JOURNAL OF CHEMICAL EDUCATION

pubs.acs.org/jchemeduc Article

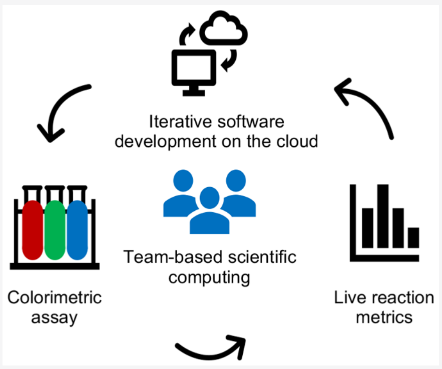
Team-Based Learning for Scientific Computing and Automated Experimentation: Visualization of Colored Reactions

Santiago Vargas, Siavash Zamirpour, Shreya Menon, Arielle Rothman, Florian Häse, Teresa Tamayo-Mendoza, Jonathan Romero, Sukin Sim,* Tim Menke,* and Alán Aspuru-Guzik*

Cite This: *J. Chem. Educ.* 2020, 97, 689−694 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: The increasing integration of software and automation in modern chemical laboratories prompts special emphasis on two important skills in the chemistry classroom. First, students need to learn the technical skills involved in modern scientific computing and automation. Second, applying these techniques in practice requires effective collaboration in teams. This work aims at developing a teaching module to help students gain both skills. In particular, we describe a modular and collaborative approach for introducing undergraduate students to scientific computing in the context of automated and autonomous chemical laboratories. Using online collaboration tools, students work in parallel teams to develop central components of an automated computer vision system that monitors color changes in ongoing chemical reactions. These components include three different aspects: image capture, communication, and data visualization. The image capture team collects and stores the images of the chemical reaction, the communication team processes the images, and the visualization team develops the tools for analyzing the processed image data. Using this educational framework, students built an open-source Python tool called AutoVis that enables the automated tracking of color and intensity changes in a liquid. The software is tested by simulating chemical reactions with dilute solutions of food coloring in water. It is shown that the system reliably tracks color and intensity, providing feedback to the experimentalist and enabling further computational analysis. Over the course of the project, students gain proficiency in scientific computing using Python and collaborate on software development using GitHub. In this way, they learn the role of software in chemical laboratories of the future.


Iterative software development on the cloud
Colorimetric assay
Team-based scientific computing
Live reaction metrics

KEYWORDS: *First-Year Undergraduate/General, Interdisciplinary/Multidisciplinary, Collaborative/Cooperative Learning, Computer-Based Learning, Hands-On Learning/Manipulatives, Problem Solving/Decision Making, Laboratory Computing/Interfacing, Laboratory Equipment/Apparatus, Qualitative Analysis, Reactions*

INTRODUCTION

Knowledge of scientific computing in chemistry is becoming increasingly important as laboratory processes move toward automation, and the Internet of Things rapidly expands into the chemical laboratory environment.[1] This development implies a need to supplement the skills required by the next generation of researchers. Computational tools have been traditionally employed either for simulations of atomic and molecular systems to guide subsequent experiments or for data analysis and modeling after completing an experiment. Innovative approaches to teaching such material have been reported in this journal.[2−4] At the same time, online collaborative tools and teamwork models from the world of software development have been made available.[5,6] Such approaches have inspired scientists to integrate scientific computing into the laboratory setup, remotely interfacing intelligent algorithms and experiments for real-time information and feedback.[7−9] In light of these developments, new opportunities emerge to introduce students to the future of scientific computing in chemistry. Hands-on experience in connecting experimental setups and computational tools constitutes an invaluable skill for the future laboratory environment. Moreover, team-based scientific computing in the classroom better accommodates different undergraduate class levels and technical backgrounds,[10] while training in the use of collaborative code-sharing platforms supports the long-term democratization of science.[11]

The scientific task in this article considers the remotely monitored, automated detection of color changes in a chemical solution, which are simulated by an aqueous solution of food coloring. The choice of using food coloring makes the setup complexity compatible with a class project, while allowing for

Received: June 28, 2019
Revised: January 10, 2020
Published: February 10, 2020

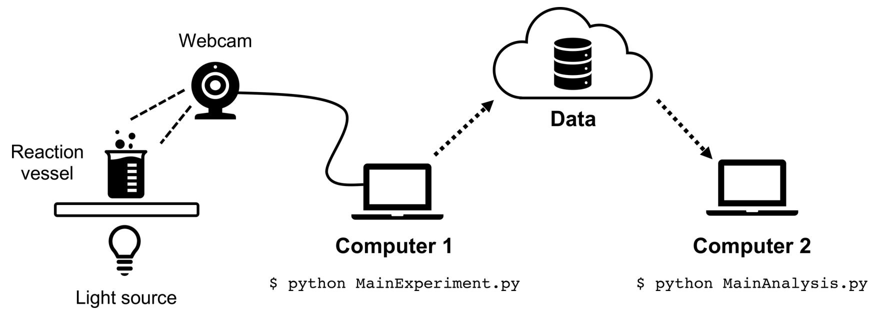


Figure 1. Schematic of the automated color change detection setup. A webcam is used to capture the images of a reaction vessel containing an aqueous solution of food coloring. The lab computer (Computer 1) connected to the webcam detects the solution, and then shares the processed image data with a remote analysis computer (Computer 2) via a cloud service.

straightforward extension to a more elaborate reaction setup when more class time or resources are available. Integrating computer vision systems into standard laboratory environments is an active field of research,[12,13] as many chemical reactions undergo a color change as they progress. Well-known examples in chemistry and biochemistry include the Briggs–Rauscher reaction, acid–base titrations with indicator solutions, and conversion of 3,3′,5,5′-Tetramethylbenzidine (TMB) to a blue product in enzyme-linked immunosorbent assays (ELISA). The automation task was assigned as a final project of an introduction to scientific computing course to nine undergraduate students, with tenure ranging from first to fourth year and previous programming background spanning none to moderate experience. The course combined basic programming lessons with applications to interdisciplinary computational problems drawn from several scientific disciplines, with a focus on chemistry.[14] The purpose of the final project was to extend the application space of the acquired scientific computing skills to the chemistry laboratory setting. It is noted that the authors refined the software and setup after completion of the course in order to increase color detection reliability and structural clarity and thus maximize the utility of the material to the chemical education community.

OBJECTIVES

The overarching goal of the challenge is 2-fold: first, the students are to gain hands-on experience in creating an interface between an experimental setup and scientific computing tools. This constitutes an increasingly important skill for emerging laboratory environments. Second, the project provides experience in collaborative software development, which is an essential expertise in a broad range of fields. In addition, team-based learning has previously been shown to improve students' performance, participation, and perceived learning experience.[15−17] Toward these ends, students independently implement software using the Python[18] programming language and open-source libraries at their discretion and carefully document their code. For example, students on the analysis team gain proficiency in the visualization of images, colors, and data by employing relevant modules, which are further described later in the article. GitHub,[5] a code sharing platform that supports version control, is used for collaborative software development, facilitating a multiteam project. Due to a three-week time limit for the project, students need to communicate effectively and coordinate tasks among teams toward the central goal. Following the completion of the project, the final deliverable is presented and demonstrated to the course teaching staff.

SETUP AND PROCEDURE

Materials for setting up the physical imaging apparatus are readily available and inexpensive (see parts list in the Supporting Information). They include a standard webcam, a light source, and a transparent reaction vessel. The vessel, which serves as the solution container in our setup, is placed over a light source with the webcam situated directly above, capturing images from a bird's-eye view. A sheet of white paper under the vessel diffuses the light and prevents overexposure of the camera. This setup, as illustrated in Figure 1, was shown to reliably provide clear, well-lit images. The lab computer, or Computer 1 in Figure 1, directly connects and communicates with the webcam to automatically detect the circular edge of the vessel containing the solution and processes the relevant pixels of the image (i.e., pixels inside the circular edge). With the processed image data shared through a cloud service, a remote analysis computer, or Computer 2 in Figure 1, checks for new data and provides a data visualization interface for the remote user. For this project, Dropbox[19] is used as the cloud service, but the code is agnostic to any service that allows for syncing data directly to a folder.

Students are divided into three teams as shown in Figure 2, each with a broadly defined task. The image capture team is

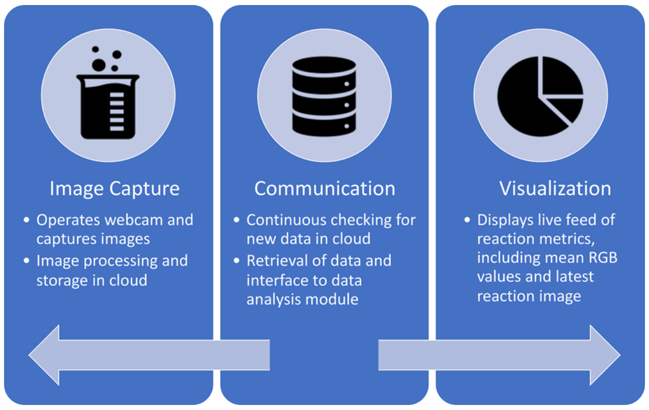


Image Capture
• Operates webcam and captures images
• Image processing and storage in cloud
Communication
• Continuous checking for new data in cloud
• Retrieval of data and interface to data analysis module
Visualization
• Displays live feed of reaction metrics, including mean RGB values and latest reaction image

Figure 2. Team structure. Division of the project into three components, including image capture, communication, and visualization, developed by teams working in parallel.

tasked with realizing an implementation for efficient and cost-effective image capture and storage. This encompasses setting up the imaging apparatus and writing software that instructs the webcam when and how to capture images. The visualization team builds software tools for visualizing data

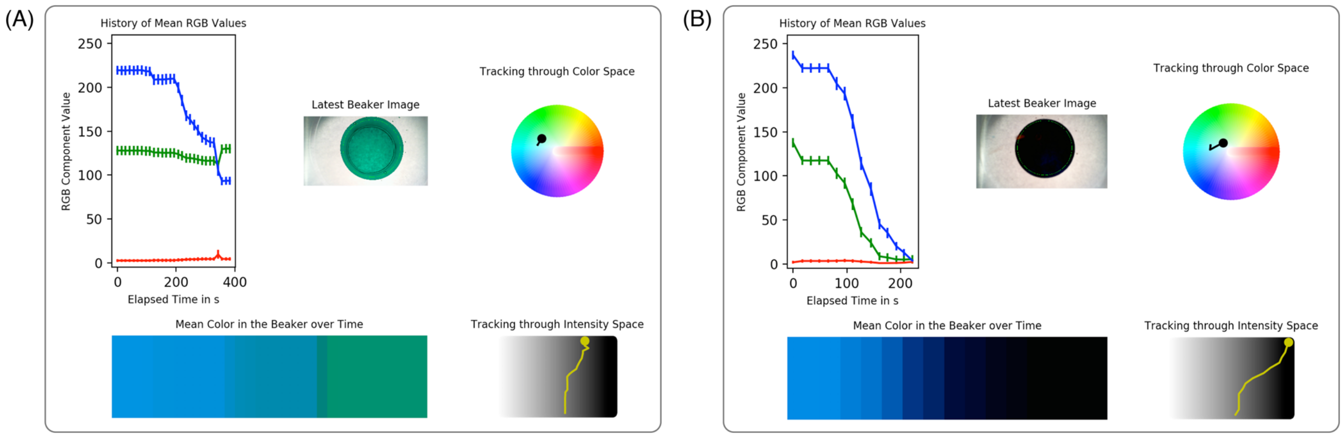

(A)
History of Mean RGB Values
RGB Component Value
Elapsed Time in s
Latest Beaker Image
Tracking through Color Space
Mean Color in the Beaker over Time
Tracking through Intensity Space
(B)
History of Mean RGB Values
RGB Component Value
Elapsed Time in s
Latest Beaker Image
Tracking through Color Space
Mean Color in the Beaker over Time
Tracking through Intensity Space

Figure 3. Screen captures from the AutoVis software demonstrating the analysis of two different experiments with changing solution color. Top left: average RGB values over time. Top middle: most recent webcam image, with detected circle overlaid in green. Top right: detected mean color on a color wheel, with latest measurement marked by a circle. Bottom left: mean color over time. Bottom right: color intensity over time, most recent at the top. (A) Color change from blue to green. (B) Gradual intensity change from blue to black.

gleaned from image processing. Finally, the communication team is responsible for interfacing the image capture and analysis tools by creating a master program that continuously checks for newly generated images and sends them to be analyzed. Within each team, students are responsible for deciding how to implement programs for their tasks using relevant classes, methods, and modules in Python. Among and across teams, students delegate and coordinate tasks, resolve overlaps in team responsibilities, and revise goals and expectations as needed. The teams meet in weekly 4 h coding sessions to encourage clear and continued communication.

Collaborative Code Development Using GitHub

Prior to code development, students learn to navigate and collaborate within a class-wide GitHub repository using their computers' command-line interface. GitHub is a platform to host collaborative projects using git, a version control system. It enables groups of programmers to track changes in software through updates pulled from a central repository. Developers can also push their own changes and updates for others to build upon. For this project, prior to merging their code into a single code base, each team works within a separate subrepository to write, test, and debug their own modules. Finalized code from each team is then merged into the master repository, in which the teams work together to test and debug the interface of their modules, ensuring their compatibility. Throughout the process, basic functionalities in GitHub allow students to work on their code simultaneously while easily resolving code conflicts and maintaining documentation. In the following section, the modules developed by the individual teams are discussed. For each module, we provide its pseudocode, a form of the code that is human readable and abstracted from the actual Python code. The software developed for this work, which we named AutoVis, is made available in a public GitHub repository (see URL in the Supporting Information).

Image Capture Team

The image capture team is tasked to build a script that runs on a local computer using an external webcam to capture images of the reaction vessel at user-specified time intervals and automatically uploads those images to the cloud, where they can be further processed by the communication team. The pseudocode for the hardware-interface function is shown below:

```
ImageCapture function: Take images and store statistics

Require: directory, reaction ID, time interval, duration, camera ID
    Set directory, reaction ID, time interval, duration, camera ID
    take image
    threshold to fit a circle
    for all pixels in circle do
        store mean, variance of each color dimension
    end for
    save statistics for mean, variance in csv file
    save image with overlaid circle
```

In the image capture module, users first specify the duration of their experiment and the interval over which to capture the reaction images. The software uses the OpenCV Python library[20] to take and process images. Alternatively, there are other freely available Python packages students could use for image acquisition, e.g., SimpleCV[21] and pygame,[22] and for image processing and analysis, e.g., scikit-image,[23] PIL (Python Imaging Library),[24] and Mahotas.[25] The image processing involves the detection of the reaction vessel within the original image, which also includes the background. An important step in the vessel detection consists of image thresholding, a method that maps each pixel to black or white based on an intensity threshold value. It then fits the largest possible circle to the black-and-white image based on a polynomial fit to the edges within the image. The circle is then added to the original image, and the combined image is saved. The module also calculates the mean and variance of the RGB values within the detected circle, which is then stored in a file in the comma-separated values (csv) format.

Visualization Team

The analysis and visualization team programs a dashboard with the purpose of providing an intuitive overview of several real-time metrics concerning the reaction. We show its pseudocode below:

```
Dashboard function: display statistics on captured images

Require: means, variances, images
    display latest image with circle overlaid
    add plot of RGB time trace
    add graph of color intensity bar and color wheel
    add history of color intensity
```

We envision that the display could be monitored remotely by an experimentalist who controls an automated laboratory.

The dashboard includes a time trace of mean RGB values, the latest image of the reaction vessel, and the histories of mean colors and intensities. Time traces of deconvoluted RGB channels are useful for monitoring colorimetric reactions, such as the Briggs−Rauscher reaction or ELISA detection introduced above. In addition, providing actual images of the physical setup is necessary to confirm safe conditions; an image of an overturned vessel will immediately cue an experimentalist to terminate the experiment, for example. Time traces of mean colors and intensities through their respective spaces give students an opportunity to explore the efficacy of different formats for data visualization. In some experiments, for example, obvious changes in the plot of mean RGB values may correspond to an unremarkable degree of movement on a color wheel, or vice versa (Figure 3). Finally, a history of the mean colors detected in the reaction vessel over time provides an intuitive glance at the evolution of the reaction. To create the dashboard, students collaborate closely with the image capture and communication teams to establish standards for data structures that are essential for code functionality, efficiency, and robustness. Useful Python packages for implementing the dashboard include matplotlib[26] and seaborn,[27] which offer tools for figures and graphics, and colorsys,[28] which allows for conversions of color data between different color spaces.

Communication Team

The communication team writes a script that runs on a separate computer to monitor the reaction remotely, as shown below in pseudocode:

```
Analysis function

change path to image directory
check if new images have been added to csv
for new images do
    read image statistics from csv
end for
pass on statistics and latest images to dashboard function
update list of processed images
```

This function continually checks the folder in the cloud for new images in the directory created by the image capture team and gathers the timestamps and statistics of the images. Timestamps practically ensure a unique identifier for an image, specifying to a high precision when an image was captured. The script then employs the real-time data analysis tools built by the analysis team to visualize the data.

Executing the Color Detection Workflow

The color detection software is started after setting up the experiment, with the camera positioned above the vessel to capture images of the solution. Two different Python scripts need to be executed, one on the lab computer and the other on the remote analysis computer. The script MainExperiment.py manages the image acquisition and stores the processed images in the cloud folder. It is executed first and runs on the laboratory computer that controls the webcam. After executing, the user will specify the path to the Dropbox directory, a unique experiment label called "reaction ID", the duration of the data acquisition, and the image-taking frequency. Data are stored in the Dropbox directory in a folder named after the reaction ID. The script for analysis and visualization, MainAnalysis.py, is subsequently executed. It is designed to run remotely on a separate computer and only requires access to the cloud folder with the image and csv file data. The user enters the Dropbox directory and reaction ID to start the real-time data analysis and visualization.

■ RESULTS

The two screen captures in Figure 3 show the final output of the dashboard function for two experiments in which the color of the solution is changed by adding a different-colored dye.[29] In Figure 3(A), yellow coloring was added to the blue-colored solution. The top middle image of the dashboard displays the most recent image captured by the webcam, with the detected circle overlaid. The solution is clearly green, which matches the expectation for adding yellow to blue liquid, which is also reflected in the other panels of the dashboard. The top left image of Figure 3(A), "History of Mean RGB Values", shows the red, green, and blue components of the detected color and elucidates the crossing from a blue- to a green-dominated color. The remaining panels of the dashboard visualize the evolution of the color in additional ways by plotting it as a time series, on a color wheel, and in intensity space. Note that the intensity of the color, i.e., a measure of its darkness, does not change much throughout the transition. In Figure 3(B), the evolution from a blue to a black solution is shown. This was achieved by adding a concentrated drop of blue coloring to the solution, darkening the color to the point where it appears black. The top left image of Figure 3(B), "History of Mean RGB Values", shows a decrease of all color components toward zero, corresponding to black in RGB space. In this experiment, as the main effect is a change in the intensity, the "Tracking through Intensity Space" panel is particularly informative.

The data were obtained after iterating over different settings of the background light, food coloring concentration, camera settings, and circle detection settings. A particularly important parameter is the binary threshold value in the image processing, which sets each pixel either to black or white depending on its saturation. This step is performed before finding circles in the image and is especially sensitive to light as well as other objects in the image frame. An empty frame, aside from the vessel itself, and medium-to-high thresholding parameters yielded the best results. In addition, a purely white background with soft lighting conditions created the best image detection environment. Additionally, it was noted that small additions of diluted dye allowed for more gradual color changes and prevented the solution from becoming very dark immediately. Finally, the time between image captures was set to 10−20 s to allow for addition and mixing of the dyes.

In a second iteration of the project, there are several potential improvements to the software and setup that could be considered. For example, it became clear that despite its visual aesthetics, the color wheel shows only marginal changes as the reaction proceeded even for dramatic shifts in the color as perceived by the human eye. It has therefore not served as a useful proxy for color change detection. However, we believe that implementing the color wheel constitutes a learning opportunity for students to explore the usefulness of different data representation techniques. Moreover, the circle detection code could be further improved to set the binary thresholding and other detection parameters automatically, using information obtained from the full image. To improve the current experimental design, another webcam device may be used to capture images of a second reaction vessel containing the same solvent as a blank. This additional information allows the user to more accurately measure the color and intensity change of the solution by subtracting an image of the blank. Alternatively, the transmittance and absorbance can be calculated by taking the ratio of the intensities measured at a specific wavelength. To further automate the procedure, dye additions and stirring could be automated using valves and a magnetic stirrer.

CONCLUSION

In this work, we presented a project to introduce undergraduate students to the use of scientific computing in automating chemical research laboratories. Students worked in three teams to collectively develop a tool to automatically capture and analyze images of colored chemical reactions. In creating this tool, which we named AutoVis, students gained proficiency in programming in Python, collaborating through GitHub, and working in teams to develop software tools for chemistry applications. They learned how the characteristics of the experimental setup inform parameter choices in the software control, such as external lighting conditions influencing the thresholding parameter in the image capture module. In this way, the color change simulation setup exposed students to technical challenges that are of importance in the modern chemistry laboratory. According to their feedback on the course, students from a wide range of programming backgrounds saw large improvements in their scientific computing skills. The teaching staff assessed this improvement by comparing codes written by the students in the first few weeks of the course and during the final project. They noted significant development in (1) the presentation of code (for example, documentation and conciseness of code), (2) advanced usage of Python features (for example, usage of classes versus simple methods), and (3) the ability to use open-source libraries. In the beginning of the course, students were learning to program by filling out missing lines of code from prewritten codes or writing short scripts in Python. By the end of the course, students were writing Python codes from scratch and were comfortably using Python libraries to implement advanced features, for example using the OpenCV library for image acquisition and processing.

Potential future directions for this work include integration of other sensors and parameters to help probe chemical change, such as pH, solubility of certain chemicals, and electrical conductivity. The reliability of color detection could be improved by using a more specialized wavelength sensor. In fact, an additional learning opportunity lies in tasking an additional team with building an inexpensive spectrophotometer as shown by Bogucki et al.[30] From a software standpoint, the results from experiments could be aggregated in central databases to help train and inform better statistical and machine learning models for predicting chemical change. Furthermore, this system could be integrated into a larger chemical control software that automates experimental procedures.[7] For example, a color change detected by this tool could trigger the next step in a procedure, such as the addition of a reagent or heating of the reaction vessel.

From an educator's perspective, it was important to present the project with a central but *broad* goal; this allowed students to demonstrate independence and creativity during the process of identifying which Python objects and functions to write and how to implement each component of this larger project. Programming, especially beyond an education setting, is highly collaborative, and students were able to learn and practice team-based coding, on- and off-site, using GitHub. That is, while students engaged in direct communication to coordinate on data structures and function specifications, they then collaborated remotely via GitHub for code development. To improve the *chemical* educational aspect of this final project, if time permits, students could be tasked with completing a report on the project, in which they describe the protocol and code they developed, analyze the data they collected (e.g., calculate the absorbance and investigate its dependence on the concentration of the absorber), and propose improved or new experiments using the automated computer vision system.

The field of chemistry is rapidly expanding, in part due to the successful integration of current and upcoming technologies with traditional methods and techniques. The automation of laboratory processes is an example of such technology that has the potential to accelerate discovery and production to help chemists advance the field. We intend this project to be a step toward developing an effective educational framework for introducing the role of automation in chemistry at the student level and providing hands-on experience in implementing both hardware and software needed to automate laboratory processes.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available at https://pubs.acs.org/doi/10.1021/acs.jchemed.9b00603.

The parts list for the experimental setup, URL for the publicly available GitHub repository of the AutoVis suite, and course syllabus (PDF) (DOCX)

AUTHOR INFORMATION

Corresponding Authors

Sukin Sim – *Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States*; Email: ssim@g.harvard.edu

Tim Menke – *Department of Physics, Harvard University, Cambridge, Massachusetts 02138, United States; Research Laboratory of Electronics and Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, United States*; Email: tim_menke@g.harvard.edu

Alán Aspuru-Guzik – *Department of Chemistry and Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3H6, Canada; Vector Institute, Toronto, Ontario M5G 1M1, Canada; Zapata Computing Inc., Cambridge, Massachusetts 02139, United States; Canadian Institute for Advanced Research (CIFAR) Senior Fellow, Toronto, Ontario M5G 1M1, Canada*; orcid.org/0000-0002-8277-4434; Email: alan@aspuru.com

Authors

Santiago Vargas – *Harvard College, Harvard University, Cambridge, Massachusetts 02138, United States*

Siavash Zamirpour – *Harvard College, Harvard University, Cambridge, Massachusetts 02138, United States*; orcid.org/0000-0002-3742-4320

Shreya Menon – *Harvard College, Harvard University, Cambridge, Massachusetts 02138, United States*

Arielle Rothman – *Harvard College, Harvard University, Cambridge, Massachusetts 02138, United States*

Florian Häse – *Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States; Department of Chemistry and Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3H6,* *Canada*

Teresa Tamayo-Mendoza − *Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States; Department of Chemistry, University of Toronto, Toronto, Ontario M5S 3H6, Canada*

Jonathan Romero − *Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States; Zapata Computing Inc., Cambridge, Massachusetts 02139, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jchemed.9b00603

Author Contributions

[Φ]These authors contributed equally.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors would like to thank Loïc Roch for insightful discussions, particularly for his contribution to the project design. They would also like to thank the other students from the Physical Sciences 50 course, Nicole Kim, Gabriel Apoj, Margaret Klureza, Christopher Chen, and Liam Power, for their contribution to the software and presentation. In addition, the authors would like to thank the four anonymous reviewers for helpful and insightful comments on the manuscript. Finally, the team would like to thank Gregg Tucci for helping and encouraging us to create the class. S.S., T.M., and A.A.-G. acknowledge financial support from Dr. Anders Frøseth.

■ REFERENCES

(1) Perkel, J. M. The Internet of Things Comes to the Lab. *Nature* **2017**, *542* (7639), 125−126.

(2) Weiss, C. J. Scientific Computing for Chemists: An Undergraduate Course in Simulations, Data Processing, and Visualization. *J. Chem. Educ.* **2017**, *94* (5), 592−597.

(3) Arrabal-Campos, F. M.; Cortés-Villena, A.; Fernández, I. Building "My First NMRviewer": A Project Incorporating Coding and Programming Tasks in the Undergraduate Chemistry Curricula. *J. Chem. Educ.* **2017**, *94* (9), 1372−1376.

(4) Fisher, A. A. E. An Introduction to Coding with Matlab: Simulation of X-ray Photoelectron Spectroscopy by Employing Slater's Rules. *J. Chem. Educ.* **2019**, *96* (7), 1502−1505.

(5) GitHub. https://github.com. (January 2019).

(6) Schwaber, K. *Agile Project Management with Scrum*. Microsoft Press: 2004.

(7) Roch, L. M.; Häse, F.; Kreisbeck, C.; Tamayo-Mendoza, T.; Yunker, L. P. E.; Hein, J. E.; Aspuru-Guzik, A. ChemOS: Orchestrating autonomous experimentation. *Science Robotics* **2018**, 3 (19), No. eaat5559.

(8) Sanderson, K. Automation: Chemistry shoots for the Moon. *Nature* **2019**, *568*, 577−579.

(9) Skilton, R. A.; Bourne, R. A.; Amara, Z.; Horvath, R.; Jin, J.; Scully, M.; Streng, E.; Tang, S. L. Y.; Summers, P. A.; Wang, J.; Pérez, E.; Asfaw, N.; Aydos, G. L. P.; Dupont, J.; Comak, G.; George, M. W.; Poliakoff, M. Remote-controlled experiments with cloud chemistry. *Nat. Chem.* **2015**, *7*, 1−5.

(10) Hambrusch, S.; Hoffmann, C.; Korb, J. T.; Haugan, M.; Hosking, A. L. A Multidisciplinary Approach towards Computational Thinking for Science Majors. *In Proceedings of the 40th ACM Technical Symposium on Computer Science Education* **2009**, 183−187.

(11) Perkel, J. Democratic Databases: Science on GitHub. *Nature* **2016**, *538*, 127−128.

(12) Eppel, S. Tracing liquid level and material boundaries in transparent vessels using the graph cut computer vision approach. *arXiv* 2016, arXiv:1602.00177 [cs.CV].

(13) Eppel, S. Setting an attention region for convolutional neural networks using region selective features, for recognition of materials within glass vessels. *arXiv* 2017, arXiv:1708.08711 [cs.CV].

(14) The course syllabus is listed in the Supporting Information.

(15) Kirkpatrick, M. S. Student perspectives of team-based learning in a CS course: summary of qualitative findings. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* **2017**, 327−332.

(16) Smith, M. K.; Wood, W. B.; Adams, W. K.; Wieman, C.; Knight, J. K.; Guild, N.; Su, T. T. Why peer discussion improves student performance on in-class concept questions. *Science* **2009**, *323*, 122−124.

(17) Aires-de-Sousa, J.; Cardoso, M. M.; Ferreira, L. M.; Lima, J. C.; Noronha, J. P.; Nunes, A. V. M.; Ponte, M. N. d. Team-Based Learning in Chemistry Courses with Laboratory Sessions. *In Proceedings of the 3rd International Conference on Higher Education Advances* **2017**, 1213−1218.

(18) Python. https://www.python.org. (April 2019).

(19) DropBox. https://www.dropbox.com. (April 2019).

(20) OpenCV. https://pypi.org/project/opencv-python. (April 2019).

(21) SimpleCV. http://simplecv.org/. (October 2019).

(22) Pygame. https://www.pygame.org/project/3186. (October 2019).

(23) van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; Yu, T. the scikit-image contributors., scikit-image: Image processing in Python. *PeerJ* **2014**, 2, No. e453.

(24) PIL. https://pillow.readthedocs.io/en/stable/. (October 2019).

(25) Coelho, L. P. Mahotas: Open source software for scriptable computer vision. *arXiv* 2013, arXiv:1211.4907 [cs.CV].

(26) Matplotlib. https://matplotlib.org. (April 2019).

(27) Seaborn. https://seaborn.pydata.org. (April 2019).

(28) Colorsys. https://docs.python.org/2/library/colorsys.html. (April 2019).

(29) Videos of the dashboard over the duration of the respective experiment are available in the GitHub repository linked in the Supporting Information.

(30) Bogucki, R.; Greggila, M.; Mallory, P.; Feng, J.; Siman, K.; Khakipoor, B.; King, H.; Smith, A. W. A 3D-Printable Dual Beam Spectrophotometer with Multiplatform Smartphone Adaptor. *J. Chem. Educ.* **2019**, *96*, 1527−1531.