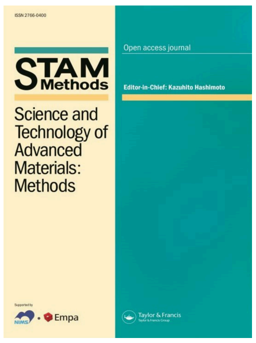


Science and Technology of Advanced Materials: Methods

ISSN: 2766-0400 (Online) Journal homepage: www.tandfonline.com/journals/tstm20

Seamless integration of legacy robotic systems into a self-driving laboratory via NIMO: a case study on liquid handler automation

Ryo Tamura, Hiromichi Taketa, Satoshi Murata, Daisuke Ryuno, Tomotaka Yokota, Koji Tsuda & Shoichi Matsuda

To cite this article: Ryo Tamura, Hiromichi Taketa, Satoshi Murata, Daisuke Ryuno, Tomotaka Yokota, Koji Tsuda & Shoichi Matsuda (2025) Seamless integration of legacy robotic systems into a self-driving laboratory via NIMO: a case study on liquid handler automation, Science and Technology of Advanced Materials: Methods, 5:1, 2565144, DOI: 10.1080/27660400.2025.2565144

To link to this article: https://doi.org/10.1080/27660400.2025.2565144

SCIENCE AND TECHNOLOGY OF ADVANCED MATERIALS: METHODS
2025, VOL. 5, NO. 1, 2565144
https://doi.org/10.1080/27660400.2025.2565144

Taylor & Francis
Taylor & Francis Group

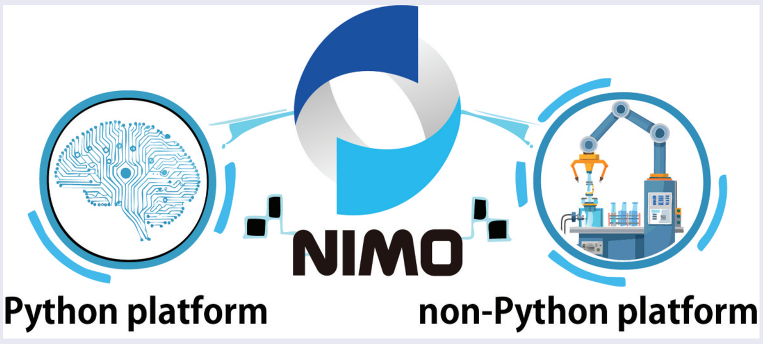
OPEN ACCESS Check for updates

Seamless integration of legacy robotic systems into a self-driving laboratory via NIMO: a case study on liquid handler automation

Ryo Tamura[a,b], Hiromichi Taketa[c], Satoshi Murata[d], Daisuke Ryuno[d], Tomotaka Yokota[d], Koji Tsuda[a,b] and Shoichi Matsuda[e]

[a]Center for Basic Research on Materials, National Institute for Materials Science, Tsukuba, Ibaraki, Japan; [b]Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Chiba, Japan; [c]ForDX, Bunkyo-ku, Tokyo, Japan; [d]Furukawa Electric Advanced Engineering Co., Ltd., Ichihara-shi, Chiba, Japan; [e]Research Center for Energy and Environmental Materials, National Institute for Materials Science, Tsukuba, Ibaraki, Japan

ABSTRACT
The orchestration software (OS) for controlling self-driving laboratories (SDLs) has been advanced significantly in recent years. We developed NIMO (formerly NIMS-OS, NIMS Orchestration System), an OS explicitly designed to integrate multiple artificial intelligence (AI) algorithms with diverse exploratory objectives. NIMO provides a framework for integrating AI into robotic experimental systems that are controlled by other OS platforms based on both Python and non-Python languages. In this study, we demonstrate the realization of an SDL via NIMO by integrating AI into a legacy robotic system. As a proof of concept, we integrated an automated liquid handling system controlled by a Visual Basic (VB) program into the SDL through NIMO and performed parameter optimization of the dispensing process using Bayesian optimization, thereby enabling autonomous and automated experiments. NIMO facilitates AI integration through straightforward file exchanges, ensuring compatibility with robotic experimental systems programmed in non-Python languages such as VB and LabVIEW, as well as SDLs managed by other OS platforms. We anticipate that NIMO's ability to support a broad spectrum of AI-driven autonomous experiments will significantly enhance the functionality and versatility of SDLs.



IMPACT STATEMENT
NIMO enables AI-driven automation in self-driving labs by bridging diverse experimental systems, including those using non-Python platforms, greatly enhancing SDL accessibility and flexibility.

ARTICLE HISTORY
Received 3 May 2025
Revised 17 September 2025
Accepted 18 September 2025

KEYWORDS
Self-driving laboratory; liquid handler automation; Bayesian optimization

1. Introduction

The advent of self-driving research has revolutionized the fields of chemistry and materials science. Robots now conduct experiments automatically, while artificial intelligence (AI) determines the conditions for subsequent experiments, creating a closed loop system that operates without human intervention. This autonomous automated approach is increasingly being implemented worldwide to develop novel materials and molecules [1–11]. Numerous AI and machine learning methods have been devised for materials and chemical research [12–15]. Many of these methods are freely available as open-source tools, facilitating their

CONTACT Ryo Tamura TAMURA.Ryo@nims.go.jp Center for Basic Research on Materials, National Institute for Materials Science, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan; Hiromichi Taketa hiromichi.taketa@fordx.co.jp ForDX, 1-33-6 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan; Shoichi Matsuda matsuda.shoichi@nims.go.jp Research Center for Energy and Environmental Materials, National Institute for Materials Science, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan
Supplemental data for this article can be accessed online at https://doi.org/10.1080/27660400.2025.2565144

widespread adoption. Simultaneously, the emergence of programmable devices has enabled the development of self-driving laboratories (SDLs) tailored to diverse experimental objectives.

These SDLs integrate multiple experimental devices with technologies, such as AI, data mining methods, and data storage systems, requiring complex procedural coordination. To achieve seamless integration and workflow management, orchestration software (OS) plays a critical role. OS is a specialized program that unifies these components, ensuring efficient operation of SDLs. Given the unique requirements of different laboratories – ranging from experimental devices to workflows – OSs are frequently custom-built to suit specific needs, resulting in the creation of proprietary SDLs. However, there is a growing movement toward the development of open-source OS solutions, particularly those written in Python, which enable users to construct SDLs with relative ease. Table 1 provides a summary of the features and characteristics of major open-source OS platforms [16–26].

To construct an SDL where AI algorithms and robotic experiments are seamlessly integrated, it is ideal for all the programming to be written in a unified language across the system. Since Python is the standard programming language in AI algorithm development, the control system for robotic experiments and the interface for data exchange between AI algorithms and robotic experiments should ideally also be constructed using Python-based code. However, in reality, many robotic experimental systems and the instruments integrated within them are controlled by programming languages other than Python. Therefore, it is important to develop an interface between AI algorithms and robotic experimental systems controlled by both Python and non-Python languages such as LabVIEW and Visual Basic.

Based on these research considerations, we recently developed NIMO (formerly NIMS-OS, NIMS Orchestration System) [21] as an OS designed to accommodate multiple AI methods with varying exploratory objectives while maintaining a unified data format. Currently, NIMO supports several AI methods as standard implementations, including Bayesian optimization via PHYSBO [27], the non-objective search method BLOX [28], the phase diagram construction method PDC [29–31], and random sampling. By introducing a common data format, NIMO facilitates the seamless integration of new algorithms for robotic experiments. The standard workflow for conducting autonomous automated experiments using NIMO comprises the following steps: (1) NIMO selects promising experimental conditions from the 'candidates.csv' file and output them to 'proposals.csv'. (2) Input files required to control the robotic experimental system are prepared by NIMO. (3) The robotic experiments are executed, and the results are outputted. (4) NIMO updates the 'candidates.csv' file and proposes subsequent experimental conditions. Unlike other OS platforms that aim to manage all operations within an SDL using a single OS, NIMO's design enables the addition of diverse AI methods as wrappers to existing robotic experimental systems that are controlled by other OS platforms based on both Python and non-Python languages.

Notably, NIMO enables legacy robotic experimental systems controlled by non-Python languages to be structured as SDLs by introducing two Python programs: 'preparation_input.py' and 'analysis_output.py', which serve as the interface between AI algorithms and robotic experiments (see Figure 1(a)). In this sense, modularizing AI algorithms and robotic experiments is a highly effective approach for converting various existing robotic experimental systems into SDLs. Since 'preparation_input.py' and 'analysis_output.py' depend on the specific robotic experimental system, they need to be individually developed by the system's developers. However, for developers who primarily use programming languages other than Python, creating these interface programs in Python may not always be an easy task. This challenge has been one of the limiting factors in the broader adoption of NIMO.

Table 1. Summary of orchestration software (OS) as open source.

Name	Year	Characteristics	GitHub
Chemios [16]	2018	Integration of pumps, spectrometers and temperature controllers	https://github.com/Chemios/chemios
ChemOS [17]	2020	Integration of experimental devices, AI, and database management for chemistry	https://github.com/aspuru-guzik-group/ChemOS
ARES_OS [18]	2021	Specializing in additive manufacturing for 3D printing technology	https://github.com/AFRL-ARES/ARES_OS
HELAO [19]	2022	Adoption of fastAPI to control the experimental devices	https://github.com/helgestein/helao-pub
Flowchem [20]	2022	Specializing in controlling chemical laboratory devices such as pumps and valves	https://github.com/cambiegroup/flowchem
NIMO (NIMS-OS) [21]	2023	Implementation of multiple AI algorithms depending on the research needs	https://github.com/NIMS-DA/nimo
LABS [22]	2023	Providing a centralized user interface for chemistry laboratory automation	https://github.com/Waldvogel-Group/LABS-User-Interface
ChemOS 2.0 [23]	2023	Realization of orchestration with ab initio calculations and portable 'Operating System'	https://github.com/malcolmsimgithub/ChemOS2.0
OCTOPUS [24]	2024	Operation control system for task optimization and job parallelization	https://github.com/KIST-CSRC/Octopus
AlabOS [25]	2024	Simultaneous execution of workflows and managing resources	https://github.com/CederGroupHub/alabos
IvoryOS [26]	2025	Automatically generation of web interfaces from Python-based SDL code	https://gitlab.com/heingroup/ivoryos

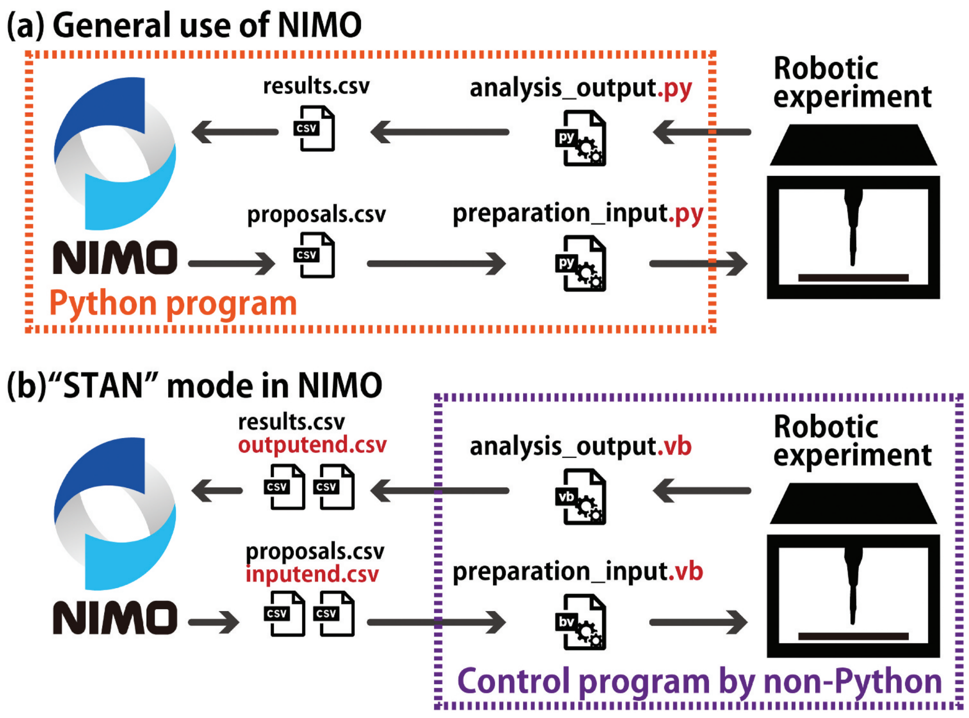


Figure 1. Workflows for (a) general use of NIMO and (b) 'STAN' mode. When 'STAN' mode is used, the interface programs can be created using programming languages employed to control robotic experiments such as LabVIEW and Visual Basic.

To address this issue, in the present study, we developed a more versatile strategy for constructing the interface between AI algorithms and legacy robotic experiments. Specifically, by placing the interface programs corresponding to 'preparation_input.py' and 'analysis_output.py' on the robotic experiment side (see Figure 1(b)), it becomes possible to create the interface using some programming languages used to control robotic experimental systems, rather than Python. This design concept is particularly realized through the use of NIMO in STANdard (STAN) mode which will be explained in Sec. 2.3. We demonstrated the effectiveness of this design by constructing an SDL for an automated liquid handler controlled by Visual Basic (VB). In an SDL, it is essential to carry out AI-proposed multi-condition solution mixing with high precision and speed, making the liquid handler a critical core instrument. A single liquid handler can automatically switch between multiple liquid samples, aspirate, and dispense them under program control, enabling complex solution mixing. While syringe pumps are effective for the precise delivery of a small number of specific solutions in an SDL, liquid handlers are more suitable for condition exploration involving many liquid samples. Although Python-controllable liquid handlers, such as those from Opentrons [32], are available, most of those exists are operated using software other than Python. Here, Bayesian optimization, one of the AI algorithms implemented in NIMO, is applied to automate liquid handling. The target problem is to optimize dispensing parameters for achieving uniform droplets under the constraint of constant volume. Maintaining droplet uniformity is crucial, as it ensures reproducibility, precise control over reactions, and consistent product quality and performance.

2. Integration of legacy liquid handler into a SDL via NIMO

2.1. Devices to construct SDL

The BioDot AD1520 liquid handler was employed for contactless dispensing. Dispensing in the BioDot is controlled by two parameters: steady-state pressure, which determines the syringe pressure, and open time, which specifies the dispensing valve opening/closing duration. A Dino-Lite AF4115ZTW camera was utilized for capturing experimental data.

2.2. Automated liquid handling system using Visual Basic

The integration of the liquid handler and camera measurement was achieved using a VB program and several software tools for Windows, and the constructed system is shown in Figure 2. The dispenser and camera are mounted on a three-axis robotic arm, which allows for precise movement along the XYZ axes. The BioDot AD1520 was connected to a USB port of Windows PC by converting RS-232C to USB for controlling the XYZ coordinate movement of the robotic arm and Ethernet for pump control. A Dino-Lite camera was mounted on the robotic arm and connected to the PC through a USB port. The system

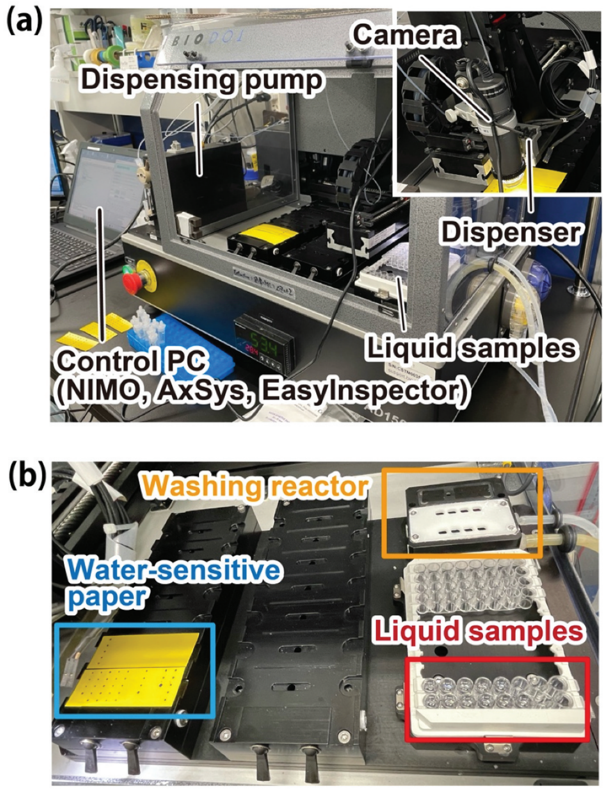


Figure 2. Images of the automated liquid handling system. (a) Overview of the system comprising a BioDot dispenser, a Dino-Lite camera, and a control PC running NIMO, AxSys, and EasyInspector software. (b) Layout of the water-sensitive paper, liquid samples, and washing reactor. The system can accommodate up to 14 sheets of water-sensitive papers.

equipped with AxSys software for controlling the BioDot dispenser and EasyInspector software (SkyLogic) for image acquisition. Both Windows software tools were installed on the PC. To ensure seamless operation, the automated experimental system was controlled via a Visual Basic (VB) program, which managed ActiveX components. ActiveX control or C++ dispatch is required for operating AxSys, while the EasyInspector software, which supports serial communication, was also controlled through VB.

Liquid handling and capturing experimental data were performed according to the procedure detailed in the supplementary movie. The process involved the following steps:

(1) Washing the inside and outside of the dispenser
(2) Injecting the liquid sample into the dispenser
(3) Ejecting the liquid sample five times onto water-sensitive papers
(4) Washing the inside and outside of the dispenser
(5) Photographing the droplet and performing image analysis (five times)

An example of a recipe file for AxSys that describes the operations from (1) to (4) is provided as 'supplementary_file1.txt'. Additionally, a text file containing the XYZ coordinates for the liquid sample positions is loaded into AxSys. During (5), VB generates a text file specifying the XYZ coordinates for the droplet positions, and a program to move the robotic arm (Fig. S1) is executed via AxSys. The image acquisition and analysis are carried out using the EasyInspector software. Supplementary codes 1 and 2 provide examples of VB code used for the interfacing with AxSys and EasyInspector, respectively. The entire liquid handling operation, encompassing (1) to (5), is controlled by the VB program. The program workflow is illustrated by the purple dotted square in Figure 3. The total time required to complete these operations is approximately three minutes. This system was controlled by a non-Python program and does not include any AI components. Thus, it is an important target for implementing closed-loop control using NIMO.

2.3. SLD construction using NIMO

NIMO's 'STAN' mode facilitates the addition of AI to the automated liquid handling system introduced in Sec. 2.2. In 'STAN' mode, when NIMO proposes the next set of experimental conditions, it outputs a file named 'proposals.csv' and creates an empty file named 'inputend.txt'. After that, NIMO remains in standby until it detects an empty file named 'outputend.txt'. Once 'outputend.txt' is detected, NIMO exits standby mode. Thus, synchronization between the robotic experimental system and the AI can be achieved using only file exchange.

To synchronize NIMO with the automated liquid handling system, minor modifications to the VB program are needed. In 'STAN' mode, when NIMO generates the next set of experimental conditions, it outputs 'proposals.csv' and 'inputend.txt' (see Figure 3). The VB program detects the presence of 'inputend.txt' and initiates the experimental procedures automatically. During this process, 'inputend.txt' is removed by VB, signalling that the experiments have started. In 'STAN' mode, NIMO remains in a standby state until it detects 'outputend.txt', which is created by VB at the end of the experimental operations (see Figure 3). Once 'outputend.txt' is generated, NIMO's standby state is terminated. Simultaneously, VB creates 'results.csv' which contains the evaluated objective function values. Using the data in 'results.csv', NIMO updates 'candidates.csv' automatically. These steps establish seamless integration between NIMO and the automated experimental system through simple file exchanges (see Figure 3). It is important to note that an SDL can be constructed with minimal modifications to the VB program for the automated liquid handling system. Specifically, VB programs for file exchange and for rewriting dispensing parameters in the recipe file need to be added, ensuring the original workflow of the robotic experiments remains unchanged. In addition, there is no need to develop Python programs to connect AI and robotic experiments.

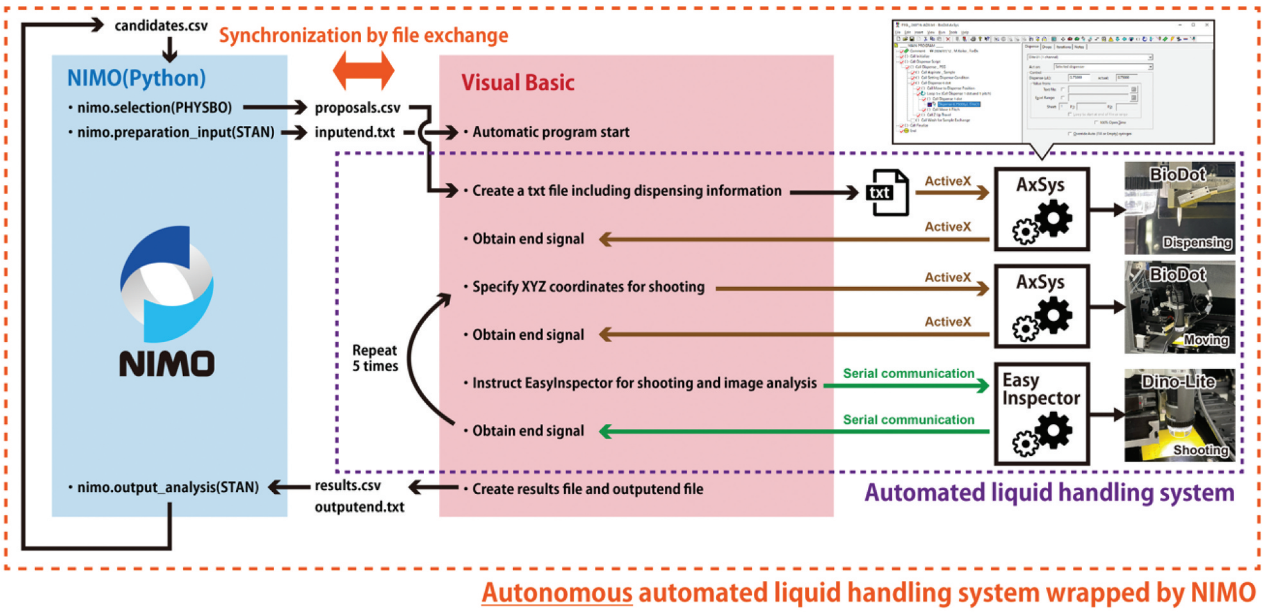

candidates.csv
Synchronization by file exchange
NIMO(Python)
• nimo.selection(PHYSBO)
• nimo.preparation_input(STAN)
proposals.csv
inputend.txt
Visual Basic
• Automatic program start
• Create a txt file including dispensing information
txt
ActiveX
AxSys
BioDot
Dispensing
• Obtain end signal
ActiveX
• Specify XYZ coordinates for shooting
ActiveX
AxSys
BioDot
Moving
Repeat 5 times
• Obtain end signal
ActiveX
• Instruct EasyInspector for shooting and image analysis
Serial communication
Easy Inspector
Dino-Lite
Shooting
• Obtain end signal
Serial communication
NIMO
• nimo.output_analysis(STAN)
results.csv
outputend.txt
• Create results file and outputend file
Automated liquid handling system
Autonomous automated liquid handling system wrapped by NIMO

Figure 3. Workflow of the program for achieving a self-driving liquid handling system. The purple dotted square illustrates the program flow in Visual Basic (VB) for the automated liquid handling system. The orange dotted square represents the complete workflow of the autonomous automated liquid handling system, managed by NIMO. Synchronization between NIMO and VB is facilitated through the exchange of text files ('inputend.txt' and 'outputend.txt') using NIMO's STAN mode.

3. Demonstration of autonomous automated liquid handling

3.1. Definition of optimization problem

In this study, we used the optimization of dispensing parameters for achieving uniform droplets with a commercial liquid handler as a model system for our closed-loop experimental investigation. In particular, we utilized a non-contact micro-dispensing system driven by a combination of a solenoid valve and a high-precision syringe pump. The droplet uniformity is measured by performing image analysis with a camera using water-sensitive paper. Droplet uniformity is defined as the average of the vertical and horizontal diameters of five independent droplets, measured after contacting the paper and before drying, with the liquid volume kept constant. Maintaining droplet uniformity affects product quality and performance stability.

The uniformity of the droplets ejected by the dispenser depends on the various factors which are collectively referred to as dispensing parameters throughout this paper. These factors include the physical properties of the solution such as viscosity and surface tension, as well as the operating parameters of the dispenser, including syringe pressure and dispensing valve opening/closing duration. Thus, achieving the desired droplet uniformity requires precise tuning of these parameters. Conventionally, this parameter optimization process has been conducted through a trial-and-error-based approach, relying heavily on the operator's experience. While common, this approach presents significant challenges in terms of efficiency and reproducibility. Based on these considerations, this study aims to systematically analyze and optimize the parameters of liquid dispensing to achieve high droplet uniformity.

The experimental procedure to measure the droplet uniformity is outlined as follows:

(1) A 100 nL sample was dispensed five times according to the specified dispensing parameters onto a water-sensitive paper at regular intervals.
(2) The vertical and horizontal diameters of each droplet were measured on the EasyInspector software (see Figure 4). The measurement is performed after the droplet contacts the paper, before drying occurs. If image analysis detects any droplet splashing, the diameters for the corresponding spot are set to zero.
(3) For the five spots, the vertical and horizontal measurements were averaged as $\bar{L}$.
(4) Assuming that the uniform droplets have an average diameter of L, the objective function was defined as:

$$O = -|\bar{L} - L|. \tag{1}$$

When $\bar{L} = L$, the value of O is maximized.

To demonstrate the autonomous automated liquid handling, a closed-loop exploration of dispensing parameters was conducted. Bayesian optimization using PHYSBO served as the AI implemented in NIMO. For the experiments, eleven aqueous solutions with varying concentrations of polyethylene glycol (PEG) were prepared, ranging from 0% to 25% in

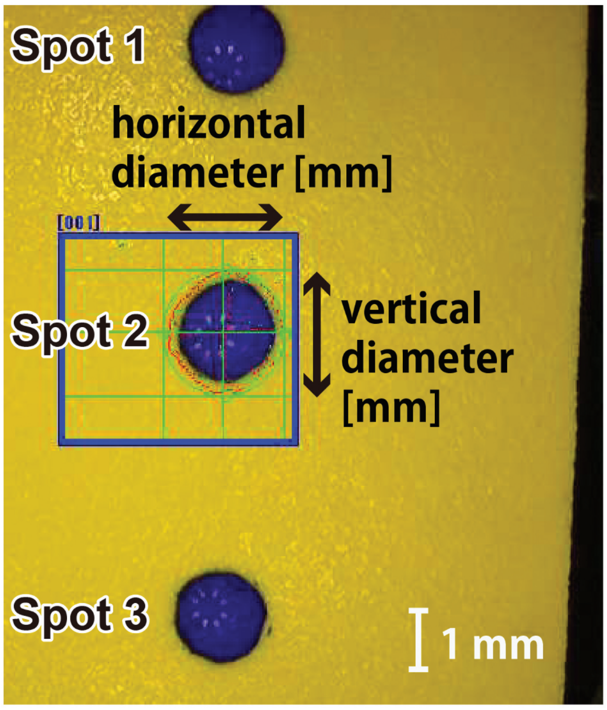


Figure 4. Example of spots on the water-sensitive paper and the corresponding image analysis results. The horizontal and vertical diameters of a total of five spots are measured, and their average value ($\bar{L}$) is used to define the objective function as described in Eq. (1).

2.5% increments. As the viscosity of these solutions increased monotonically with PEG concentration, this value set as variable parameter. In addition, the process parameter of dispenser, steady-state pressure and dispensing valve opening/closing duration are also set as variable parameters. Note that these two process parameters correspond to the 'Open Time' (line 226) and 'Transfer Volume' (line 1669), respectively, and these are modified by the VB program. Overall, these three variables were assigned discrete values, as summarized in Table 2, resulting in a total of 2,860 candidate experimental conditions. In this experiment, L was set to 2.4 mm. Since the Bayesian optimization method PHYSBO in NIMO proposes experimental conditions that maximize the objective function, the objective function defined as Eq. (1) was defined to achieve its maximum value of 0 when $\bar{L} = L$.

3.2. Optimization results

Our closed-loop system enables high-throughput experiments at a rate of 20 samples per hour. Initially, 10 conditions were selected, randomly. Subsequently, 220 cycles of Bayesian optimization were performed. For comparison, another set of experiments was conducted in which all 230 cycles were randomly selected by NIMO. The transitions of the objective function (O) defined in Eq. (1) across iteration cycles are shown in Figure 5(a).

Table 2. Control parameters for dispensing. For steady-state pressure, open time, and concentration, 13, 10, and 11 discrete values were selected as possible options, respectively. The total number of experimental conditions is $13 \times 20 \times 11 = 2860$.

Steady-state pressure	Open time	Concentration
0.3 nL	50 μs	0%
0.4 nL	100 μs	2.5%
0.5 nL	150 μs	5%
0.6 nL	200 μs	7.5%
0.7 nL	250 μs	10%
0.8 nL	300 μs	12.5%
0.9 nL	350 μs	15%
1.0 nL	400 μs	17.5%
1.1 nL	450 μs	20%
1.2 nL	500 μs	22.5%
1.3 nL	550 μs	25%
1.4 nL	600 μs	
1.5 nL	650 μs	
	700 μs	
	750 μs	
	800 μs	
	850 μs	
	900 μs	
	950 μs	
	1000 μs	

The results indicate that Bayesian optimization identified dispensing parameters that achieve uniform droplets with fewer experiments than random sampling. The histogram of the objective function values for the 230 experiments, presented in Figure 5(b), further demonstrates that the Bayesian optimization identified numerous dispensing parameters with high objective function values. These findings confirm that the Bayesian optimization effectively controls the self-driving liquid handling system.

The distributions of the dispensing parameters and the average of droplet diameters $\bar{L}$ for the experimental results obtained via Bayesian optimization are shown in Figure 5(c). Pearson's correlation coefficients are also provided, revealing a negative correlation between PEG concentration and $\bar{L}$. This suggests that increasing $\bar{L}$ requires a reduction in PEG concentration, a result consistent with the relationship between viscosity and $\bar{L}$, that is, lower viscosity results in larger droplets. Conversely, the correlations between steady-state pressure, open time, and $\bar{L}$ were minimal, with steady-state pressure showing a correlation coefficient close to zero. These findings indicate that for PEG samples, droplet uniformity on the water-sensitive paper is influenced more significantly by the liquid properties than by the operating parameters of the dispenser. On the other hand, in general, when handling high-viscosity liquids, increasing both the pressure and the open time is expected to result in larger $\bar{L}$ upon impact with water-sensitive paper. In practice, when viscosity is kept constant, a positive correlation between open time and $\bar{L}$ has been observed. For example, at a concentration of 0%, the Pearson correlation coefficient between open time and $\bar{L}$ is over 0.55. For higher-viscosity PEG samples, it

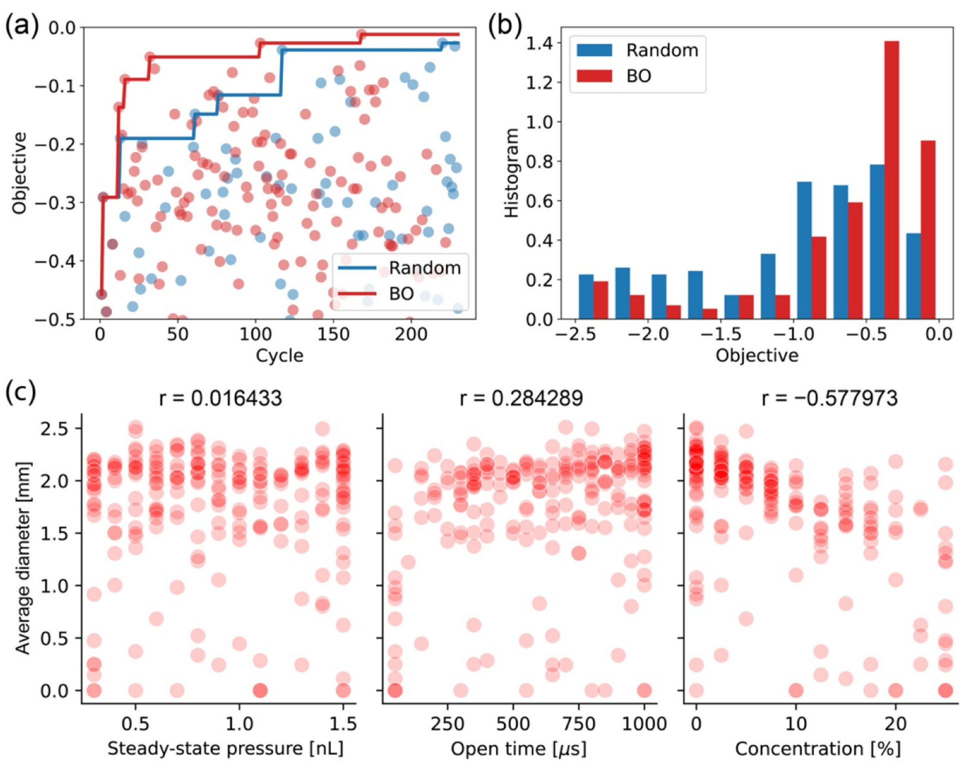


Figure 5. Results of autonomous automated liquid handling. (a) Objective function values plotted as a function of iteration cycle for random sampling and Bayesian optimization (BO). The lines represent the maximum objective function values observed up to each cycle. (b) Histograms showing the distribution of objective function values obtained through random sampling and BO. (c) Average droplet diameter $\bar{L}$ plotted as a function of steady-state pressure, open time, and PEG concentration. Pearson's correlation coefficients (r) are also provided to indicate the strength and direction of the relationships.

might be possible to achieve larger $\bar{L}$ by extending the open time beyond the current upper limit of 1000 μs in the exploration range. However, even when concentration is fixed, no correlation is observed between pressure and $\bar{L}$, indicating that pressure is not an influential parameter in the current exploration problem.

4. Conclusion

In this study, we explored the integration of AI into an existing automated experimental system using NIMO. By facilitating the exchange of input and output files, NIMO enables the addition of AI without disrupting the workflow of the robotic experimental system. We demonstrated the feasibility of constructing a self-driving system by incorporating AI into an automated liquid handling system controlled by a Visual Basic (VB) program. Using NIMO's 'STAN' mode, we successfully synchronized the Python-based AI program in NIMO with the VB-controlled liquid handling system. This approach highlights the versatility of NIMO, which allows seamless integration of AI into existing robotic experimental systems programmed in non-Python languages such as VB and LabVIEW, as well as self-driving laboratories (SDLs) managed by other orchestration software (OS) platforms. We believe that the incorporation of diverse AI methodologies implemented in NIMO will significantly enhance the capabilities and efficiency of SDLs, paving the way for advancements in experimental research and innovation.

Author contributions

R.T., H.T., and S.Ma. conceived the idea and designed the research. H.T. performed the experiments. S.Mu., D.R., and T.Y. developed Visual Basic program. R.T., K.T., and S.Ma. developed the NIMO package. All members contributed to the preparation of the manuscript.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by Ministry of Education, Culture, Sports, Science, and Technology (MEXT) Program: Data Creation and Utilization Type Materials Research and Development Project [Grant Number JPMXP1121467561]. R.T. was partially supported by the JST, PRESTO [Grant Number JPMJPR24T8].

Data availability statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

References

[1] Shimizu R, Kobayashi S, Watanabe Y, et al. Autonomous materials synthesis by machine learning and robotics. APL Mater. 2020;8(11):111110. doi: 10.1063/5.0020370

[2] Burger B, Maffettone PM, Gusev VV, et al. A mobile robotic chemist. Nature. 2020;583(7815):237–241. doi: 10.1038/s41586-020-2442-2

[3] Cao L, Russo D, Felton K, et al. Optimization of formulations using robotic experiments driven by machine learning DoE. Cell Rep Phys Sci. 2021;2(1):100295. doi: 10.1016/j.xcrp.2020.100295

[4] Li J, Li J, Liu R, et al. Autonomous discovery of optically active chiral inorganic perovskite nanocrystals through an intelligent cloud lab. Nat Commun. 2020;11(1):2046. doi: 10.1038/s41467-020-15728-5

[5] Matsuda S, Lambard G, Sodeyama K. Data-driven automated robotic experiments accelerate discovery of multi-component electrolyte for rechargeable Li–O_2 batteries. Cell Rep Phys Sci. 2022;3(4):100832. doi: 10.1016/j.xcrp.2022.100832

[6] MacLeod BP, Parlane FGL, Rupnow CC, et al. A self-driving laboratory advances the Pareto front for material properties. Nat Commun. 2022;13(1):995. doi: 10.1038/s41467-022-28580-6

[7] Volk AA, Epps RW, Yonemoto DT, et al. Alphaflow: autonomous discovery and optimization of multi-step chemistry using a self-driven fluidic lab guided by reinforcement learning. Nat Commun. 2023;14(1):1403. doi: 10.1038/s41467-023-37139-y

[8] Szymanski NJ, Rendy B, Fei Y, et al. An autonomous laboratory for the accelerated synthesis of novel materials. Nature. 2023;624(7990):86–91. doi: 10.1038/s41586-023-06734-w

[9] Tom G, Schmid SP, Baird SG, et al. Self-driving laboratories for chemistry and materials science. Chem Rev. 2024;124(16):9633–9732. doi: 10.1021/acs.chemrev.4c00055

[10] Sheng H, Sun J, Rodríguez O, et al. Autonomous closed-loop mechanistic investigation of molecular electrochemistry via automation. Nat Commun. 2024;15(1):2781. doi: 10.1038/s41467-024-47210-x

[11] Šiaučiulis M, Knittl-Frank CM, Mehr SH, et al. Reaction blueprints and logical control flow for parallelized chiral synthesis in the Chemputer. Nat Commun. 2024;15(1):10261. doi: 10.1038/s41467-024-54238-6

[12] Ramprasad R, Batra R, Pilania G, et al. Machine learning in materials informatics: recent applications and prospects. npj Comput Mater. 2017;3(1):1–13.

[13] Butler KT, Davies DW, Cartwright H, et al. Machine learning for molecular and materials science. Nature. 2018;559(7715):547–555. doi: 10.1038/s41586-018-0337-2

[14] Terayama K, Sumita M, Tamura R, et al. Black-box optimization for automated discovery. Acc Chem Res. 2021;54(6):1334–1346. doi: 10.1021/acs.accounts.0c00713

[15] Back S, Aspuru-Guzik A, Ceriotti M, et al. Accelerated chemical science with AI. Digit Discov. 2024;3(1):23–33. doi: 10.1039/D3DD00213F

[16] Welcome to Chemios! — Chemios framework 0.1.0 documentation [internet]. [cited 2025 May 4]. Available from: https://chemios.readthedocs.io/en/latest/?

[17] Roch LM, Häse F, Kreisbeck C, et al. ChemOS: an orchestration software to democratize autonomous discovery. PLOS ONE. 2020;15(4):e0229862. doi: 10.1371/journal.pone.0229862

[18] Deneault JR, Chang J, Myung J, et al. Toward autonomous additive manufacturing: Bayesian optimization on a 3D printer. MRS Bull. 2021;46(7):566–575. doi: 10.1557/s43577-021-00051-1

[19] Rahmanian F, Flowers J, Guevarra D, et al. Enabling modular autonomous feedback-loops in materials science through hierarchical experimental laboratory automation and orchestration. Adv Mater Interface. 2022;9(8):2101987. doi: 10.1002/admi.202101987

[20] Rahmanian F, Flowers J, Guevarra D, et al. Flowchem 1.0.0a3 documentation [internet]. [cited 2025 May 4]. Available from: https://flowchem.readthedocs.io/en/latest/index.html#

[21] Tamura R, Tsuda K, Matsuda S. NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science. Sci Technol Adv Mater Meth. 2023;3(1):2232297. doi: 10.1080/27660400.2023.2232297

[22] Hielscher MM, Dörr M, Schneider J, et al. LABS: laboratory automation and batch scheduling – a modular open source Python program for the control of automated electrochemical synthesis with a web interface. Chem Asian J. 2023;18(14):e202300380. doi: 10.1002/asia.202300380

[23] Sim M, Vakili MG, Strieth-Kalthoff F, et al. ChemOS 2.0: an orchestration architecture for chemical self-driving laboratories. Matter. 2024;7(9):2959–2977. doi: 10.1016/j.matt.2024.04.022

[24] Yoo HJ, Lee K-Y, Kim D, et al. OCTOPUS: operation control system for task optimization and job parallelization via a user-optimal scheduler. Nat Commun. 2024;15(1):9669. doi: 10.1038/s41467-024-54067-7

[25] Fei Y, Rendy B, Kumar R, et al. AlabOS: a Python-based reconfigurable workflow management framework for autonomous laboratories. Digit Discov. 2024;3(11):2275–2288. doi: 10.1039/D4DD00129J

[26] Zhang W, Hao L, Lai V, et al. IvoryOS: an interoperable web interface for orchestrating Python-based self-driving laboratories. Nat Commun. 2025;16(1):5182. doi: 10.1038/s41467-025-60514-w

[27] Motoyama Y, Tamura R, Yoshimi K, et al. Bayesian optimization package: PHYSBO. Comput Phys Commun. 2022;278:108405. doi: 10.1016/j.cpc.2022.108405

[28] Terayama K, Sumita M, Tamura R, et al. Pushing property limits in materials discovery via boundless objective-free exploration. Chem Sci. 2020;11(23):5959–5968. doi: 10.1039/D0SC00982B

[29] Terayama K, Tamura R, Nose Y, et al. Efficient construction method for phase diagrams using uncertainty sampling. Phys Rev Mater. 2019;3(3):033802. doi: 10.1103/PhysRevMaterials.3.033802

[30] Tamura R, Deffrennes G, Han K, et al. Machine-learning-based phase diagram construction for high-throughput batch experiments. Sci Technol Adv Mater Meth. 2022;2(1):153–161. doi: 10.1080/27660400.2022.2076548

[31] Tamura R, Morito H, Deffrennes G, et al. AIPHAD, an active learning web application for visual understanding of phase diagrams. Commun Mater. 2024;5(1):1–11. doi: 10.1038/s43246-024-00580-7

[32] Opentrons Labworks Inc [Internet]. Opentrons.com. [cited 2025 Aug 11]. Available from: https://opentrons.com