Exploring Benchmarks for Self-Driving Labs using Color Matching

Tobias Ginsburg
Argonne National Laboratory
Lemont, United States
tginsburg@anl.gov

Kyle Hippe
Argonne National Laboratory
Lemont, United States
khippe@anl.gov

Ryan Lewis
Argonne National Laboratory
Lemont, United States
ryan.lewis@anl.gov

Doga Ozgulbas
Argonne National Laboratory
Lemont, United States
dozgulbas@anl.gov

Aileen Cleary
Northwestern University
Evanston, United States
aileencleary2026@u.northwestern.edu

Rory Butler
UChicago & Argonne National Lab
Lemont, United States
rorymb@uchicago.edu

Casey Stone
Argonne National Laboratory
Lemont, United States
cstone@anl.gov

Abraham Stroka
Argonne National Laboratory
Lemont, United States
astroka@anl.gov

Rafael Vescovi
Argonne National Laboratory
Lemont, United States
ravescovi@anl.gov

Ian Foster
UChicago & Argonne National Lab
Lemont, United States
foster@anl.gov

ABSTRACT

Self Driving Labs (SDLs) that combine automation of experimental procedures with autonomous decision making are gaining popularity as a means of increasing the throughput of scientific workflows. The task of identifying quantities of supplied colored pigments that match a target color, the *color matching problem*, provides a simple and flexible SDL test case, as it requires experiment proposal, sample creation, and sample analysis, three common components in autonomous discovery applications. We present a robotic solution to the color matching problem that allows for fully autonomous execution of a color matching protocol. Our solution leverages the WEI science factory platform to enable portability across different robotic hardware, the use of alternative optimization methods for continuous refinement, and automated publication of results for experiment tracking and post-hoc analysis.

KEYWORDS

Automation, Self-Driving Labs, Color Matching, Artificial Intelligence

ACM Reference Format:
Tobias Ginsburg, Kyle Hippe, Ryan Lewis, Doga Ozgulbas, Aileen Cleary, Rory Butler, Casey Stone, Abraham Stroka, Rafael Vescovi, and Ian Foster. 2023. Exploring Benchmarks for Self-Driving Labs using Color Matching. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3624062.3624614

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624614

1 INTRODUCTION

Self-Driving Laboratories (SDLs) combine automation of individual experimental steps with automated decision procedures (e.g., optimization or AI methods) for selecting new experiments. By thus enabling closed-loop, autonomous discovery, SDLs can accelerate scientific discovery by enhancing precision, efficiency, and repeatability of experimental protocols [1, 6, 7, 10, 15]. When combined with programming frameworks [9, 11, 13, 14] that facilitate the substitution of alternative experimental apparatus, experimental protocols, and decision procedures, SDLs become broadly applicable, multi-purpose research instruments.

The color-mixing problem [2, 9] has been employed by several SDL groups for pedagogical purposes and to permit low-cost experimentation with alternative decision procedures. The task here is to identify quantities of a small number of provided colored pigments that, when mixed, best match a specified target color [2, 9]. While simple, this problem captures important elements of SDL applications: in particular, the need to propose samples, create samples, characterize samples, and process characterization results to generate new proposals.

The authors and their colleagues in Argonne's Rapid Prototyping Lab (RPL) have created a modular, re-configurable robotic workcell to enable multiple robotic workflows on a shared set of hardware. Using this system, we have implemented a color mixing application, *color picker*, that employs five devices that serve other roles in other workflows. The color mixing is performed fully autonomously,

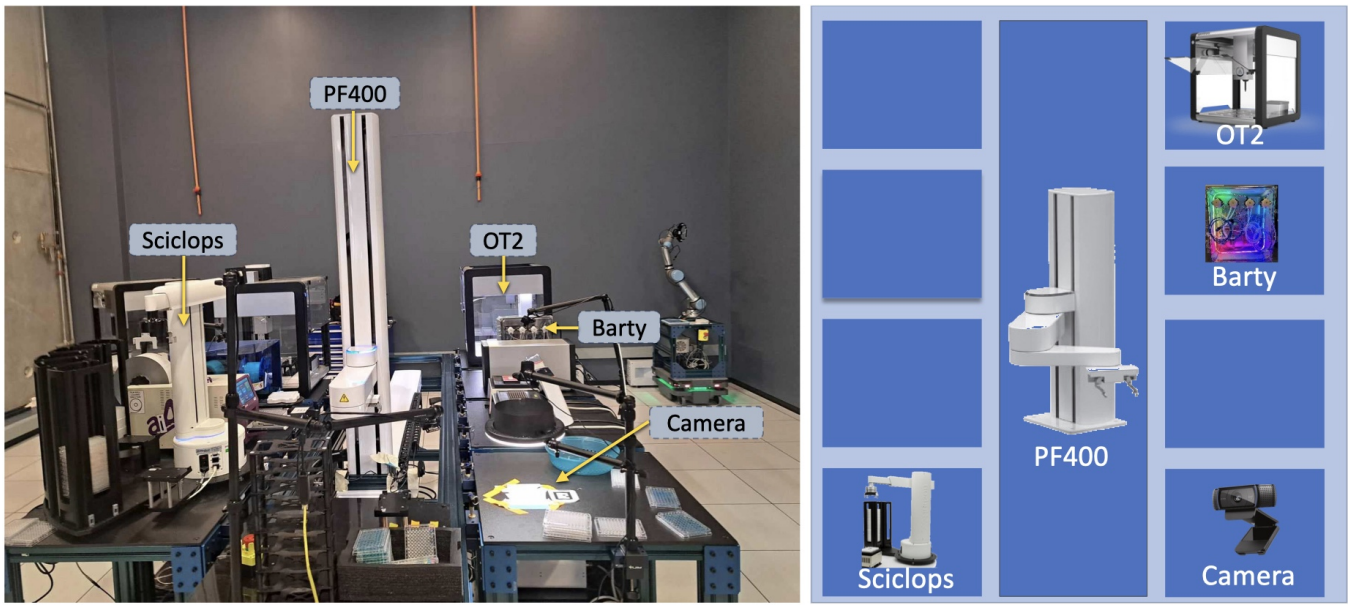

Figure 1: A photograph and diagram of the robotic workcell used for the color picking experiment. The `sciclops` picks up a 96-well plate from its plate storage towers, and transfers it to its exchange location. The `pf400` then transfers the plate to the `ot2`, which mixes the three target colors. When the liquid reservoirs in the system are empty, the custom robot, `barty`, refills them by using peristaltic pumps. Once the mixing is completed, the plate is transferred to the Camera location to be imaged. The plate is then looped between the camera and the OT2 until the experiment is over.

with everything from fetching a new plate to analyzing the color image to uploading data to the web performed completely without human intervention. Over time, we were able to increase the lab's reliability while measuring its performance across a number of relevant metrics, showing the color mixing application's usefulness as an SDL benchmark.

An earlier version of the color picker application was described in a previous paper [13]. Here we describe extensions to that work that incorporate an additional device, the Barty liquid replenisher, provide a more detailed description of the application, and discuss its use as a possible SDL benchmark.

2 METHODS

We describe in turn the color-picker problem, our workcell architecture, our implementation of the problem on the workcell, and other aspects of our approach.

2.1 Application

The color picker application that we present here mixes four component liquids, specifically cyan, yellow, magenta, and black dyes, in a microplate to attempt to produce a liquid sample that matches a target color as closely as possible.

This process consists of three steps. First, our optimization algorithm leverages its (initially empty) set of data obtained to date to propose a set of experiments to perform, expressed as a set of volumes for each liquid. Second, our robotic workcell is instructed to produce these samples by dispensing and mixing the appropriate quantities of each liquid. Third, the workcell is instructed to measure the colors of the samples produced. The results from the third step are then fed back to step 1, until a termination criteria is satisfied. We accomplish these various tasks by using the execution framework described by Vescovi et al. [13].

2.2 Workcell

We have defined in previous work a modular architecture for SDLs in which *modules* encapsulating scientific instruments and other devices (e.g., plate crane) are linked with manipulators to form *workcells* [13]. Each *module* is represented by a software abstraction that exposes a single device and, via interface methods, the actions that the device can perform; a declarative YAML notation is used to specify how a workcell is configured from a set of modules.

Users can specify, again using a declarative notation, *workflows* that perform sets of actions on modules. They can also write Python *applications* that run one or more workflows on specified workcell(s) and typically also invoke associated computational and data manipulation steps, for example to run solvers and publish results. Workflows can be reused in different applications, and modules in different workflows, and workflows can be retargeted to different modules and workcells that provide comparable capabilities. Workflow steps are translated into commands sent to computers connected to devices, which then call driver functions specific to their attached device.

The color picker application that we present here targets five modules: plate crane, manipulator, liquid handler, liquid replenisher, and camera. In our experiments, we execute this application on the

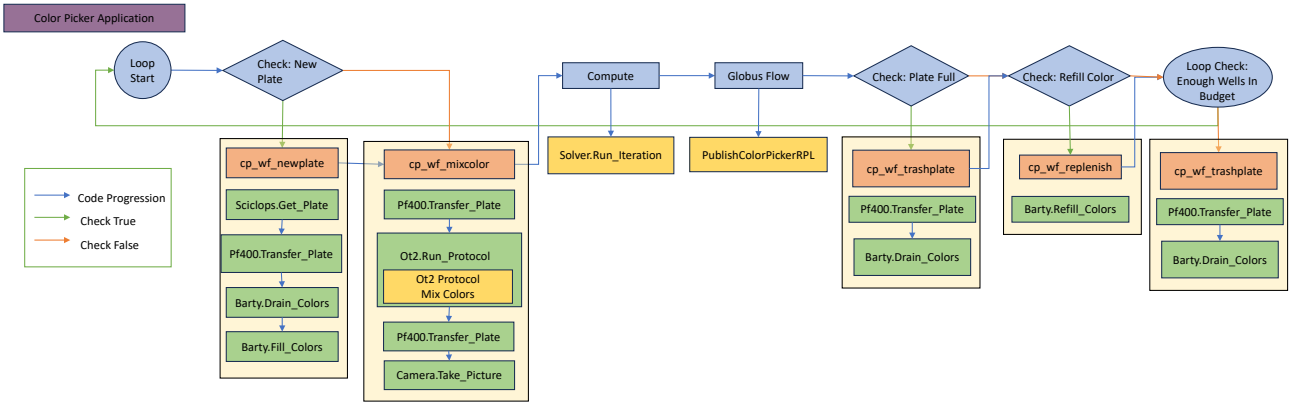

Figure 2: The color-picker application. The Python code, `color_picker_app.py`, implements logic that runs three distinct WEI flows, with the second (plus associated `publish` and `compute` steps) called repeatedly until termination criteria are satisfied. The orange box below the `ot2 run_protocol` action gives the name of the protocol file. Module names are as in Section 2.2.

Argonne Rapid Prototyping Lab workcell, a configuration that comprises 10 modules useful for a variety of experimental protocols in biology and education. Here, we use just the five modules shown in Figure 1, and as specified in the YAML file at https://bit.ly/3ZCVniT.

The complete RPL workcell also includes (not shown) modules used for PCR and for growing and analyzing cells, but this application targets only those depicted in Figure 1:

- `sciclops` is the Hudson SciClops Microplate Handler, a microplate storage and staging system that can access multiple storage towers, facilitating the housing of plates.
- `pf400` is the workcell's manipulator, a robotic arm used to transfer microplates between different plate stations. Operating on a rail mechanism, this robot acts as the central transportation unit within the workcell. Its core function is to shuttle microplates between different modules.
- `ot2` is an automatic pipetting device that contains four separate color reservoirs and a set of pipette tips. Once the `pf400`has delivered a plate to the `ot2` deck, it mixes liquids in the proportions set by the optimization algorithm to generate new sample colors.
- `barty` is a robot developed in RPL with four peristaltic pumps that transfer liquid from large storage vessels to the reservoirs of the the `ot2`. Our application instructs `barty` to refill the `ot2` reservoirs periodically so that experiments can run for extended periods.
- `camera` is a Logitech webcam mounted with a ring light that is used to capture images of the microplate. This module incorporates a microplate mount designed to allow the `pf400` to place the microplate in the same location each time.

2.3 The Color Picker Application

Our color picker application (accessible at https://bit.ly/3ZCigTB) engages four workflows (see the folder https://bit.ly/45bzFUb), as shown in Figure 2. It proceeds as follows:

(1) If a new plate is needed (as when starting or when the previous plate was full), call workflow `cp_wf_newplate` to instruct modules as follows:
   (a) `pf400`: Retrieve new plate from `sciclops`, place it at `camera`
   (b) `barty`: Fill `ot2` reservoirs
(2) Run workflow `cp_wf_mix_colors` to:
   (a) `pf400`: Transfer the plate to `ot2`
   (b) `ot2`: Dispense and combine specified pigment amounts
   (c) `pf400`: Return plate to `camera`
   (d) `camera`: Photograph the plate.
(3) Process the image as described in Section 2.4.
(4) Publish the obtained data, as described below.
(5) Invoke a specified Color Picking Solver to evaluate the data and, if the termination criteria are not met, select the next set of colors.
(6) If the plate is full, run workflow `cp_wf_trashplate` to dispose of the plate and drain the `ot2` reservoirs using `barty`.
(7) If the reservoirs need refilling, run workflow `cp_wf_replenish` to drain and refill them.

Once termination criteria are satisfied (e.g., target color matched or resources exhausted), the application runs `cp_wf_trashplate` again to finalize the experiment.

For each workflow that is run, a file is created that details the step names run, their start time, end time and total duration. These files are saved locally to the maching running the workflow manager.

The publication step engages a Globus flow [4] to publish data to the ALCF Community Data Co-Op (ACDC) data portal (Figure 3). For each run, the data created includes the colors produced, the timing of each step, the scoring results from the solver, and the raw plate images for quality control.

2.4 Image Processing

To process the webcam images for color detections, we station the plate at a known distance from an ArUco marker [5]. Using OpenCV [3], we detect the ArUco marker in the image, and use the

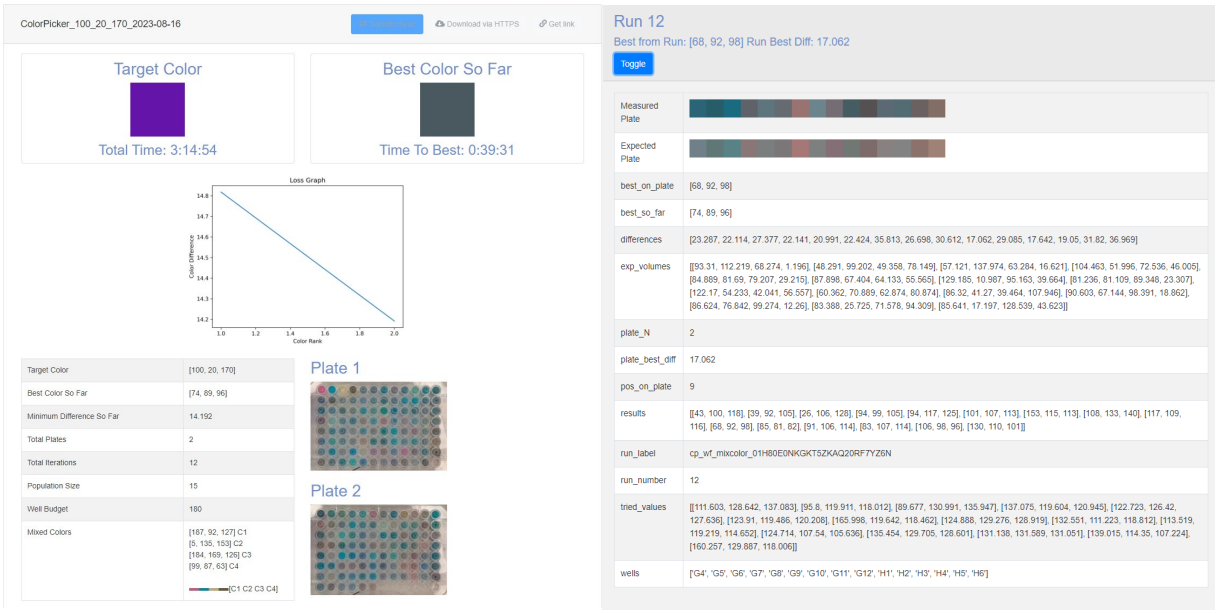

Figure 3: Two views of a Globus Search portal for data generated by the color-picker application, at https://acdc.alcf.anl.gov. *Left*: Summary view for an experiment performed on August 16th, 2023, involving 12 runs each with 15 samples, for a total of 180 experiments. The images are those taken by the camera. *Right*: Detailed data from run #12.

size and position of the marker to determine the approximate pixel-coordinate boundaries of the microplate. To account for potential shifting in the camera position, we further refine the location of the microplate by drawing on a particular physical characteristic of the plate: the circular wells. With the HoughCircles algorithm from OpenCV, we can detect circular features in the image to precisely identify the center of wells. As this method is prone to false negatives, we further align a grid to all well-sized circles within the approximate plate position, and use this grid's size and orientation to predict the center points for all wells in the image, even those originally missed by the HoughCircles algorithm. Finally, the detected colors at the well center points are reported to the optimization algorithm.

2.5 Color Picking Solvers

The color picking problem admits to an analytic solution, given accurate models of how colors combine and the properties of our color sensor. However, treating the problem as a black box, as we do here, allows us to employ the problem as a surrogate for more complex problems and to experiment with different decision procedures in a simple setting. We have implemented to date two such decision procedures, a simple evolutionary solver (a genetic algorithm) and a Bayesian solver, thus demonstrating the ability to run multiple optimization algorithms without changes to other elements of the system.

Genetic algorithms (GAs) are optimization and search heuristics inspired by natural evolution. Initially populated with random or heuristically chosen solutions, each iteration—termed a generation—evaluates individuals based on a fitness function. The fittest individuals are selected, and the remainder of the population is augmented. Over multiple generations, the population converges towards optimal or near-optimal solutions. For the initial population, points are sampled from a uniform grid of proper dimensions (corresponding to the number of mixing colors). Once a population of colors with grades (floating point values representing *delta e* distance to the target) a new population is created. The most accurate element of the previous population is propagated into the new generation. One third of the new population is created by randomly selecting two elements of the previous population and taking the average of them. One third of the population is created by taking a random element of the previous population and randomly shifting its ratios. The final third of the population is created by randomly creating a new set of ratios. The code for this is at https://bit.ly/3rGxI4m. We present results obtained with the genetic algorithm in Figure 4.

We also implemented a Bayesian optimization method based on scikit-learn [8]: see https://bit.ly/3EVDNgs. Bayesian optimization leverages a surrogate probabilistic model, commonly Gaussian Processes, to approximate the objective function and iteratively refines this based on evaluations. We do not present results obtained with this approach as they do not yield a systematic improvement over the genetic algorithm.

3 RESULTS

We present results obtained when running the color picker application with the simple evolutionary solver, a fixed target color, and the total number of samples fixed at N=128. We varied the batch size B across different experiments by powers of two from 1 to 64.

The results, as depicted in Figure 4, show outcomes from varying batch sizes. Each dot in the figure represents the time elapsed in

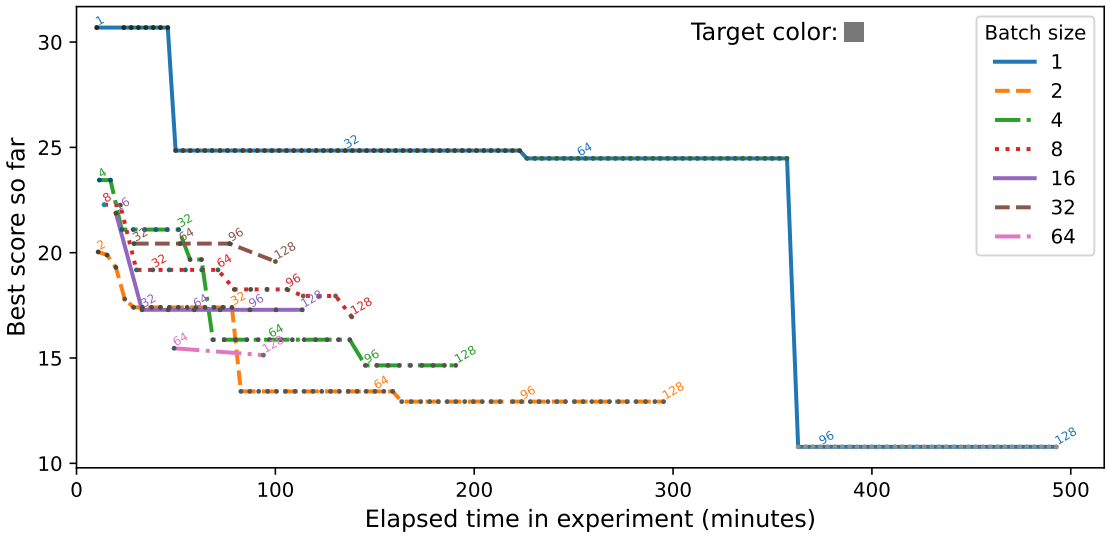

Figure 4: Results of seven experiments, in each of which the color picker application creates and evaluates 128 samples, in batches of an experiment-specific size B = 1, 2, 4, 8, 16, 32, and 64. In each experiment, the target color is RGB=(120,120,120), the first sample(s) are chosen at random, and later samples are chosen by applying a solver algorithm to `camera` images. Each dot has as x-value the elapsed time in the experiment and as y-value the Euclidean distance in three-dimensional color space between the target color and the best color seen so far in the experiment. The evolutionary algorithm used has random elements, which means that improvement between iterations is not guaranteed, leading to the long flat stretches in the graph. The numbers in the graph represent selected sample sequence numbers. Results depend on the original random guesses, but overall, as we might expect, experiments with smaller batch sizes achieve lower scores, but take longer to run. (Figure adapted from [13].)

the experiment (x-axis) against the Euclidean distance in the three-dimensional color space between the target color and the best-matched color at that point (y-axis).

It is evident from the results that the initial random guesses influence the outcomes. However, a general trend observed is that experiments with smaller batch sizes took longer to run but were more accurate in color matching.

The data publication capabilities allowed us to capture these results efficiently. Figure 3 provides two views of the Globus Search portal, detailing data generated by the application.

Our longest run with B=1 (i.e., a single sample per iteration) completes in 8 hours and 12 minutes without human intervention. This run consisted of 387 distinct robotic actions that were performed without error, and 128 distinct data upload steps that allowed for fine-grained tracking of the progress of the experiment. Data uploads occurred on average every 3 minutes and 48 seconds, and required moving the microplate from the camera module to the `ot2` and back, meaning the `pf400` had to pick and place the microplate precisely twice per time period.

4 DISCUSSION

We have presented a solution to the color matching problem as an exemplary implementation of experiment-in-the-loop computing. Leveraging a recent proposed modular SDL architecture [13], our implementation combines fine-grained control of various experimental apparatus with invocations of computing and data resources, accessed and coordinated by using Globus automation service mechanisms [12]. Continuous publishing of results permits monitoring of progress, and its modular architecture allows for the substitution of alternative instruments, optimization solvers, computing resources, and storage resources. While certainly not extreme-scale, we expect this application to be of interest to XLOOP attendees as a test case for their own experiment-in-the-loop approaches.

The color-mixing application allows us to test aspects of a setup that we believe are necessary for any multipurpose self-driving lab application. Specifically, it requires a synthesis step and a data acquisition step, with distinct instruments for each step. We suggest this sets a self-driving lab apart from single instrument stations that can perform multiple functions. It also requires multiple iterations of a similar process, necessary for using optimization or machine-learning algorithms to generate samples. The reliability with which we were able to run it allowed us to test the resiliency of our work-cell infrastructure and learn more about the nature of errors that could occur, and also to compare different optimization algorithms. Because the color picking experiment uses inert reagents, it can be run on multiple different forms of hardware. As such, we believe it could serve as a benchmark application for self-driving labs. Specifically, we will discuss metrics around a color-mixing run with a batch size of 1, as this required the most robot coordination, and ran for the greatest amount of time.

Table 1: Proposed metrics for self-driving labs and our best results for a color picker batch size of 1.

Metric	Value for B = 1
Time without humans	8 hours 12 mins
Completed commands without humans	387
Synthesis time	5 hours 10 mins
Transfer time	3 hours 2 mins
Total colors mixed	128
Time per color	4 mins

As the number of multi-purpose self-driving labs increases, the creation of metrics to compare and contrast their capabilities becomes relevant both to designers hoping to measure their performance of their systems and to researchers hoping to best utilize the available resources. We propose three metrics for evaluating the performance of the color-mixing experiment. We suggest that metrics such as these can provide a basis for comparison across different SDL implementations, and ultimately may allow facilities to convey their capabilities accurately to possible users.

The first metric is *time without human input* (TWH), equivalent to the longest time that an experiment ran without human intervention. This metric gives a sense of how much human time is being saved by automating the lab. However, it does not necessarily reflect a level of automation, as it is sensitive to how long individual actions take to run. Our best run according to this metric was 8 hours and 12 minutes.

The second metric is *commands completed without human input* (CCWH), i.e., the number of commands sent and successfully executed by the instruments over the course of the experiment without human intervention. A *command* is defined as one or more actions carried out consecutively by a single instrument without input from the control system. A *completed* command is one for which the instrument successfully performs each constituent action and reports success to the system. In our experience, most failures occur during reception and processing of commands, making CCWH a good measure of the resiliency of the SDL's communications and the reliability of its constituent instruments. Our best run according to this metric involved 387 successful commands. An interesting future experiment would involve integrating additional OT2s in our workflow, so that multiple plates of colors could be mixed at once This would lead to an increase in CCWH, but potentially a lower TWH for the same experimental results.

The third metric, which speaks to the efficiency of the lab overall, is *time per color*, i.e., total experiment run time divided by the number of color samples produced. This metric captures the per-sample efficiency and allows comparisons across different self-driving lab configurations and levels of parallelism. We achieved around four minutes per color for our B=1 run. We can also divide the total run time into *synthesis time*, that used specifically to mix colors, and *transfer time*, that used to move samples between instruments, which can help provide more information on where the bottlenecks in the system lie. Our B=1 run achieved 5 hours 10 minutes synthesis time and 3 hours 2 minutes transfer time, meaning that 63% of the total time time was spent mixing colors using the OT2, possibly an area for improvement.

In future work we would like to integrate our system with Baird and Sparks' closed-loop spectroscopy lab code [2] so as to permit experimentation with their various optimization codes and different search approaches.

ACKNOWLEDGMENTS

We thank Eric Codrea, Yuanjian Liu, and other students for their contributions, and Ryan Chard, Nickolaus Saint, and others in the Globus team for their ongoing support. This work was supported in part by Laboratory Directed Research and Development funds at Argonne National Laboratory from the U.S. Department of Energy under Contract DE-AC02-06CH11357.

REFERENCES

[1] Milad Abolhasani and Eugenia Kumacheva. 2023. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis* (2023), 1–10.
[2] Sterling G Baird and Taylor D Sparks. 2023. Building a "Hello World" for self-driving labs: The Closed-loop Spectroscopy Lab Light-mixing demo. *STAR Protocols* 4, 2 (2023), 102329.
[3] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
[4] Ryan Chard, Jim Pruyne, Kurt McKee, Josh Bryan, Brigitte Raumann, Rachana Ananthakrishnan, Kyle Chard, and Ian T Foster. 2023. Globus automation services: Research process automation across the space-time continuum. *Future Generation Computer Systems* 142 (2023), 393–409.
[5] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292.
[6] Florian Häse, Loïc M Roch, and Alán Aspuru-Guzik. 2019. Next-generation experimentation with self-driving laboratories. *Trends in Chemistry* 1, 3 (2019), 282–291.
[7] Benjamin P MacLeod, Fraser GL Parlane, Thomas D Morrissey, Florian Häse, Loïc M Roch, Kevan E Dettelbach, Raphaell Moreira, Lars PE Yunker, Michael B Rooney, Joseph R Deeth, V. Lai, G. J. Ng, R. H. Zhang H. Situ, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein, and C. P. Berlinguette. 2020. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances* 6, 20 (2020), eaaz8867.
[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[9] Loïc M Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars PE Yunker, Jason E Hein, and Alán Aspuru-Guzik. 2020. ChemOS: An orchestration software to democratize autonomous discovery. *PLoS One* 15, 4 (2020), e0229862.
[10] Eric Stach, Brian DeCost, A Gilad Kusne, Jason Hattrick-Simpers, Keith A Brown, Kristofer G Reyes, Joshua Schrier, Simon Billinge, Tonio Buonassisi, Ian Foster, Carla P. Gomes, John M. Gregoire, Apurva Mehta, Joseph Montoya, Elsa Olivetti, Chiwoo Park, Eli Rotenberg, Semion K. Saikin, Sylvia Smullin, Valentin Stanev, and Benji Maruyama. 2021. Autonomous experimentation systems for materials development: A community perspective. *Matter* 4, 9 (2021), 2702–2726.
[11] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip Kitson, Davide Angelone, and Leroy Cronin. 2019. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science* 363, 6423 (2019).
[12] Rafael Vescovi, Ryan Chard, Nickolaus D Saint, Ben Blaiszik, Jim Pruyne, Tekin Bicer, Alex Lavens, Zhengchun Liu, Michael E Papka, Suresh Narayanan, Nicholas Schwarz, Kyle Chard, and Ian T. Foster. 2022. Linking scientific instruments and computation: Patterns, technologies, and experiences. *Patterns* 3, 10 (2022), 100606.
[13] Rafael Vescovi, Tobias Ginsburg, Kyle Hippe, Doga Ozgulbas, Casey Stone, Abraham Stroka, Rory Butler, Ben Blaiszik, Tom Brettin, Kyle Chard, Mark Hereld, Arvind Ramanathan, Rick Stevens, Aikaterini Vriza, Jie Xu, Qingteng Zhang, and Ian Foster. 2023. Towards a Modular Architecture for Science Factories. ArXiv https://arxiv.org/abs/2308.09793.
[14] Justin Vrana, Orlando de Lange, Yaoyu Yang, Garrett Newman, Ayesha Saleem, Abraham Miller, Cameron Cordray, Samer Halabiya, Michelle Parks, Eriberto Lopez, Sarah Goldberg, Benjamin Keller, Devin Strickland, and Eric Klavins. 2021. Aquarium: Open-source laboratory software for design, execution and data management. *Synthetic Biology* 6, 1 (2021), ysab006.
[15] Aikaterini Vriza, Henry Chan, and Jie Xu. 2023. Self-driving laboratory for polymer electronics. *Chemistry of Materials* 35, 8 (2023), 3046–3056.