A software architecture for cell manipulation

Release 0.00

Mahdi Azizian[1,4], Rajni Patel[1,2,4], Cezar Gavrilovici[3,5] and Michael Poulter[3,5]

June 15, 2011

[1]Dept. of Electrical & Computer Engineering, [2]Dept. of Surgery, [3]Dept. of Physiology & Pharmacology
[4]Canadian Surgical Technologies & Advanced Robotics (CSTAR)
[5]Molecular Brain Research Group, Robarts Research Institute
The University of Western Ontario, London, Ontario, Canada

Abstract

A software architecture has been developed for image-guided robot-assisted cell manipulation. The system provides tools for positioning and navigation of multiple micromanipulators under live microscope images. The main application of this system is in multi-electrode patch clamp electrophysiology; however the same architecture can be used for any other type of cell micromanipulation as well as other applications such as micro-assembly.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3283]

Contents

1 Introduction

There are certain applications where multi-robot micromanipulation of cells is required. Patch clamp electrophysiology is one such application, where multiple micro-electrodes are used to study ion channels of excitable cells, e.g., neurons [1, 2]. Glass micropipettes are used to electrically isolate a cell membrane surface area or patch which may contain as few as one or two ion channel molecules. The micropipette tip is pressed against a cell membrane and a negative pressure is used to form a high resistance seal between the glass micropipette tip 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, thereby resulting in a high signal to noise ratio (SNR). This is used to monitor and record the electrophysiological activity of cells which can be used to study different kinds of neurological disorders such as epilepsy and Parkinson's.

Several research groups have developed systems and software for cell manipulation applications [3, 4, 5]. However, very few have reported a versatile architecture for such systems. Ritala et al. [6] developed motion control software which was used for cellular microinjection [4]. Kazanzides et al. [7] have developed a modular software framework (*SAW*: Surgical Assistant Workstation) for rapid development in computer-assisted surgical systems which was adopted by Gao et al. [8] for development of a modular software architecture for biomanipulation based on the *SAW* framework.

We have developed a versatile software-architecture for sub-micron scale manipulation of cells under microscope images [9]. The system supports multiple micromanipulators to control the microscope position as well as the position of tools (e.g. micropipettes). The overall architecture of the system is shown in Figure. 1. Software platforms such as *ViSP* [10] have been developed and used for visual guidance and control in general robotic applications but no such platform has been specifically designed for micromanipulation under microscope image guidance. One of the unique features of the developed architecture is the deployment of microscope images and development of a variety of image processing algorithms for tasks ranging from detection and tracking of micropipette tips [11] to visual servoing of micromanipulators and partial 3D reconstruction of cells [12].

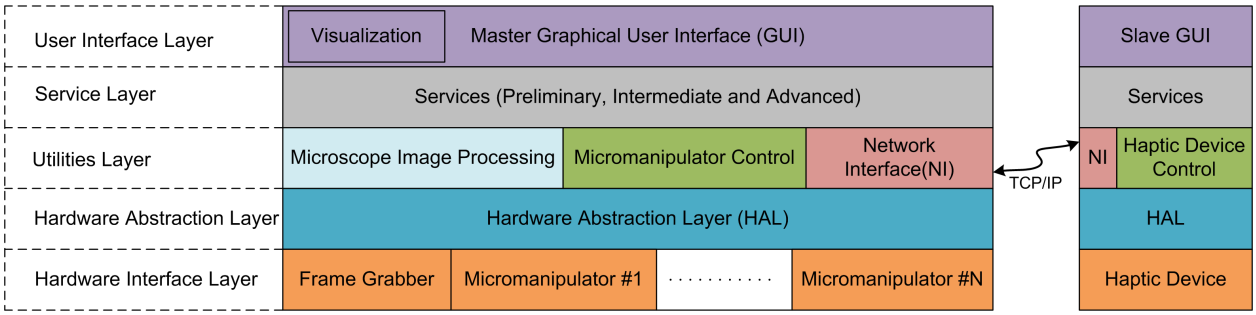

Figure 1: Overall system architecture of the master-slave system for cell micromanipulation.

In Section 2, the experimental setup is shown and its major parts (as used in the *Hardware Interface Layer*) are listed. The architecture of the developed software is discussed in Section 3, where a list of services implemented in the *Service Layer* is given with a brief description for each service (Section 3.1); The graphical user interfaces developed in the *User Interface Layer* are shown in Section 3.2. A brief description of the experiments is included in Section 4 and concluding remarks are given in Section 5.

2 Experimental Setup

The experimental setup is shown in Figure 2. It consists of a microscope, and several micromanipulators all mounted on an anti-vibration table. A micromanipulator ($\mathcal{M}^1$) moves the microscope with respect to a stage which holds the brain tissue slice. A linear objective lens changing mechanism was designed [13] to change between a 4× dry objective lens and a 20× water immersion objective lens using a micromanipulator $\mathcal{M}^2$. Four micromanipulators $\mathcal{M}^3_{1\ldots4}$ are used to move micropipettes in three degrees of freedom (DOF). Each of these 3-DOF manipulators is installed on a 1-DOF linear manipulator $\mathcal{M}^4_{1\ldots4}$ to help retract it for the purpose of tool changes. A list of all these micromanipulators and their specifications is given in Table 1. A *PHANTOM OMNI*® haptic device (Sensable Technologies Inc., Wilmington, MA) was used for master-slave control.

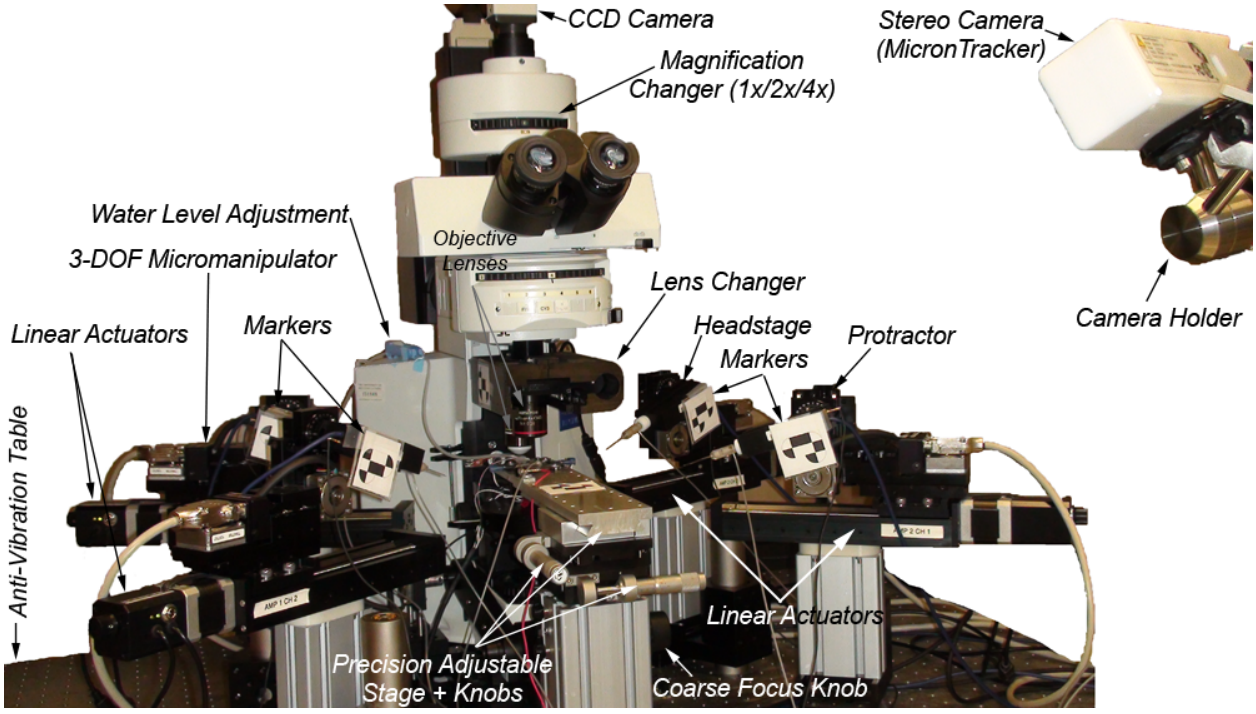

Figure 2: Experimental setup consisting of linear stages ($\mathcal{M}^4_{1\ldots4}$), 3-DOF micromanipulators ($\mathcal{M}^3_{1\ldots4}$), microscope, a CCD camera for capturing microscope images, a custom-designed objective lens changer, a stereo tracking system for coarse registration of micromanipulators, etc.

	$\mathcal{M}^1$	$\mathcal{M}^2$	$\mathcal{M}^3_{1\ldots4}$	$\mathcal{M}^4_{1\ldots4}$
Manufacturer/Model	Sutter/MP-285	Zaber/T-LA60A-S	Sutter/MPC-200	Zaber/T-LSR150A
Manipulated Object	Microscope	Objectives	Micropipettes	$\mathcal{M}^3_{1\ldots4}$
Maximum Speed (mm/sec)	2.9	4.0	4.0	5.0
Resolution (μm)	.04	.05	.0625	.05
Range of Motion (mm) - X	25.4	60	25	150
Range of Motion (mm) - Y	25.4	–	12.5	–
Range of Motion (mm) - Z	25.4	–	25	–
Computer Interface	RS-232	RS-232	USB	RS-232

Table 1: Specifications of micromanipulators used in the experimental setup.

3 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 while incorporating 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 graphical user interface (*GUI*) at the same time. The haptic device controller, visual servoing, micromanipulator control and visualization and rendering are run on the same system and access 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.

On the other side, this software provides different services (Service Layer, Figure 1), a list of most of the provided services is given in Section 3.1. To have coordination among different services and to process user commands, we have used a finite-state machine architecture which prevents any confusion among different tasks. Each program thread may have a different functionality at each different state. The hierarchical structure of the software platform is shown in Figure 3. We have used VTK (Visualization Toolkit) for graphics and visualization purposes and KWWidgets (a *GUI* Toolkit based on VTK) to create the graphical user interfaces. A snapshot of the *GUI* can be seen in Figure 5.

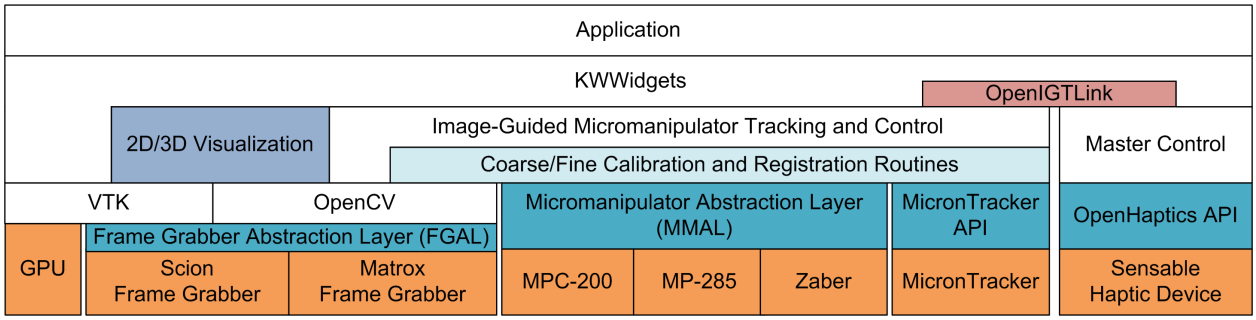

Figure 3: Hierarchy of the software.

The micromanipulator abstraction layer (*MMAL*) was used to provide uniform access to the different types of micromanipulators in the system. The frame grabber abstraction layer (*FGAL*) is built on top of the Matrox Imaging Library (MIL) and the Scion frame grabber API. It grabs an image asynchronously (as soon as a new image is available) and stores images in a time-stamped buffer. The higher level functions can query for availability of new image frames and/or acquire copies of the latest image frame(s) or get direct memory address of the buffer. The *FGAL* also supports simple preprocessing functions including de-noising, masking, flipping, rotating, brightness/contrast adjustment and histogram equalization. The image processing algorithms were developed based on OpenCV libraries. OpenIGTLink was used for master-slave communication.

3.1 Services

The system supports different functionalities which are accessible to the user as *services* through the graphical user interface. The services are categorized into three levels: (i) preliminary services, (ii) intermediate services, and (iii) advanced services. The main services at each level are listed below. The abbreviation *MM* is used instead of micromanipulator.

- Preliminary services:
  - <Move/Absolute/Joint space>: Moving an *MM* to a desired position in its joint space.
  - <Move/Relative/Joint space>: Moving an *MM* to a desired position in its joint space with respect to its current position.
  - <Detect tool tip>: Detecting the position of a tool tip in the microscope image.
  - <Undo Motion/Fast>: Returning the specified *MM* to its previous known position.
  - <Register/Coarse>: Registering each *MM* to the microscope coordinates using the stereo tracking system (sub-milimeter accuracy).
- Intermediate services:
  - <Autofocus/Passive>: Automatic focusing on an object or a region of interest (ROI) in the image by moving the objective up/down.
  - <Autofocus/Active>: Bringing the tool tip to focus, automatically.
  - <Track tool tip>: Tracking the tool tip(s) in real-time in microscope image.
  - <Calibrate/Camera>: Calibrating the microscope images, i.e. finding pixels sizes, skewness and image rotation. It can be automatic, semi-automatic or manual.
  - <Register/Fine>: Registering each *MM* to the microscope coordinates (sub-micron accuracy).
  - <Move/Absolute/Cartesian space>: Moving an *MM* to a desired position in the reference coordinate system.
  - <Move/Relative/Cartesian space>: Moving an *MM* to a desired position with respect to its current position in the reference coordinate system.
- Advanced services:
  - <Coordinated Motion>: All of the tools are moved in the same direction along one of the axes (X,Y or Z), while the objective tracks the same path to keep all the tool tips in focus.
  - <Click & Locate>: Locating the tip of a tool at a point specified by the user on the microscope image using calibration/registration information and/or visual servoing.
  - <Move/Safe>: Moving while avoiding collisions, following a diagonal path (to avoid breaking micropipettes).
  - <Undo Motion/Safe>: Undoing the last motion, avoiding collisions, following a diagonal path (to avoid breaking micropipettes).
  - <Refine calibration/registration>: Refining the calibration/registration information using the tool tip tracking data or user clicks.
  - <Touch water surface>: Moving the objective down, stopping when water/lens contact is detected [13].
  - <Retract Tool>: Retracting tool from the field of view.
  - <Change Tool>: Fully retracting the tool and micromanipulator, in order to change the tool.
  - <Change Objective>: Changing the objective lens, focusing on the same spot.
  - <Master-Slave Control>: The selected *MM* follows the master position.

A finite-state machine (FSM) was developed to organize the software functionalities. A simplified representation of the FSM is shown in Figure 4. The system is in *INIT* state when the software is started. In the *INIT* state, the hardware interfaces are initialized and a diagnostics routine is executed. If there is no fatal error (e.g., image capturing error), the state is changed to *Wait for User Interaction*. In this state, the live 2D microscope image is displayed on the screen and the system goes into idle waiting for the user to request a service. Depending on the requested service, the state changes to the appropriate service which may request for the same level or lower level services. The system returns to the *Wait for User Interaction* state if the requested service is acknowledged with no fatal error. In case of fatal errors, the system throws an exception and the state changes to the *Exception Handling* state. As an example, when a micromanipulator controller becomes unresponsive, the exception handling state asks the user to restart the controller and then the software restarts the connection. In the *END* state, all the allocated resources are released, calibration/registration information and system status are saved and the software is terminated.

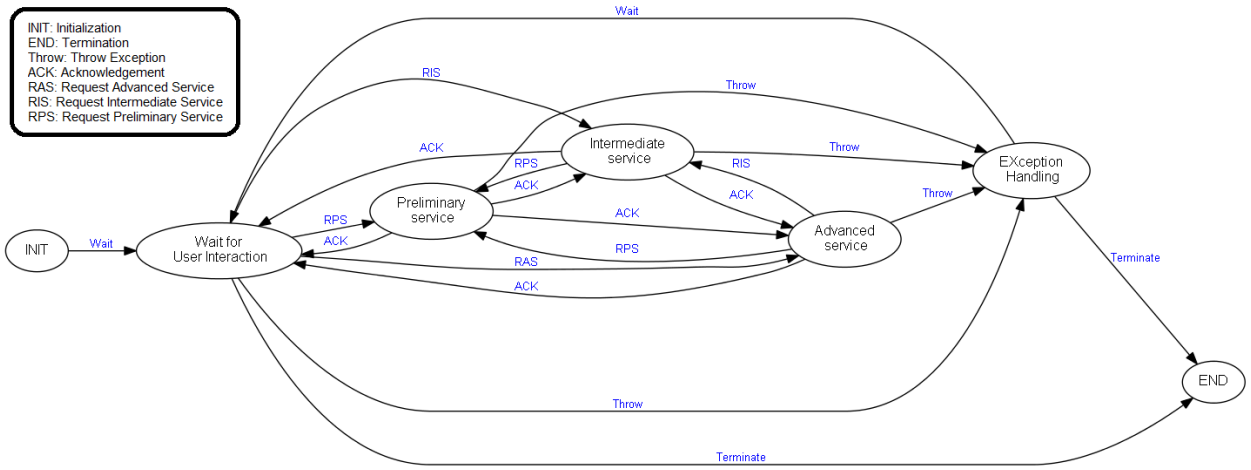

Figure 4: Simplified representation of the finite-state machine (*FSM*). The graph is generated using *Grahviz DOT Language*.

3.2 Graphical User Interface

Figure 5 shows a snapshot of the GUI for the master and slave interfaces. The GUIs were created using KWWidgets as illustrated in the software hierarchy (Figure 3). The master program can run on a different platform as it only communicates with the slave program through OpenIGTLink [14].

4 Experiments

Coronal rat brain slices of 400μm thickness were used for the experiments. The procedure of patch clamping with the assistance of our developed platform includes: (a) system calibration and registration; (b) locating the region of the slice which contains the appropriate cells for the study; (c) autonomous placement of the micropipettes at the user specified locations by using calibration/registration information, visual servoing or master-slave control (using the haptic device) or a combination of these methods; (d) changing the objective to water immersion lens automatically [13]; (e) bringing the objective and micropipettes close to target cells automatically; and (f) using the haptic device to approach each cell membrane individually in order to perform patch clamping. Services at different levels are used to perform these tasks.

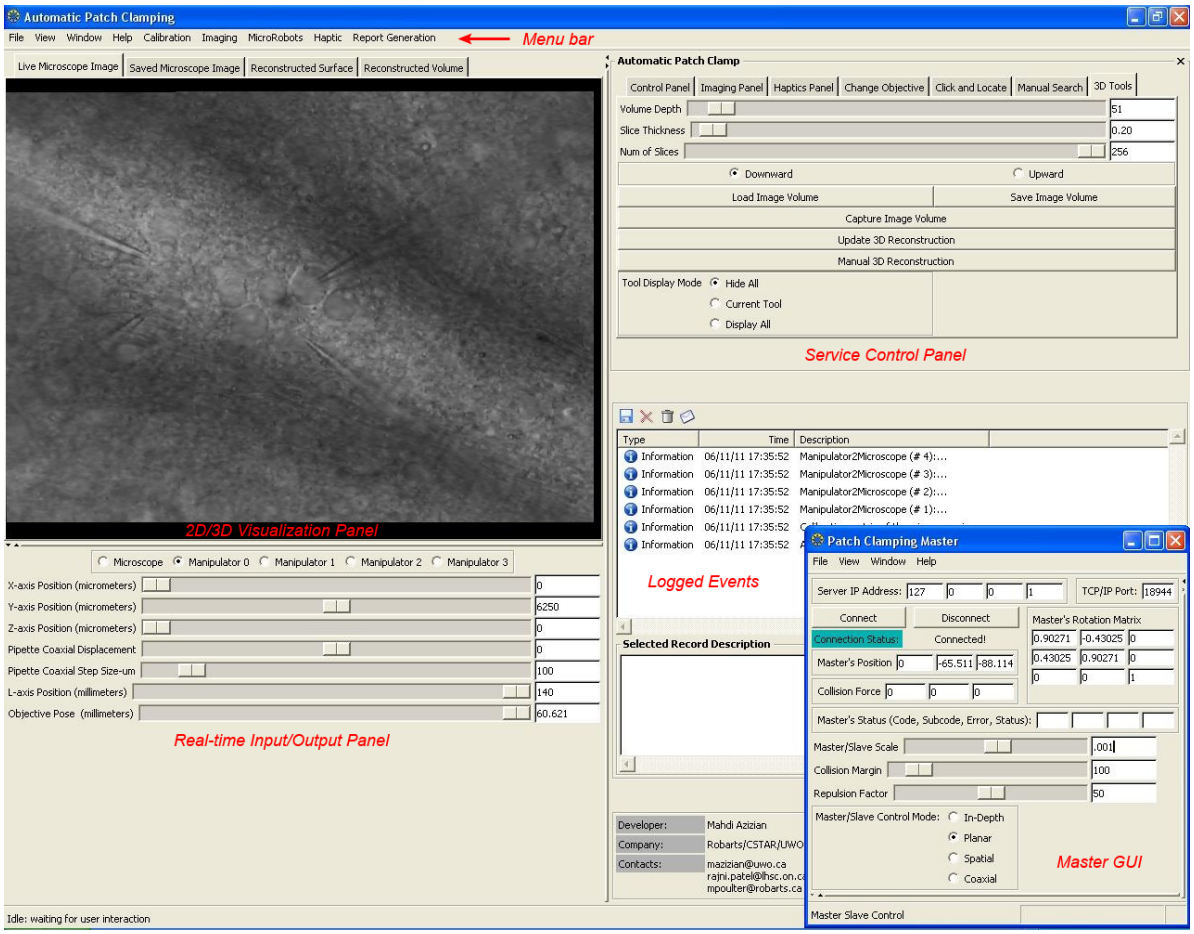

Figure 5: Graphical User Interface (GUI) for master and slave.

5 Conclusion

A modular software framework was developed with the main application in multi-electrode patch clamp electrophysiology. The developed framework can be applied to any type of microscope image-guided micromanipulation system including different applications in cell manipulation, micro-assembly, etc. The major contribution of the developed system is in the application of microscope images in system calibration, registration and control, support for multiple slave micromanipulators with collision avoidance and enhanced 2D/3D visualization. Due to the hierarchical structure of the system, support for new hardware and/or services can be easily added to the system.

Acknowledgements

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.

References

[1] E. Neher and B. Sakmann. Single-channel currents recorded from membrane of denervated frog muscle fibres. *Nature*, 260:799–802, 1976. 1

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

[3] Y. Sun and B.J. Nelson. Biological cell injection using an autonomous microrobotic system. *International Journal of Robotics Research*, 21(10-11):861–868, October 2002. 1

[4] P. Kallio, T. Ritala, M. Lukkari, and S. Kuikka. Injection guidance system for cellular microinjections. *International Journal of Robotics Research*, 26(1-12):1303–1313, November 2007. 1

[5] H.B. Huang, D. Sun, J.K. Mills, and S.H. Cheng. Robotic cell injection system with position and force control: Toward automatic batch biomanipulation. *IEEE Transactions on Robotics*, 25(3):727 –737, june 2009. 1

[6] T. Ritala, P. Kallio, and S. Kuikka. Real-time motion control software for a micromanipulator. In *Proceedings of the IFAC Workshop on Programmable Devices and Embedded Systems*, pages 304–309, 2006. 1

[7] P. Kazanzides, S. DiMaio, A. Deguet, B. Vagvolgyi, M. Balicki, C. Schneider, R. Kumar, A. Jog, B. Itkowitz, C. Hasser, and R. Taylor. The Surgical Assistant Workstation (SAW) in minimally-invasive surgery and microsurgery. In *MICCAI Workshop on Systems and Architectures for Computer Assisted Interventions*, 2010. 1

[8] Y. Gao, R. Taylor, and R. Kumar. A modular architecture for biomanipulation. In *MICCAI Workshop on Systems and Architectures for Computer Assisted Interventions*, 2010. 1

[9] M. Azizian, R. Patel, C. Gavrilovici, and M.O. Poulter. Computer-assisted patch clamping. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4131 –4136, 2010. 1

[10] E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics Automation Magazine*, 12(4):40–52, December 2005. 1

[11] M. Azizian, R. Patel, C. Gavrilovici, and M.O. Poulter. Image processing techniques in computer-assisted patch clamping. In *SPIE Imaging, Manipulation, and Analysis of Biomolecules, Cells, and Tissues VIII*, volume 7568, pages 75681E1–12, 2010. 1

[12] M. Azizian. *Image-Guided Robot-Assisted Techniques with Applications in Minimally Invasive Therapy and Cell Biology*. PhD thesis, The University of Western Ontario, 2011. 1

[13] M. Azizian, R. Patel, C. Gavrilovici, and M.O. Poulter. Image-guided robot-assisted microscope objective lens positioning: Application in patch clamping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010. 2, 3.1, 4

[14] Junichi Tokuda, G.S. Fischer, Xenophon Papademetris, Ziv Yaniv, L. Ibanez, Patrick Cheng, Haiying Liu, Jack Blevins, Jumpei Arata, A.J. Golby, and Others. OpenIGTLink: an open network protocol for image-guided therapy environment. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5(4):423–434, 2009. 3.2