UEMtomaton: A Source-Available Platform to Aid in Start-up of Ultrafast Electron Microscopy Labs

Daniel X. Du, Spencer A. Reisbick, and David J. Flannigan[a)]

AFFILIATION
Department of Chemical Engineering and Materials Science, University of Minnesota, 421 Washington Avenue SE, Minneapolis, MN 55455, USA

[a)]**Author to whom correspondence should be addressed:** flan0076@umn.edu

ABSTRACT

The steady rise in the number of ultrafast electron microscopy (UEM) labs, in addition to the opacity and lack of detailed descriptions of current approaches that would enable point-by-point construction, has created an opportunity for sharing common methods and instrumentation for (for example) automating data acquisition to assist in efficient lab start-up and to learn about common and robust protocols. In the spirit of open sharing of methods, we provide here a description of an entry-level method and user interface (UI) for automating UEM experiments, and we provide access to the source code and scripts (source-available) for ease of implementation or as a starting reference point for those entering or seeking to enter the field (https://github.com/CEMSFlannigan/UEMtomaton/releases/tag/v1.0). Core instrumentation and physical connections in the UEM lab at Minnesota are described. Interface communication schemes consisting of duo server-client pairs between critical components – the optical delay stage and the UEM digital camera – are presented, with emphasis placed on describing the logic and communications sequence designed to conduct automated series acquisitions. An application designed and programmed with C++/CLI as Windows Forms in Microsoft Visual Studio – dubbed UEMtomaton – is also presented. Key to the UI layout is centralization of the automation tasks and establishment of communication within the software rather than by interfacing with each individual workstation. It is our hope that this note provides useful insight for current and future UEM researchers, particularly with respect to generalizability and portability of the approach to emerging labs. We note that while this basic, entry-level approach is certainly not the most sophisticated or comprehensive of those currently in use, we feel there is nevertheless value in clearly communicating a proven straightforward method to hopefully lower the barrier to entry into the field.

Keywords: automation, pump/probe, source-available, femtosecond

Transmission electron microscope (TEM) temporal resolution can be extended to femtosecond (fs) timescales *via* incorporation of optical ports and coupling to short-pulsed lasers in an approach called ultrafast electron microscopy (UEM).[1-6] Accordingly, optical-pump/electron-probe experiments can be conducted in the manner common to other ultrafast techniques.[7-9] Because electrons in the probe packet experience Coulombic interactions that degrade spatiotemporal resolution, UEM experiments are often conducted in a stroboscopic manner so that the number of electrons per packet can be limited.[10,11] Signal at a specific time point is generated with repeated pump/probe cycles and, following readout, the next time point is generated by changing the relative arrival time of the pump and probe at the specimen using a retroreflector mounted on a mechanical stage (*i.e.*, the delay stage), ideally in a randomized fashion to control for systematic errors. The process is repeated until all time points are acquired, and the sequence is assembled post experiment.

Because a single UEM series may be comprised of many individual acquisitions, experiments are ideally automated. This is important because it reduces movement of the researcher in the lab and minimizes total experiment time. This translates into increased stability and reduces the impact of deleterious effects like specimen drift and temperature fluctuations. Automation requires establishing robust communications across key pieces of equipment, especially between the optical delay-stage controller used to set individual time points and the TEM detector used to acquire data. Automation of UEM experiments (and automation of TEM methods in general – of which there are a multitude of examples) is not a new idea and indeed has been a critical (some would argue required) part of labs seeking to obtain femtosecond-picosecond time resolutions since the first sustained efforts emerged.[11-14] However, to our knowledge, what is lacking is a dedicated description of the technical elements of such an approach freely available and published in the literature, including the associated platforms, logic approaches, and access to the source code and scripts. Without this,

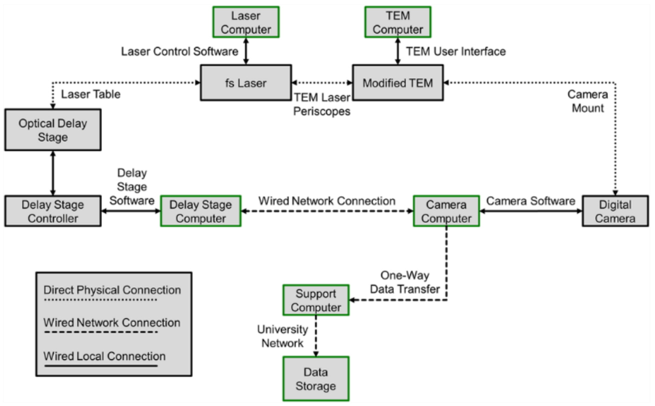


FIG 1. Layout and connections for the UEM system at the University of Minnesota. Green rectangles denote components with a user-controlled interface.

establishment of new labs can be challenging and time consuming, requiring the PI or facility to dedicate significant funds to either developing their own approach (*i.e.*, to reinvent the wheel) or to hiring outside consultants or companies to do the work for them. The latter is particularly unappealing, because the PI or facility, though now equipped with a functioning system, may not have the expertise to modify, update, or properly maintain the system without continuous outside help. Indeed, we view automation as a practical but non-trivial barrier to establishing UEM labs, as gleaned from discussions with our fellow colleagues and newcomers to the field.

Here we describe a basic, entry-level approach to automating UEM experiments, and we have included access to the associated source code and scripts through a GitHub page in the hopes that this will aid in efficient lab establishment or, at a minimum, provide ideas for starting points on constructing new systems. The main goal for this publication then is to communicate one simple approach taken to automating UEM experiments and to provide access to the necessary code and scripts. It is our hope that this will also inspire similar open sharing of technical approaches to UEM instrumentation and automation in the community – which are certainly much more sophisticated and comprehensive than the basic approach described here – so that new labs may benefit from the community's collective efforts and thus come online more rapidly than if the researchers had to develop their own approaches with nothing provided in advance.

The hardware layout and connections scheme is shown in Figure 1. Key pieces of equipment include a modified 200 kV thermionic TEM (Thermo Fisher Tecnai Femto), a fs pulsed laser (Light Conversion PHAROS), a mechanical delay stage (Aerotech PRO165LM) equipped with a broadband hollow retroreflector (Newport UBBR2.5-1UV), and a digital camera (Gatan OneView). Critical elements for automation are the stage controller (Aerotech Soloist CP10-MXU) and the camera; communication between these two components largely constitutes the automation process. The laser and TEM are passive actors. The camera begins acquiring signal once the stage is in position. Upon completion of data collection, the stage is automatically moved to a new random position, and signal for the next time point is acquired. The communications logic is centered on movement and positioning of the stage and on signal acquisition by the camera. The fs laser and the TEM remain on and unchanged during series acquisition.

The two computers connected to the stage and camera are actively used during setup and execution of the automated series acquisition. Further, these two components are in communication during the experiment. The other computers shown in Figure 1 serve as interfaces for completing tasks specific to a particular instrument or process but are not involved in the automation itself. Communication occurs *via* a duo of socket server-client pairs with the Windows Sockets library.[15] One communicates stage information and commands, while the other communicates the camera status and current series acquisition. A user interface (UI) for initiating the automated process, called UEMtomaton, was developed using the C++/CLI Windows Forms package in Microsoft Visual Studio.

Two buffers in the socket server-client pairs are 1,024-character strings that contain relevant passed data at each step. The first character in each dictates the operation required on each side of the system, while the remainder contain numerical data for parameters of the specific operation. The first bit of the stage communication buffer can be set to '1' or '0'. The '0' character passes the length of time the stage computer and the camera computer have been connected, while the '1' character indicates an update to stage position. The first bit of the camera communication

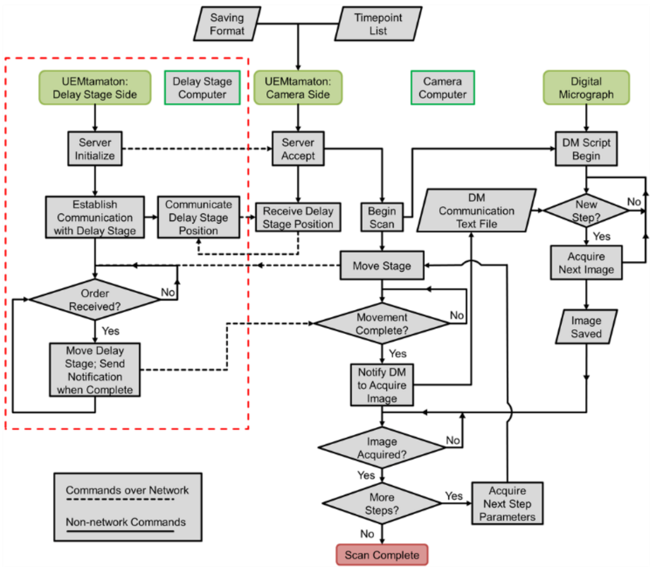
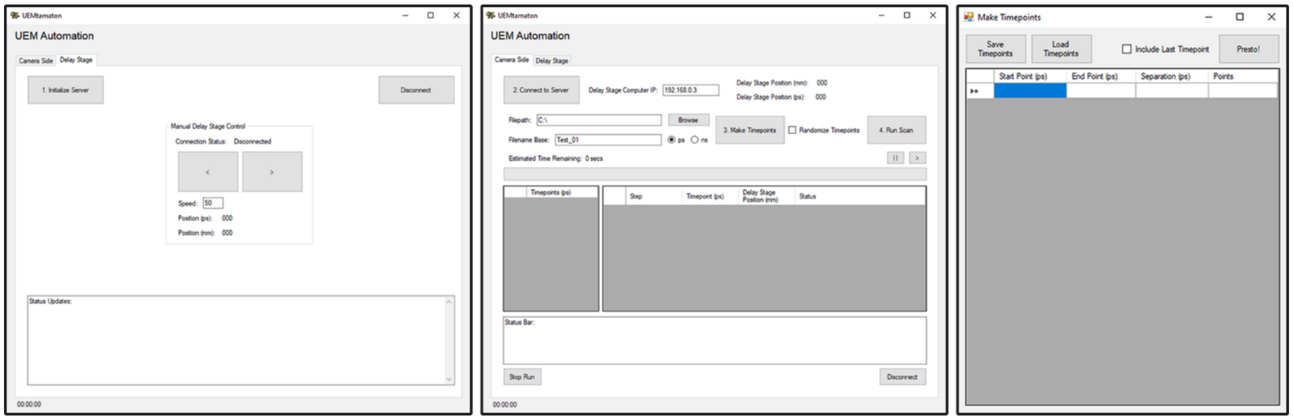


FIG 2. Logic for the automated UEM system. The red dashed rectangle contains the operating procedures of the delay-stage side, while everything else constitutes the camera side. Operations are performed using the "Delay Stage Computer" and the "Camera Computer".

buffer can also be set to '1' or '0'. The '1' character acts as confirmation for completion of the current process, while '0' indicates the process is currently active. Another communication link between the UI and the camera is established *via* a script running in DigitalMicrograph on the camera computer (DM, Gatan). This script is responsible for saving the acquired data. Communication between DM and an external program is accomplished using a text file that dictates the name of the current image in the series, as well as the path and other metadata necessary to determine the action the script is to perform.

Figure 2 shows the logic used to guide programming and automation. The requirements and processes are separated into three components, two of which are operated with UEMtomaton and the other with DM. Two instances of UEMtomaton are initiated, one for controlling the stage and one for controlling the series acquisition. Before starting the acquisition, the two instances of UEMtomaton communicate the current status and position of the stage. Upon initialization, the delay-stage side waits to receive a command indicating the desired position. Upon receipt, the stage moves into position, and UEMtomaton on the camera side begins the data acquisition by sending a command to the DM script. The script also saves the individual acquisitions under a specific path and



Connection Status: Disconnected
Speed: 50
Position (ps): 000
Position (mm): 000

Delay Stage Computer IP: 192.168.0.3
Delay Stage Position (mm): 000
Delay Stage Position (ps): 000
Filepath: C:\
Filename Base: Test_01
[✓ ps] ns
Estimated Time Remaining: 0 secs

FIG. 3. Screenshots of the UEMtomaton user interface (UI). (Left) Delay stage UI. (Center) Camera side UI. (Right) Applet attached to UEMtomaton used to construct time points.

filename, which are defined using the UI prior to initiating the experiment. The process that follows initialization consists of a series of logic and communication steps between the three components that are designed to collect the series items in a randomized fashion.

The UEMtomaton UI (Fig. 3) was designed and programmed to consolidate and centralize tasks. The UI was developed using the C++/CLI Windows Forms package in Microsoft Visual Studio. Visual Studio was chosen for its straightforward placement and adjustment of individual elements, the ability to directly program each in C++/CLI, and the ability to easily communicate with the stage *via* the C++ application programming interface in the Aerotech Soloist software. (Note that the Aerotech software elements are not provided as part of the open-source materials as they are the property of Aerotech.) Windows Forms also allows for multiple asynchronous processes to run in the background, as is needed when communicating stage position while other operations take place. While this simple application is designed around the UEM lab at Minnesota, the basic principles governing functionality are universal, and the source code and logic is immediately extendable to similarly-outfitted labs.

The UI consists of a Delay Stage tab and a Camera Side tab. The delay stage side was designed to minimize user input so all other operations are performed on the camera computer (*i.e.*, the camera side). On the delay-stage side, the process is started by initializing the communication servers between the delay-stage side and the camera side. There is a Disconnect button that will disconnect the stage and the camera computer from the application. In the center there is a panel labeled "Manual Delay Stage Control" for positioning the stage during individual acquisitions. A status bar displays updates on the present position. The date and time appear on the bottom left after initialization.

User operation is concentrated mainly on the camera side. The sequence of operations to initiate automated acquisition are sequentially numbered (Fig. 3, central panel). Following initialization, the camera-side server connection is established. The delay stage IP address is customizable, and one needs to run only two instances of the application and set the IP to "127.0.0.1" if the stage and camera are controlled using the same computer. The stage position is reported in both spatial and temporal units; the temporal indicator is provided so that the user knows the time range being scanned. Below this are locations for entering a file-naming scheme and a path for saving the data. The next step makes use of the "3. Make Timepoints" command, which opens an applet that allows the user to enter delay-stage parameters that comprise the entire automated series (Fig. 3, right panel). The automated series parameters are then entered into the main UEMtomaton program using the "Presto!" button, and the experiment is started using the "4. Run Scan" button under the Camera Side tab. Progress is updated in the table, and the status of each step is displayed in the status bar. The scan can be stopped using the "Stop Run" button.

In summary, we have briefly described here a simple and proven entry-level approach to automating UEM experiments that centralizes operations to a single user interface programmed in an accessible and portable source-available platform. We have interfaced the delay-stage side with the camera side through, among other elements, scripting in DigitalMicrograph. Owing to the goal of combined high spatiotemporal resolutions in UEM experiments, automation that increases efficient collection of data is key, as deleterious effects of user presence and movement, as well as prolonged experiment times, are minimized. It is hoped that this basic, entry-level description, as well as the ability to access the associated source code and scripts through the GitHub site, will serve as both a clear introduction to UEM automation approaches and also as a useful starting point for those entering the field and seeking to establish a new lab.

This material is based on work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Award No. DE-SC0018204. We thank Ben Miller and Bernhard Schaffer for the information regarding image listeners in DM scripting.

SOURCE CODE AND SCRIPTS AVAILABILITY

The source code and scripts are freely available at https://github.com/CEMSFlannigan/UEMtomaton/releases/tag/v1.0. UEMtomaton is copyrighted by the Regents of the University of Minnesota. It can be used only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for their use provided that the copies are not sold or distributed, and are used under the same terms and conditions.

As unestablished research software, this code is provided on an "as is" basis without warranty of any kind, either expressed or implied. The downloading, or executing any part of this software constitutes an implicit agreement to these terms. These terms and conditions are subject to change at any time without prior notice.

REFERENCES

[1]A. H. Zewail, Science **328**, 187 (2010).
[2]D. J. Flannigan and A. H. Zewail, Acc. Chem. Res. **45**, 1828 (2012).
[3]L. Piazza, D. J. Masiel, T. LaGrange, B. W. Reed, B. Barwick, and F. Carbone, Chem. Phys. **423**, 79 (2013).
[4]D. A. Plemmons, P. K. Suri, and D. J. Flannigan, Chem. Mater. **27**, 3178 (2015).
[5]A. Feist, N. Bach, N. R. da Silva, T. Danz, M. Möller, K. E. Priebe, T. Domröse, J. G. Gatzmann, S. Rost, J. Schauss, S. Strauch, R. Bormann, M. Sivis, S. Schäfer, and C. Ropers, Ultramicroscopy **176**, 63 (2017).
[6]C. Zhu, D. Zheng, H. Wang, M. Zhang, Z. Li, S. Sun, P. Xu, H. Tian, Z. Li, H. Yang, and J. Li, Ultramicroscopy **209**, 112887 (2020).
[7]H. Park, Z. Hao, X. Wang, S. Nie, R. Clinite, and J. Cao, Rev. Sci. Instrum. **76**, 083905 (2005).
[8]J. R. Dwyer, C. T. Hebeisen, R. Ernstorfer, M. Harb, V. B. Deyirmenjian, R. E. Jordan, and R. J. D. Miller, Philos. Trans. R. Soc. A **364**, 741 (2006).
[9]S. P. Weathersby, G. Brown, M. Centurion, T. F. Chase, R. Coffee, J. Corbett, J. P. Eichner, J. C. Frisch, A. R. Fry, M. Gühr, N. Hartmann, C. Hast, R. Hettel, R. K. Jobe, E. N. Jongewaard, J. R. Lewandowski, R. K. Li, A. M. Lindenberg, I. Makasyuk, J. E. May, D. McCormick, M. N. Nguyen, A. H. Reid, X. Shen, K. Sokolowski-Tinten, T. Vecchione, S. L. Vetter, J. Wu, J. Yang, H. A. Dürr, and X. J. Wang, Rev. Sci. Instrum. **86**, 073702 (2015).
[10]D. A. Plemmons and D. J. Flannigan, Chem. Phys. Lett. **683**, 186 (2017).
[11]V. A. Lobastov, R. Srinivasan, and A. H. Zewail, Proc. Natl. Acad. Sci. U.S.A. **102**, 7069 (2005).
[12]B. W. Reed, M. R. Armstrong, N. D. Browning, G. H. Campbell, J. E. Evans, T. LaGrange, and D. J. Masiel, Microsc. Microanal. **15**, 272 (2009).
[13]A. Kulovits, J. M. K. Wiezorek, T. LaGrange, B. W. Reed, and G. H. Campbell, Philos. Mag. Lett. **91**, 287 (2011).
[14]A. Feist, K. E. Echternkamp, J. Schauss, S. V. Yalunin, S. Schäfer, and C. Ropers, Nature **521**, 200 (2015).
[15]B. Quinn and D. Shute, *Windows Sockets Network Programming* (Addison-Wesley Professional, Reading, MA, 1995).