, and outside tip diameter of about $0.4\mu m$ are used to electrically isolate a membrane surface area or patch which may contain as few as one or two ion channel molecules. The micropipette is pressed against a cell membrane and suction is applied to assist in the formation of a high resistance ($G\Omega$) seal between the glass and the cell membrane. The high resistance of this seal makes it possible to electronically isolate the currents measured across the membrane patch with little competing noise, as well as providing some mechanical stability to the recording.

M. Azizian and R.V. Patel are with the Department of Electrical and Computer Engineering, University of Western Ontario and Canadian Surgical Technologies & Advanced Robotics (CSTAR), London, Ontario, Canada, e-mail: mazizian@uwo.ca, rajni@eng.uwo.ca

R.V. Patel is also with the Department of Surgery, University of Western Ontario, London, Ontario, Canada

C. Gavrilovici and M. Poulter are with the Department of Physiology and Pharmacology, University of Western Ontario and Molecular Brain Research Group, Robarts Research Institute, London, Ontario, Canada, e-mail: cgavrilo@uwo.ca, mpoulter@robarts.ca

This work was supported by an operating grant from the Canadian Institutes of Health Research (M.O. Poulter). The work of M. Azizian and R.V. Patel was partly supported by funds from the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant RGPIN-1345 (R.V. Patel). Financial support for M. Azizian was also provided by an NSERC Alexander Graham Bell Canada Graduate Scholarship. C. Gavrilovici was supported by an Ontario Graduate Student Scholarship.

While this technique has been employed for more than twenty years, it is a difficult and time consuming process, requiring the use of micromanipulators having high precision ($< 0.2\mu m$), no vibration and no drift over long ($>$ 2 hours) periods of time. It is also difficult to train researchers to perform patch clamping experiments and hence the data throughput is extremely low [2]. These difficulties become significant when more than one simultaneous recording is attempted. Only a handful of laboratories routinely do more than one recording at a time, while recordings of three or more at a time are done by only one or two laboratories in the world. One of these difficulties is that each micromanipulator is moved individually and there is no coordination between the micromanipulators; besides, the objective lens is also moved independently which makes it very difficult for the user to follow the manipulator movement when it is out of the field of view. There is also a high possibility of collision between micropipettes. Focusing is done manually and it is like a blind search for an object before it comes close to the field of view which makes the procedure even more difficult and time consuming.

There are several automated patch clamp tools on the market today but almost all of them use a different approach than conventional patch clamping. Farre et al. [2] have listed most of the automated patch clamping systems available in market. Most can only record from cells that approach the micropipettes instead of the conventional way of approaching a cell by a micropipette. Therefore, this approach cannot be used to monitor the electrophysiological activity of cells located in tissue (such as a brain slice). To the best of our knowledge, there is no system available for automated multi-channel patch clamping which could be used on living cells in a tissue, i.e., approaching cells by multiple micropipettes.

This paper describes an approach that we have developed to semi-automate patch clamping, making it faster and easier. It does not require adding any extra specialized hardware or modifications to the commercially available equipment used for patch clamping. We have developed a software platform which is able to control the position of the microscope lens as well as the tip positions of the micropipettes individually. This software platform makes it possible to automatically focus on different objects, move the manipulators or the objective to a desired position selected by the user and avoid collision between the micropipettes. It also provides a graphical user interface (GUI) with the capability of visualization of the process and different control tools. The software also makes it possible to use a haptic device to move any of the micromanipulators or the microscope lens in a master-slave scheme. The haptic device exerts forces on the user's hand to generate forces for the collision avoidance algorithm and act when the micropipette comes close to an obstacle (e.g. another micropipette).

In Section II the experimental setup is described and a list of all the equipment is provided. Calibration of the system is discussed in Section III. The collision avoidance technique is presented in Section IV and Section V describes the haptic-enabled master-slave control. The software architecture is discussed in Section VI. Experimental results are included in Section VII and Section VIII concludes the paper with some suggestions for future work.

II. Experimental Setup

Fig. 1 illustrates the experimental setup including the microscope, lens, camera, micromanipulators and other equipments. The system includes an anti-vibration table, a Olympus BX51WI microscope, a dry and a water immersion objective lens. A 3-DOF micromanipulator (Sutter Instruments MP-285) with $0.04\mu m$ resolution, $2.9mm/sec$ maximum speed and $25.4mm$ traveling range in each direction is used to move the objective lens. Four[1] 3-DOF micromanipulators (Sutter Instruments MPC-200) with $0.0625\mu m$ resolution, $3.0mm/sec$ maximum speed and maximum $25mm$ traveling range is used to carry micropipettes. Micropipettes have a length of around $50.0mm$. A haptic device is used for master-slave control of micropipettes under the microscope. It is worth mentioning that all this equipment (or similar equipment) except the haptic device, is used in conventional patch clamping where the micromanipulators are moved manually using knobs.

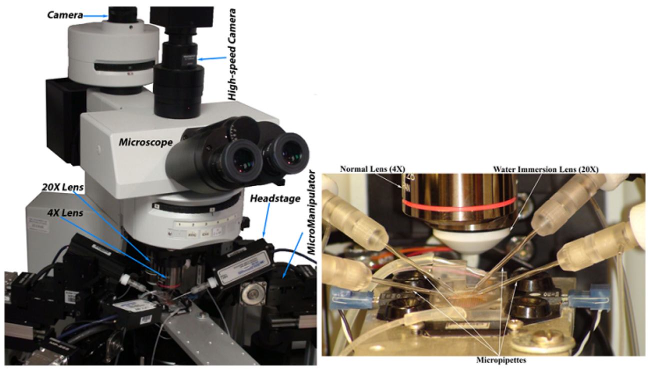

Fig. 1. Experimental setup (Left): all the equipment is installed on an anti-vibration table. The high-speed camera is used for voltage-sensitive fluorescent dye imaging and the image frames captured by the CCD camera are used for visualization and processing purposes in patch clamping. Headstage is a signal conditioner that holds the micropipette. (Right): closer view of micropipettes, lenses and substrate.

III. System Calibration

System calibration includes estimation of critical system parameters including microscope lens and camera parameters, relative coordinates of the micromanipulators and micropipettes with respect to a reference coordinate system and relative coordinates of external obstacles with respect to the reference coordinate system if applicable. $\mathcal{C}^{ref}$ is the reference Cartesian coordinate system which specifies the task space. $\mathcal{C}^{mic}$ is the coordinate system attached to the microscope's microrobot and moves with it. $\mathcal{C}^{man_n}$ is the coordinate system attached to n^{th} micromanipulator and is fixed with respect to $\mathcal{C}^{ref}$. This coordinate system specifies the robot joint space, i.e., its axes are along the micromanipulator's prismatic joints and the micropipette tip is at the origin when the robot is in home position. $\mathcal{C}^{pipette_n}$ is the coordinate system attached to the tool on the n^{th} micromanipulator, its origin is at the same point as $\mathcal{C}^{man_n}$ but its X-axis is along the micropipette's central axis. Fig. 2 illustrates different coordinate systems on a schematic of the setup. We also define an image coordinate system $\mathcal{C}^{image}$ attached to the center of the image and $\mathcal{C}^{haptic}$ attached to the haptic device as shown in Fig. 3. The focal plane is orthogonal to Z^{mic}. The whole setup is installed on an anti-vibration table in such a way that all micromanipulators have parallel Z coordinates, i.e., $\forall n,\ Z_n^{man} || Z^{mic}$.

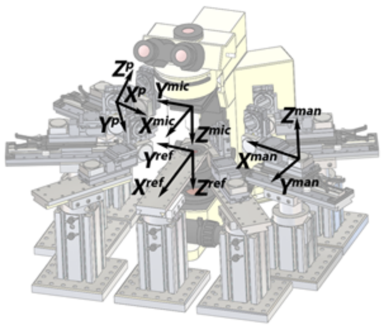

Fig. 2. Coordinate systems on a schematic of the setup: Cartesian coordinate systems $\mathcal{C}^{ref}$, $\mathcal{C}^{mic}$, $\mathcal{C}^{man}$ and $\mathcal{C}^{pipette}(X^p, Y^p, Z^p)$ are shown. (Background image courtesy of *Sutter Instruments* [3])

T_{mic}^{ref} is the transformation that maps $\mathcal{C}^{mic}$ to $\mathcal{C}^{ref}$; therefore:

$$T_{mic}^{ref} = \begin{bmatrix} 1 & 0 & 0 & x_{mic} \\ 0 & 1 & 0 & y_{mic} \\ 0 & 0 & 1 & z_{mic} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where $(x_{mic}, y_{mic}, z_{mic})$ is the position of the microscope's robot with respect to the reference. $\mathcal{C}^{mic}$ is identical to $\mathcal{C}^{ref}$ when the microscope's robot is in its initial position. $T_{man_n}^{ref}$ is the transformation matrix which maps $\mathcal{C}^{man_n}$ to the $\mathcal{C}^{ref}$, therefore $T_{man_n}^{ref} = T_z(\theta_n)T_y(\pi)T_d(d_n)$, where $T_z(\theta_n)$ is a rotation of $\theta_n(rad)$ around z-axis, $T_y(\pi)$ is a $\pi(rad)$ rotation around y-axis and $T_d(d_n)$ is a fixed translation of $d_n = [d_{x_n}\ d_{y_n}\ d_{z_n}]^T$. The transformation matrices are all 4×4 matrices.

The image coordinate system $\mathcal{C}^{image}$ is attached to $\mathcal{C}^{mic}$, i.e., the center of the image lies on the origin of $\mathcal{C}^{mic}$ and they have parallel z-coordinates. Therefore, the mapping between the image and microscope coordinates can be stated as:

$$T_{image}^{mic} = \begin{bmatrix} s_x \cos\theta_I & -s_y \sin\theta_I & 0 & 0 \\ s_x \sin\theta_I & s_y \cos\theta_I & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

where we have used a simple camera model in which s_x and s_y determine the pixel size in x_I and y_I-directions, respectively; And θ_I is a rotation angle to map (x_I, y_I) to (x_{mic}, y_{mic}).

The micropipette coordinate system $\mathcal{C}^{pipette_n}$ has a fixed angle Θ_n around the Y-axis compared to $\mathcal{C}^{man_n}$. Therefore,

[1] can be increased as long as there is physical space

$T^{man_n}_{pipette_n} = T_y(\Theta_n)$ which is a pure rotation without translation around Y-axis. We also define the Jacobian matrix J_n for each robot. This is to relate displacements/velocities in $\mathcal{C}^{image}$ to $\mathcal{C}^{man_n}$; We have $\delta P^{man_n} = J_n \delta P^{image}$ where δP^{man_n} is displacement of an object in $\mathcal{C}^{man_n}$ and δP^{image} is the displacement of the same object in $\mathcal{C}^{image}$. The Jacobian matrix J_n is simply part of the transformation matrix $T^{man_n}_{image}$:

$$T^{man_n}_{image} = T^{man_n}_{ref} T^{ref}_{mic} T^{mic}_{image} = \begin{bmatrix} J_n & D_n \\ 0_{1\times 3} & 1 \end{bmatrix} \quad (3)$$

The system calibration can be performed fully autonomously, semi-autonomously or manually, in each case, the user has to bring the micropipettes to the field of view, one by one, before calibration starts. The software is capable of taking each miropipette to a home position after it is calibrated, to avoid any collisions; It can also bring it back when all of the micropipettes are calibrated. In autonomous mode, there is no user interaction but in semi-autonomous and manual modes, the user is responsible for clicking at micropipette tip. The calibration algorithm is as follows:

1) **Camera Calibration**: estimating T^{mic}_{image} including estimation of the image pixel size and rotation with respect to the base. This includes recording coordinates of a fixed micropipette tip and the objective, when the objective is moved around. The software supports manual, semi-automatic and automatic camera calibration where in automatic mode the objective is moved randomly and the micropipette tip is detected as described in [4]. At least 3 points (not located on same line) are detected and a RANSAC algorithm with a least-squares error estimation is used to find the calibration matrix (refer to Section III-A for details).
2) **Registration of Microrobot Coordinate Systems**: registration of micromanipulator coordinates to a reference coordinate system, i.e., estimating $T^{ref}_{man_n}$ is performed the same way as the camera calibration but this time the objective is fixed and the micropipettes are being moved around. $3D$ position of the objective (P_M), $3D$ location of the micromanipulator (P_R) and the $2D$ location of the micropipette tip in the image (P_I) are recorded at each step. Minimum of 3 points (not on same line) are then used to estimate $T^{ref}_{man_n}$ using RANSAC with least-squares error estimation.
3) **Micropipette Calibration**: estimating $T^{man_n}_{pipette_n}$ is equivalent to estimating the angle Θ_n. The estimation of this angle can be simply provided by reading the protractor on the micromanipulator where it holds the headstage. Although this estimation may not be very accurate, it is only used on special occasions to perform coaxial movement along micropipette axis during patch clamping and small misalignments will not affect the procedure.

A. Parameter Estimation

We have used the *RANdom SAmple Consensus* (RANSAC) algorithm to estimate the parameters of the registration matrices based on the measured data. The RANSAC algorithm can be described as follows ([5], pp. 117-121):

1) Randomly select a minimum number of points required to determine the model parameters (3 in this case).
2) Solve for the parameters of the model (i.e., using the LSE algorithm to estimate the parameters).
3) Determine how many points from the set of all points fit with a predefined tolerance.
4) If the fraction of the number of inliers over the total number of points in the set exceeds a predefined threshold , re-estimate the model parameters using all the identified inliers and terminate.
5) Otherwise, repeat steps 1 through 4 (maximum of N times).

We take u as the probability that any selected data point is an inlier and p the probability that at least one of the sets of random samples does not include an outlier. Then we can simply conclude that $1 - p = (1 - u^m)^N$ [6] where N is the number of iterations and $m = 3$ is the minimum number of samples required for estimation. Now if we choose $p = 0.99$ and $u = 0.9$ (which is considered as the worst case when 10% of the selected points are wrong), then we will have $N = \frac{log(1-p)}{log(1-u^m)}$ which will round up to $N = 4$. In other words the RANSAC algorithm should converge in 4 iterations.

IV. Collision Avoidance

Considering the limited space of operation, there is a high chance of collision among the micropipettes and between micropipettes and fixed obstacles. It is not possible to rely on the human operator to avoid collisions manually because the micropipette tip is very small, tips come very close to each other and they may also collide outside the field of view of the microscope.

Artificial potential fields have been used in several different real-time obstacle avoidance applications for robot manipulators and mobile robots [7]. An artificial potential field (*APF*) algorithm has been implemented to avoid collision among micropipettes and obstacles in the workspace. A virtual force is generated based on this *APF*. The micropipette under control is assumed as an object with positive charge; Other micropipettes, the microscope lens and the bath are also assumed as objects with positive charges. To keep the algorithm computationally efficient, we have assumed charges concentrated at a point. The virtually positive-charged point of an obstacle is the closest point of that obstacle to the micropipette under control. The virtual positive charge on the controlled micropipette, is assumed to be located on the closest point of that micropipette to each obstacle. Although it is also possible to extend *APF*s to perform full navigation including target tracking and collision avoidance at the same time by assigning an opposite charge to the target [8], we have only used *APF*s for collision avoidance to avoid the local minima problem associated with this method.

For each micropipette P_i, $\mathcal{P}_{i,j}$ is the vector connecting the closest points between P_i and P_j, pointing towards P_i. If the vector with minimum length is $\mathcal{P}_{i,k_{min}}$, then:

$$F_i = \gamma_i \sum_{k \in \mathcal{S}_i} \frac{\mathcal{P}_{i,k}}{||\mathcal{P}_{i,k}||^3}, \; \mathcal{S}_i = \{k|||\mathcal{P}_{i,k}|| \leq (1+\epsilon_i)||\mathcal{P}_{i,k_{min}}||\} \tag{4}$$

where γ_i is the repulsion factor which determines the amount of force used for collision avoidance on P$_i$ and $\mathcal{S}_i$ is the set of indices of the closest obstacles to P$_i$, and k_{min} is the index of the closest obstacle to P$_i$. ϵ_i is taken into account to exclude those micropipettes/obstacles which are not closest to P$_i$. To be on the safe side, it is taken as $\epsilon_i = 0.1, \; \forall i$ to cover up to 10% of the distance to closest obstacle around P$_i$. Without a small enough ϵ_i, the superposition of repulsion forces may vanish and a collision may occur. A collision margin is defined by the user in terms of pixels. The collision avoidance force is ignored if the minimum distance is out of this margin.

To mark fixed obstacles (other than micropipettes), the user should move a micropipette close to each obstacle manually and the software would register the coordinates of that obstacle in $\mathcal{C}^{ref}$.

V. Haptic Interface

A haptic device is used to give the user the opportunity to move a specified microrobot while feeling virtual forces which help the user to ensure that the microrobots do not collide with each other or with the environment. This is an intuitive and easy to use interface which accelerates the process of positioning the micropipettes. The coordinates of the haptic device are aligned with the coordinates of the live image on the screen which makes it easy and efficient to train a new user to work with the system. An illustration of the coordinates of the system is given in Fig. 3.

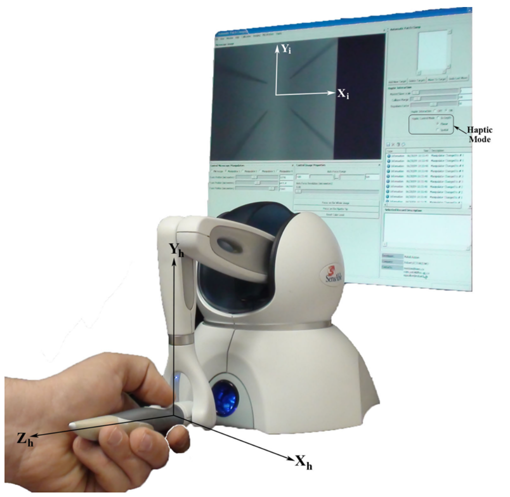

Fig. 3. Haptics-enabled control: the coordinates of the haptic device are aligned with those of the microscope in such a way that X_h, Y_h are matched with X_i, Y_i, and Z_h is matched with the depth. There are four modes defined for master/slave control: (a) In-depth motion, where the motion of the haptic device is limited along Z_h; (b) planar motion in a plane orthogonal to Z_h, the corresponding micropipette moves only in the image plane; (c) 3D motion, where the user has full control over the corresponding micropipette; and (d) coaxial motion where Z_h is mapped on to the X-axis in $\mathcal{C}^{pipette_n}$, activated by a switch on the haptic device while modes (a), (b) and (c) are activated through the GUI.

A. Master-Slave Control

The objective lens is moved by an MP-285 microrobot and the micropipettes are mounted on MPC-200 microrobots. All of these microrobots have a closed architecture controller which means that a user-specified control scheme cannot be used. In a robot with a closed architecture controller, it is not possible to do direct torque or velocity control and it is only possible to issue position or velocity commands and read motor torques and/or forces (e.g. contact forces measured by a force sensor) in defined time intervals with specific protocols through a network connection. When a command is issued, the controller replies with an acknowledgement response after accomplishment the task, or generates an error code when the task cannot be completed. When an inquiry is issued, the controller responds with appropriate data or an error code if the inquiry cannot be completed. Inquiries usually take a certain (fixed) amount of time while control commands can take variable time. For example, if a move command is issued, it takes more time if the requested displacement is longer.

In the system considered here, the closed-architecture robot takes position commands and responds to inquiries on the position of the robot. Each inquiry for the position of the robot takes $T_i = 15.5ms$ and each *move command* takes approximately $T_c(ms)$ to complete, where:

$$T_c = \begin{cases} T_0, \text{ when } ||d|| \leq ||d_0|| \\ T_0 + \frac{||d||-||d_0||}{v}, \text{ when } ||d|| > ||d_0|| \end{cases} \tag{5}$$

where $T_0 = 64.0ms$ is the minimum delay, $||d||$ is the displacement, $||d_0||$ is the maximum displacement which is done whithin T_0 and $v = 3000\mu m/ms$ is the constant speed of the robot.

Therefore, each iteration of the control loop takes at least $\min(T_c + T_i) = 79.5ms$ for the MPC-200. Adding a delay of $33.4ms$ for capturing an image frame and assuming the calculation of collision forces is performed in $1.0ms$, the minimum total delay would be $T_d = 113.9ms$ which gives us a maximum bandwidth of $8.78Hz$ on the slave side. Although large delays in master-slave systems might create serious instability problems [9], [10], the amount of delay in our case lies within the human reaction time and is not expected to create instabilities. Brooks [11] has reported a minimum bandwidth of $3.9Hz$ and a maximum desired bandwidth of $9.7Hz$ (according to $100ms$ human reaction time) based on the consensus of teleoperation experts. We do not expect any stability problems resulting from the inherent delays of the closed architecture controllers, because:

- There is no significant extra delay induced by the communication channel between the master and the slave (i.e., master and slave controllers running on the same PC)
- The slave bandwidth ($8.78Hz$) is within the desired range reported in [11]. It is actually close to the maximum desired bandwidth ($9.7Hz$).
- The environment model, including the artificial potential field used for collision avoidance is passive, i.e., there is no active component in the environment.

A block diagram of the master-slave control system is shown in Fig. 4. Experimental results of master-slave tracking are reported in Section VII-B.

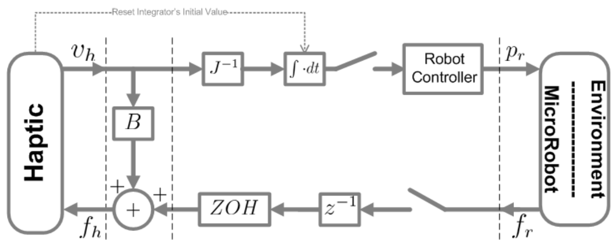

Fig. 4. Master-Slave control architecture: J^{-1} is the Jacobian inverse as defined in Section III; it is the same for the water immersion lens with a scale modification to compensate for the change in pixel size.

VI. Software Architecture

The developed system involves several hardware access routines as well as user interaction and visualization processes; Hence it is very important to use a versatile software design that avoids any collision among hardware access routines, provides real-time visual feedback to the user and at the same time incorporate user commands during real-time processes. As an example, when the Haptic mode is enabled, the user interacts with the software through the haptic device and the GUI at the same time. The haptic device controller, microrobot control and visualization and rendering are running on the same system and accessing the same resources. We have developed an event-driven multi-threaded software environment that handles all of the required tasks and takes care of mutual access problems and real-time coordination among different threads. The software has been developed and tested on a PC with Intel Core 2 Quad Q9650, $3.0GHz$ CPU, $4GB$ of RAM and NVIDIA GeForce GTX285 Graphics Adapter running *WinXP Professional*.

On the other hand, this software has different operation modes, including the autofocusing mode, different haptic interaction modes, the calibration mode, etc. To have coordination among different operation modes, we have used a finite state-machine architecture which prevents any confusion among the different modes and tasks. Each program thread may have a different functionality at each different state.

VII. Experimental Results

The procedure of patch clamping with the assistance of our developed platform includes: (a) system calibration using the dry objective; (b) locating the region of the slice which contains appropriate cells; (c) automatically placing the micropipettes to user specified locations by using registration information or the haptic device or a combination of both; (d) changing the objective to water immersion lens; (e) bringing the objective and micropipettes down to the slice using the software (which has several different modes to do this); and (f) using the haptic device to do patch clamping. The results for the algorithms which are used in this procedure are given below.

We have developed an autofocusing algorithm which moves the objective or a micropipette up and down to maximize a sharpness measure of the image [4]. A description of this algorithm has been omitted for lack of space. A precision of $1.445\mu m$ has been achieved for focusing.

A. Calibration Error

Due to the limited pixel size of the microscope image, the error caused by the user while clicking on the micropipette tip (or automatic tip detection error), the inherent inaccuracy of the micromanipulators, imperfection of the lenses and vibrations of the testbed, there is some error in calibration and registration. We have proposed a method to evaluate this error and the results of the evaluation are reported in this paper.

A micropipette is moved on a 21×15 grid in the field of view, at each node the micropipette tip is detected using the tip detection algorithm. The error between the detected tip point and the tip point calculated using T_{man}^{image} is calculated at each point. The errors on this set of 315 points are then processed to evaluate the calibration/registration algorithm. The results are shown in Table. I. The average square root error in X and Y directions ($mean(\sqrt{e_x^2 + e_y^2})$) is estimated to be 2.5093 pixels.

$mean(e_x)$	$\approx 10^{-12}$	$mean(e_y)$	$\approx 10^{-12}$
$mean(\|e_x\|)$	1.8059	$mean(\|e_y\|)$	1.5346

TABLE I
Calibration Error in Pixels

B. Master-Slave Control

Master-slave control of a micropipette in proximity of another micropipette is performed for an arbitrary path selected by the user; In other words, the user is able to move each micropipette around while the virtual collision avoidance forces are reflected to user's hand and prevent him/her from approaching obstacles. Fig. 5 represents the path followed by a micropipette during an actual experiment. The micropipette under control is shown at the start of the path along with a fixed obstacle which is another micropipette in this case. The collision avoidance forces are illustrated in Fig. 5.

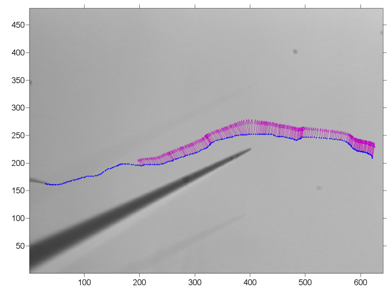

Fig. 5. Collision avoidance forces are shown as arrows. The length and orientation of each arrow shows the magnitude and orientation of the repulsion force at each point on the trajectory.

The master-slave tracking results and the forces applied to the user's hand are shown in Fig. 6. The results show very good tracking in the X and Y directions except that an inherent delay of around $100msec$ is observed. This is due to the closed architecture of the micromanipulator's controller. The slave does not follow the master along the Z direction as observed in Fig. 6 because the system is in planar control mode which is supposed to keep the micropipette in the focal plane. The force F_{h_z} is applied to the user's hand to limit the haptic device motion in a plane.

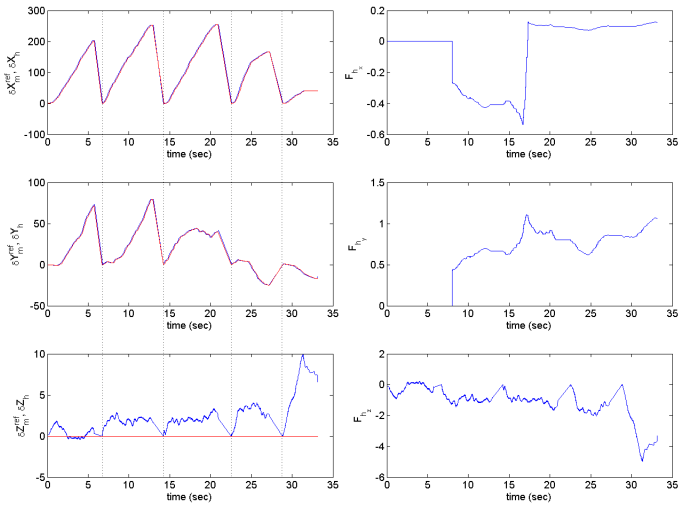

Fig. 6. Master slave tracking results: the left column shows master (blue) vs. slave (red) displacements in the X, Y and Z directions; the right column shows the haptic force (measured in N) in these directions. The master coordinates are in mm while the slave coordinates are in μm. A viscosity factor of $B = .05$ is used.

C. Patch Clamping Results

The patch clamp consists of an electrode inside a glass pipette. The pipette which contains a salt solution resembling the fluid normally found within the cell, is lowered to the cell membrane where a tight seal is formed. When a little suction is applied to the pipette, the **patch** of membrane within the pipette ruptures, permitting access to the whole cell as shown in Fig. 7. The electrode, which is connected to specialized circuitry, can then be used to measure the currents passing through the *ion channels* of the cell (Voltage-clamp protocol as shown in Fig. 8). Furthermore, we can use our electrical circuitry to **clamp** the membrane potential to any desired voltage which is useful when measuring the activity of voltage-dependent channels (Current-clamp protocol as shown in Fig. 9). The oscillations appearing during the recordings were caused by an external noise.

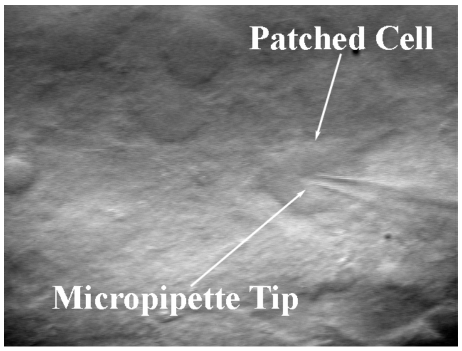

Fig. 7. Image showing the patch-clamped cell in a rat brain slice.

VIII. Conclusion and Future Work

To the best of our knowledge, this is the first computer-integrated robot-assisted patch clamping system that is able to work with existing patch clamp setups without adding any new hardware or making any physical changes. There is no need for a specialized setup or equipment and no mechanical change is required.

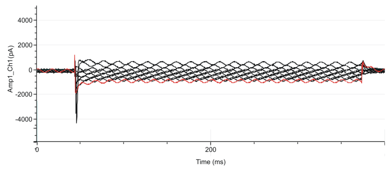

Fig. 8. Voltage-clamp recording showing the sodium current in pA (downwards spikes) vs. time in ms.

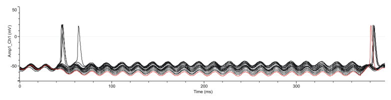

Fig. 9. Current-clamp recording indicating action potentials in mV (upwards spikes) vs. time in ms.

Future work includes the development of a more complicated control scheme to perform patch clamping automatically by visually guided contact force control between a micropipette and a cell is also part of our future research. It would be quite difficult to develop versatile visual contact force estimation and control, due to the unknown shape and size of live brain cells [12].

References

[1] A. Molleman, *Patch Clamping: An Introductory Guide to Patch Clamp Electrophysiology*, 1st ed. Wiley, 2003.

[2] C. Farre, M. George, A. Bruggemann, and N. Fertig, “Ion channel screening - automated patch clamp on the rise,” *Drug Discovery Today: Technologies*, vol. 5, no. 1, pp. e23 – e28, 2008.

[3] “Sutter Instruments: Micromanipulation,” http://www.sutter.com/products/micromanipulation.html.

[4] M. Azizian, R. Patel, C. Gavrilovici, and M. Poulter, “Image processing techniques in computer-assisted patch clamping,” in *Imaging, Manipulation, and Analysis of Biomolecules, Cells, and Tissues VIII*, vol. 7568, no. 13. SPIE, 2010.

[5] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2003.

[6] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Transactions of ACM - Graphics and Image Processing*, vol. 24, no. 6, pp. 381–395, 1981.

[7] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[8] N. Cowan, J. Weingarten, and D. Koditschek, “Visual servoing via navigation functions,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 521–533, 2002.

[9] P. F. Hokayem and M. W. Spong, “Bilateral teleoperation: An historical survey,” *Automatica*, vol. 42, no. 12, pp. 2035–2057, December 2006.

[10] M. Tavakoli, A. Aziminejad, R. Patel, and M. Moallem, “Discrete-time bilateral teleoperation: modelling and stability analysis,” *Control Theory and Applications, IET*, vol. 2, no. 6, pp. 496–512, 2008.

[11] T. L. Brooks, “Telerobotic Response Requirements,” in *IEEE International Conference on Systems, Man and Cybernetics*, Nov 1990, pp. 113–120.

[12] Y. Sun and B. J. Nelson, “Biological cell injection using an autonomous microrobotic system,” *The International Journal of Robotics Research (IJRR)*, vol. 21, no. 10–11, pp. 861–868, 2002.