Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2025, **4**, 1083

Robotic integration for end-stations at scientific user facilities†

Chandima Fernando, Hailey Marcello, Jakub Wlodek, John Sinsheimer, Daniel Olds, Stuart I. Campbell and Phillip M. Maffettone *

The integration of robotics and artificial intelligence (AI) into scientific workflows is transforming experimental research, particularly at large-scale user facilities such as the National Synchrotron Light Source II (NSLS-II). We present an extensible architecture for robotic sample management that combines the Robot Operating System 2 (ROS2) with the *Bluesky* experiment orchestration ecosystem. This approach enabled seamless integration of robotic systems into high-throughput experiments and adaptive workflows. Key innovations included a client-server model for managing robotic actions, real-time pose estimation using fiducial markers and computer vision, and closed-loop adaptive experimentation with agent-driven decision-making. Deployed using widely available hardware and open-source software, this architecture successfully automated a full shift (8 hours) of sample manipulation without errors. The system's flexibility and extensibility allow rapid re-deployment across different experimental environments, enabling scalable self-driving experiments for end stations at scientific user facilities. This work highlights the potential of robotics to enhance experimental throughput and reproducibility, providing a roadmap for future developments in automated scientific discovery where flexibility, extensibility, and adaptability are core requirements.

Received 24th January 2025
Accepted 17th March 2025

DOI: 10.1039/d5dd00036j

rsc.li/digitaldiscovery

1 Introduction

Robotics is revolutionizing scientific discovery and science-at-scale across diverse disciplines, including biomedical,[1–3] earth,[4] and materials sciences.[5–7] Specifically, the combination of automation and artificial intelligence (AI) has enabled scientific experiments to be conducted faster, more safely, more accurately, and with greater reproducibility.[7–11] Large scientific user facilities, such as the National Synchrotron Light Source II (NSLS-II), provide prime examples for the potential of transforming scientific laboratories into automated factories of discovery,[12–15] and rely on robotic systems for high throughput and sensitive experiments.[16–19] However, contemporary robotic integrations lack the flexibility required to cater to diverse experimental environments, do not include dynamic decision-making capabilities, and are weakly integrated into the experimental orchestration software stack.[20] This creates a substantial opportunity for new architectures that integrate widely used experimental orchestration suites with robotics and AI, and those that provide flexibility, extensibility, and adaptability.

Deploying robotics inside or alongside the unit operations of scientific experiments—or 'end stations' in the parlance of user facilities—has been explored as a potential solution for increasing scientific throughput, efficiency, and safety.[7–11,16] These deployments include robotic arms at the core of workcells,[21,22] mobile robots as integrators between various unit operations,[23] and active cooperation between cobots and human researchers.[7,24] This latter direct cooperation is not possible in certain environments due to safety constraints, albeit robotic arms can integrate with other actuation for mobility within protected spaces.[16] Many of the existing approaches depend on vendor supplied software for robot orchestration[6,7,17–19,23] and develop *ad hoc* tooling to combine robotics and existing equipment.[1,22,25] We recently demonstrated the use of the Robotic Operating System 2 (ROS2) for facile integration of a pick-and-place robot solution into an existing end station;[16] however, there remain limitations in flexibility, adaptability to failure, and extensibility to new experiments.

With these advancements in automation, there is a growing motivation in the scientific community for autonomous, or self-driving, experiments that leverage artificial intelligence in their operational and scientific decision making.[20] Robots can leverage AI agents in their internal planning, for example in path planning[26] or environment recognition.[4] At a higher level, these systems can also leverage agents for choosing their next experiment.[27] As such, when integrating robots into experiment orchestration, it is crucial to ensure the availability of algorithmic integrations at varying levels of decision making.

National Synchrotron Light Source II, Brookhaven National Laboratory, Upton, NY 11973, USA. E-mail: pmaffetto@bnl.gov

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d5dd00036j

An optimal arena for the development of scientific experiments driven by smart robotics is provided by large scientific user facilities, such as the NSLS-II. The NSLS-II is a light source that offers state-of-the-art capabilities for probing the structural and electronic properties of materials at atomic and microscopic scales.[13] As next-generation light sources have continually increased the flux of their facilities, attention has turned toward experimental orchestration that can make the most effective use of these photon beams through advanced automation and AI techniques. Experiment end stations at NSLS-II (called beamlines) that offer routine techniques at scale represent a strategic evolution towards leveraging these advanced capabilities for high-throughput characterization tasks. Building on this concept, Beamline as a Service (BaaS) has emerged as a pioneering model, re-imagining how researchers interact with and utilize the facilities at a synchrotron.[20] A BaaS framework envisions a network of self-driving beamlines, equipped with core information technologies, robotics, and agentic AI that can operate autonomously or in concert with human researchers. This potential ecosystem will maximize resource utilization and collaboration to catalyze advancements in energy and sustainability. Nonetheless, this will require substantive effort in technology development, and even more so in technology integration.

Given the complexity and diversity of scientific experiments conducted at NSLS-II, the *Bluesky* project‡ was developed as an open-source ecosystem offering unparalleled capabilities for orchestration, data management, and analysis.[28] This project can be utilized piecemeal and has increasing adoption globally, including across all six U.S. Department of Energy light and neutron sources. At its core, the *Bluesky* RunEngine coordinates intricate experimental workflows, while complementary packages like *Ophyd*, *Tiled*, and *Bluesky Adaptive* enable seamless instrument integration, advanced data management, and adaptive experimentation, respectively.[29] Its intuitive Python interface empowers researchers to design and execute experiments that were previously infeasible, including those which leverage robotics.[16] Furthermore, the project commitment to open-source development and community-driven innovation ensures that it remains at the forefront of scientific software. By bridging cutting-edge tools with accessible design, *Bluesky* not only enhances experimental efficiency but also opens new opportunities to deploy computational agents, drive adaptive science, and tackle the nonlinear challenges of discovery in materials science and beyond.

Considered in combination, these advancements in robotics for self-driving laboratories, the cutting edge infrastructure of large scientific user facilities, and the development of experimental orchestration tooling have created substantial opportunities for accelerating scientific discovery, albeit with some unresolved limitations. Robotic deployments at user facilities must be reconfigurable, extensible, and robust to slight variations in environments. To achieve this, they must interface simply with existing experiment orchestration and other contemporary technologies (*e.g.*, AI). These open challenges are largely related to the daunting task of integrating complex tools as opposed to the development of new tools.

Herein, we describe a generic and extensible architecture for the flexible deployment of robotic sample management at experimental end stations using *Bluesky*. We combined ROS2 control with Python abstractions that provide seamless integration of a robotic arm into *Bluesky* orchestration. We leveraged a development pipeline that includes simulation and test environments for flexibility and extensibility to new end stations. We then extended this application with computer vision and a sample database to ensure adaptability to failure and variable environments. Lastly, we closed the experimental loop by leveraging the tooling of *Bluesky Adaptive* for autonomous agent integrations in self-driving experiments. Our unique contributions in this work include the extension of a ROS-*Bluesky* interface to include real-time feedback and interruption capability, the integration of computer vision and sample management in the application, and the first demonstrated integration of *Bluesky Adaptive* with a robotic system for autonomous experiment execution. This solution has immediate implications at the many facilities deploying *Bluesky*, brings the community closer to the BaaS vision, and provides a road-map for other researchers or facilities seeking to leverage robotics to accelerate scientific research.

2 Software architecture for robotic sample management

The software architecture we developed depends on functionality from the ROS and *Bluesky* ecosystems, as well as widely available database and computer vision technologies.

2.1 ROS integration with *Bluesky*

The foundational innovation necessary to leverage robotics effectively at user facilities, was the harmonization between robotic control and end station orchestration software. To this end, we first integrated ROS2 driven robotics into the *Bluesky* project. At its core, this widely used platform for experiment orchestration serves as the nervous system for scientific workflows, enabling researchers to define, execute and monitor experiments through a high-level, intuitive Python interface. We performed the integration at the hardware abstraction layer, *Ophyd*, which is used to define devices such as motors, detectors, sensors, and state controllers. With the robotic service integrated in *Ophyd*, the robotic task could be integrated into the orchestration of other devices of the end station using the RunEngine. We successfully evaluated this integration in a simulated environment, multiple test environments, and at the Pair Distribution Function (PDF) beamline at NSLS-II.

The *Ophyd* integration primarily depends on the use of Python's Future objects for asynchronous applications. In recent work, we demonstrated how the Future of a ROS2 Action could be exchanged for an *Ophyd* Status object to enable the RunEngine (akin to a ROS2 executor) to conduct an experiment that combines an existing end station and a new robot application (*e.g.*, sample

‡ https://blueskyproject.io.

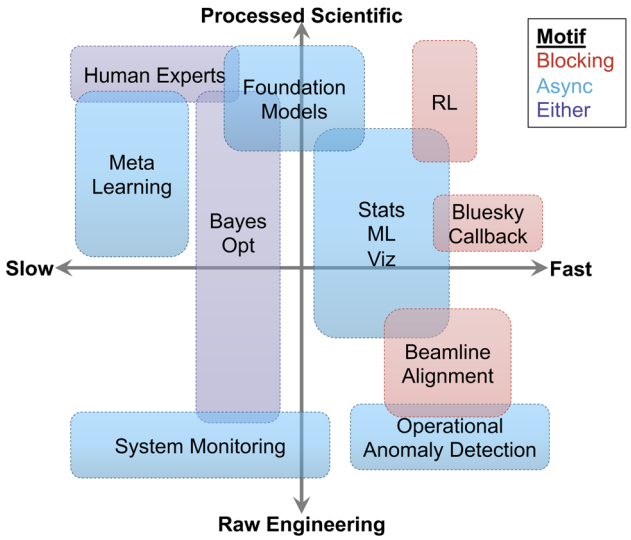

Fig. 1 Axes of Adaptivity from *Bluesky Adaptive* documentation. The three conceptual axes of adaptive experimental orchestration, using color to show processing motif. The *y*-axis considers the degree of processing necessary for an agent to consume data, varying between raw sensor current and scientifically relevant state variables. The *x*-axis considers the rate of the decision making with respect to the experiment. These axes were used in the conceptual design of *Bluesky Adaptive*. Several approaches and application areas for agents are shown as examples, indicating where they may fall on these axes.

management).[16] In the work described herein, we extended our previous developments to use Action feedback to create progress visualization of the completion of a robot action, and to enable the RunEngine to "interrupt" an ongoing experimental plan. In this case, the Actions were equipped with logic to pause, stop, abort, cleanup, and rewind to a recent state. In our example of sample management, this empowers the user to change their mind while a robot is loading a selected sample, pause, return the chosen sample if it is already grasped, then select a new sample.

To leverage robots for self-driving experiments in *Bluesky* we depended on the *Bluesky Adaptive* package. This is an actively developed and growing component of the *Bluesky* ecosystem, designed to provide a harness for adaptivity and intelligent decision making in experiment workflows. At its core, *Bluesky Adaptive* provides a flexible API that supports a spectrum of adaptive algorithms, ranging from simple rule-based approaches to complex AI-driven models. Primarily for enabling experiments to dynamically respond to data and adjust measurement strategies in real-time, it also accommodates non-interventional agents that process data and generate visualizations to guide researchers without directly controlling the experiment. To categorize the varying levels of adaptivity achievable within this framework, we conceptualize adaptive behavior along three key axes—decision-making rate, degree of signal abstraction, and processing modality [Fig. 1]. Up-to-date details on the design, implementation, and instructions for use can be found in the online documentation.§ Herein, we implemented a simple agent that consumed data and performed a random walk based on that data using the *Bluesky Adaptive* harness.

§ **https://blueskyproject.io/bluesky-adaptive**.

2.2 Client-server architecture

Beyond the initial integration of the powerful open source tools, ROS2 and *Bluesky*, we developed a generic client-server architecture for intelligent sample management at the beamline. While ROS2 provides a number of architectural styles—namely publisher-subscriber and client-server models—through their available primitives, here we expand on their asynchronous client-service model provided by ROS2 Action nodes. Our client is both a ROS2 Action client and a *Bluesky* RunEngine client, which maintains its own sample database and manages the higher level experiment orchestration. The primary server manages robotic tasks, and contains the Action service for the robot orchestration, alongside other ROS2 Action and Service nodes for object pose estimation, grasping, and kinematics [Fig. 2].

The client integrates ROS2 Action-client nodes as *Ophyd* objects, and provides the overall experiment orchestration. This request for a robot action and subsequent data acquisition is facilitated by the *Bluesky* RunEngine. The next subsequent sample to be measured is suggested by *Bluesky Adaptive*—a library from the *Bluesky* ecosystem designed for intelligent and adaptive decision-making—based on the previous measurements.[29] The proposed sample from *Bluesky Adaptive* is then queried against a database containing sample information to retrieve the corresponding ID for the pose estimation process. Together, data acquisition, database integration, and sample proposals complete the feedback loop between the *Bluesky Adaptive* agent and the RunEngine execution. The final step in the client-side operations is packaging the sample ID into the ROS2 Action goal message, which is sent to the Action server.

We implemented three key functionalities in the design of the Action server. Our design allowed the cancellation of a goal

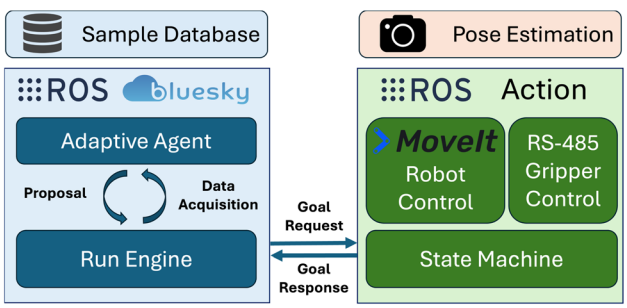

Fig. 2 Architecture schematic for robotic beamline scientist. Primarily a client-server architecture, the approach to integrating the robotic beamline scientist into the experiment orchestration with *Bluesky* leveraged a sample database, ROS2 Action and pose estimation service. The *Bluesky* RunEngine orchestrates the overall experiment in a closed loop, where an adaptive agent provides recommendations for the next experiment and the Action server orchestrates the robotic subsystem.

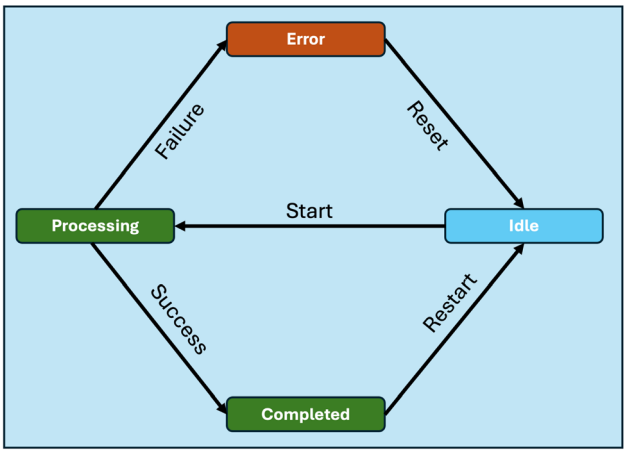

Fig. 3 A generic finite state machine. This demonstrative example shows how a FSM can be used for process management. Starting in an 'Idle' state, it undergoes a 'Start' transition to a 'Processing' state. 'Success' or 'Failure' of the process will lead to different transitions to 'Completed' or 'Error' states, respectively. Finally, each penultimate state has a distinct transition back to the 'Idle' state, and the process repeats.

during execution, tracked the progress of goal execution, and reported updates back to the client. To accomplish this, we used a finite state machine (FSM) to orchestrate the pick-and-place sequence for a given sample [Fig. 3]. We integrated robot control with this state machine and broke it into components that included robot arm path planning, pose estimation, and gripper control. We used MoveIt,¶ a ROS2 library for robot motion planning and execution, alongside ROS2 drivers provided by the robot component manufacturers. We further developed the gripper control module using ROS2 Service nodes, enabling the gripper to operate synchronously with the state machine while transitioning its state based on real-time gripper status feedback. We handled sample pose estimation with a separate Service node that interfaced with the state machine through ROS2 communication. Each ROS2 node in this architecture was launched in a separate container to provide robust availability and process management.

Several types of messages and information are passed inside this architecture. Primarily there is the ROS message between the primary client and server describing the overall goal of the robot action. This message contained the robot pose for approaching and placing the sample in the end station, a boolean flag for whether the sample was being placed or returned to the library, and a sample ID. The communication to the client included an update of the fractional completion of the pick-and-place task, and whether it completed successfully. Without computer vision, the pose for the robot to approach and pickup the sample were also included. Messages for MoveIt services were abstracted using the built-in "Move Group Interface". Messages for the gripper contained the percentage of the range of motion for the gripper to open or close to. Within *Bluesky* information is passed to the RunEngine using Python generator coroutines, and to the adaptive agents using the document model prescribed by *Bluesky*.

¶ https://moveit.ai/.

2.3 Finite state machine

We chose to break down the act of grasping a chosen sample into manageable, defined sub-tasks using a FSM [Fig. 4]. This offered several advantages in both development and deployment. States track and communicate the robot's progress, enabling swift identification of failures during an experiment. A mapping between RunEngine commands and the state machine's states enhanced usability, allowing seamless control with directives and interruptions (*e.g.*, user orders like 'Pause', 'Abort', or 'Rewind'). In case of a failure, RunEngine commands could transition the system back to the Home state aided by a user-desired sequence of actions using the state diagram. This functionality was designed to meet the protocols outlined in the *Bluesky* documentation, and further supported by six internal states for each primary state in the state machine: RESTING, MOVING, PAUSED, ABORT, HALT, STOP, CLEANUP. In order to extend the below FSM to modified or new tasks, additional states and transitions would need to be added to the C++ implemented Enums and case statements for transitions.

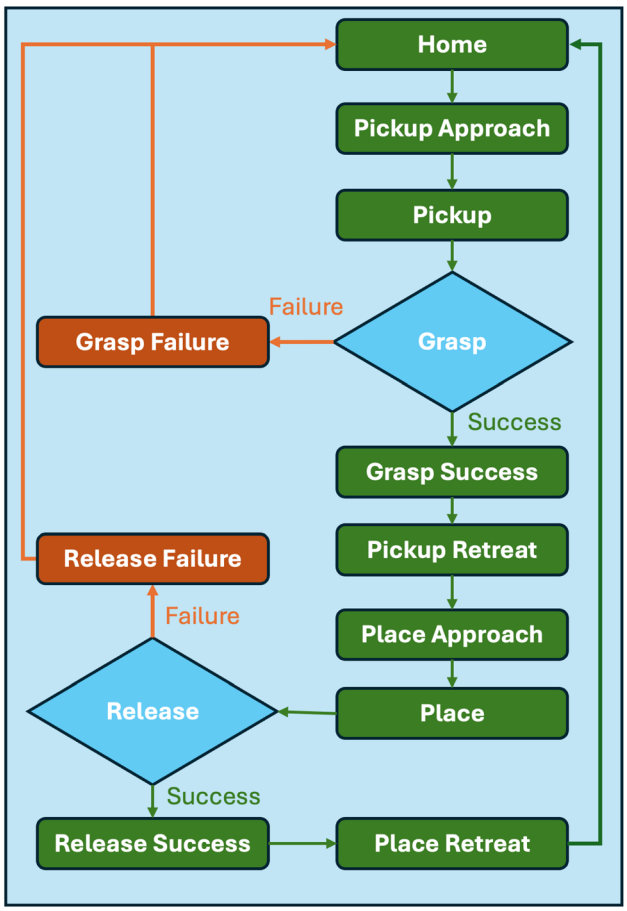

Fig. 4 The robot Action progression used a finite state machine to track progress. This FSM provided dynamic feedback to the *Bluesky* RunEngine for progress updates and synchronization management.

The state machine was designed to provide an intuitive pick-and-place process that was configurable and flexible for multiple end-stations. Each state transition is contingent upon the successful execution of the motion planning and execution of the robot trajectory. Instances of failure break the state propagation, and a failure message is communicated back to the action client. The 12-state FSM begins in a predetermined Home pose, which is configurable through ROS2 node parameters, that provides a safe robot position for other actuation at the end station and opens the gripper.

Upon accepting a goal request from the client, the Action server begins to execute the FSM, re-initializing to Home then progressing to Pickup Approach. State Pickup Approach was designed to reduce path planning complexity in approaching the library of available samples. When using computer vision to determine the location of the desired sample, this state determines the field of view for the camera. Upon successful completion of the trajectory, and successful sample pose return by the Pose Estimation Service, the FSM transitions to the Pickup state. In this state, the robot moves to a pose where the gripper fingers are orthogonal to the face of the sample, then moves such that the gripper surrounds the sample. Success here triggers a grasp attempt closing the gripper fingers, that progresses the FSM to a specific Grasp Success or Grasp Failure state depending on the closure of the gripper. Grasp success was determined by achieving the desired fraction of closure, *e.g.*, if the gripper was to close too much or too little this would be considered a failure. In the future, this could be extended to include visual confirmation or other signal fusion. At any point prior to this grasp, the user has the option to change course, halt the FSM, and return to Home.

Following a successful grasp of the sample, the FSM transitions the robot back to the Pickup Approach pose in a new state Pickup Retreat. The placing of the sample into the experiment apparatus, in our case the X-ray beam path, then follows a similar series of states as picking up the sample. The Place Approach state uses a trajectory in joint space to prepare the sample for placement, and Place uses a cartesian path to align the sample into the apparatus. This location is configured as a static location similarly to the Home state, and can be rapidly reconfigured using the ROS parameters at startup. Both of these parameters are meant to be relatively static throughout a deployment, as such are loaded at startup and not reloaded at each relevant state execution in the FSM. Adjusting the parameters required a reboot of the Action server container. Other parameters such as the obstacles in the environment could be modified dynamically and would update the planning scene through callbacks. A Release operation by default assumes success as long as the gripper opens; however, could include additional validation using sensors at the end station. Lastly, Place Retreat moves the robot to a position where it can clear the geometry of the sample in its path planning to return to Home.

At this stage, Action execution is finalized and the client can continue the rest of the experiment as orchestrated in *Bluesky*. In our development environments, this involved a simulated detector taking noise. At the PDF beamline, this involved moving the platform that the robot was mounted on such that the robot would not interfere with the X-ray beam, then progressing with the automated data collection on the sample [Fig. S1†]. Following the experiment execution, the sample could be returned back to storage using a new Action goal from the client. The same FSM drives this object manipulation, with the Pickup and Place poses reversed.

3 Experiment results and discussion

3.1 Hardware configuration

In this work we used a Universal Robot, UR3e, 6 degree of freedom robotic arm. This robot was chosen for its compatibility with ROS2 and available simulator. For an end effector, we used a Robotiq Hand-E gripper, which has two fingers and controls grip strength and grasp ratio. We designed and 3-d printed a custom coupling to mount the camera (described below) above the gripper. The gripper was powered by a pass-through connection to the robot arm and interfaces *via* an RS483 interface. The camera required the use of external power.

We used computer vision and fiducial makers for pose estimation of the sample holder. We deployed an Azure Kinect DK 12-megapixel camera with a 1-megapixel Time of Flight (ToF) depth camera. Captured images had a 640 × 576 pixel resolution and a field of view of 75° × 65°. We deployed all services as containers on virtual machines within a common subnet with the exception of the camera services. We connected the camera to a physical server (System 76 Meerkat computer powered by 13th gen Intel processor and 64GB of memory) to accommodate its USB interface. The fiducial markers were from the marker family DICT_APRILTAG_36h11 in the ArUco class of markers, and were affixed to a common relative location on all samples, and printed to at 26.65 cm in width and height. DICT_APRILTAG_36h11 family of makers are compatible with both ArUco pose detection and AprilTag pose detection.[30,31] The samples we used for this study were brackets that hold varying sizes and counts of capillaries for the PDF beamline. The brackets at the beamline were made of machined aluminum, and those at the experimental testbed were 3-d printed to match this form factor.

3.2 Deployment pipeline

The software and hardware setup facilitated an auxiliary goal of a short deployment and re-deployment time into the experiment end station. This process historically could take days of effort, squandering precious resources of time with the synchrotron beam.[16] We further facilitated the rapid deployment across different environments with minimal modifications to the existing setup by using generalized development configurations including simulations and tabletop development environments. Fig. 5 illustrates the mock up of two sample environments for two distinct end-station configurations and their projection within the RViz visualization tool.[32] The first development environment used pre-defined and static locations for where to find and load the samples. The second setup leveraged computer vision to detect and define the sample

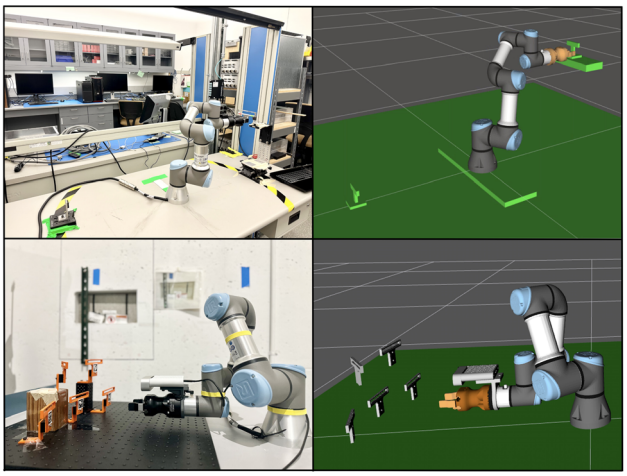

Fig. 5 Tabletop environment and RViz simulation environment. The combination of a vendor provided digital twin, RViz simulation, and tabletop test environments empowered distributed teams to develop applications without the use of valuable facility resources. Each row shows two different laboratory test setups in the development pipeline and their associated simulations.

poses in real time. We visualized both configurations in RViz during development to identify obstacles or constraints, fine-tune the system, and debug as necessary. Lastly, we leveraged containerization to make the solution portable across development environments, distributed across resources, and secure within facility infrastructure. In combination, this development pipeline enabled our team of engineers to develop solutions outside of the beamline environment, then deploy, remove, or re-deploy the robot in a few person-hours.

3.3 Software configuration

We estimated the pose of the samples in real-time by affixing fiducial markers detected and described with computer vision. The ArUco fiducial markers were associated unique identifiers that could be mapped back to the sample database. Next, we implemented pose estimation inside a ROS2 node using OpenCV, leveraging its ArUco library integration. The 6-d pose estimation was calculated for the sample of interest from the stream of image data. This pose includes the distance from the camera to the fiducial marker along the x, y, and z axes, as well as the roll, pitch, and yaw rotations of the fiducial marker. We passed the estimated poses through a median filter to eliminate any outliers in the stream that could be caused by the camera jitter. The ROS2 transform server (tf2 library) automatically provided updates to the coordinate reference frames and transformations relating the samples, camera, gripper, and robot joints. The resulting camera view and projection of the estimated poses in the RViz visualization are shown in Fig. 6. In this scene the robot is in the Pickup Approach state, where pose estimation occurs.

We then closed the experimental loop using *Bluesky Adaptive* in combination with a sample database. From the pose estimation, the *Bluesky*-ROS client could request any sample be

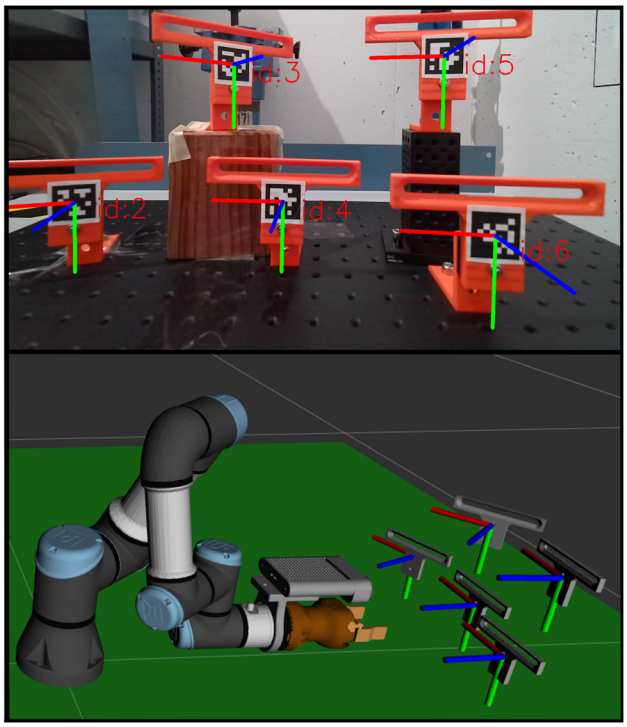

Fig. 6 Pose detection live. The robot views the sample library and dynamically detects the sample data and coordinate reference frames for the sample holders (top). This information can be added to the RViz visualization (bottom).

loaded by referencing a unique identifier, then perform a measurement on that sample. We built an in-memory sample database using Redis‖ to store sample data in relation to the ArUco identifier, so that the resulting *Bluesky* plan could reference the ArUco tag directly. We defined the measurement as a *Bluesky* plan, and used a simulated detector to create data in our tabletop setup. Lastly, we used *Bluesky Adaptive* to integrate a procedure that consumed data and decided which sample to measure next. For this, we used a lockstep approach in *Bluesky Adaptive* that placed the agent in-process with the RunEngine. In this approach the RunEngine piped it's document stream to the agent, which consumed images from the detector, and requested the next sample to measure based on that image data. Thus, we closed the loop in a workflow that measured a sample, chose which sample to measure next, cross referencing that sample to an ArUco identifier, and loading that sample prior to measurement.

The *Bluesky Adaptive* package provides a harness for arbitrary agents to act on data and drive experiments in the *Bluesky* ecosystem.[29] In practice with *Bluesky Adaptive* these agents can be assembled in a simple closed loop, or a large agentic network of communicating decision makers. As the degree of intelligence and type of reasoning can be highly experiment specific,[27] we endowed our experiment with limited intelligence to demonstrate the pipeline integrating a closed loop

‖ https://redis.io/.

measurement. We used a Markov Chain Monte Carlo that consumed the readings that are produced by the *Bluesky* simulated detector, and suggested the next sample to load and measure. The agent transition probabilities were based on a uniform distribution with acceptance probabilities based on a random pixel value. In a production setting, this agent should be endowed with knowledge of the physics or experiment characteristics. This demonstrated the integration of two levels of adaptive decision making in the architecture: first computer vision at a control level, and secondly agent authority at a scientific level.

3.4 Results

We successfully tested this pipeline overnight, performing 195 continuous sample manipulations without error. On a tabletop setup, we arbitrarily placed 5 sample mounts across an M6 grid optical table, consistent with those used in the PDF beamline environment. Each sample mount consisted of a 50 mm long and 25 mm high rail T-slotted framing bracket, secured to the table at various heights and angles in the workspace of the robot. We then secured a 25 mm square kinematic base to the top of each framing bracket, and secured the receiving end to the sample. We affixed a unique ArUco tag to the face opposite the kinematic mount on each sample, aligned consistently for all samples. Lastly, we launched our services across the previously described infrastructure, and launched an adaptive experimental campaign through a Python interface. The self-driving campaign ran overnight without failure, choosing a sample, loading it into a receiving sample mount, conducting a *pseudo* measurement, returning the sample to where it was found, then choosing the next sample based on the measurement.

To evaluate the robustness of our system to variations in sample placement and angular positioning, we conducted a series of repeatability tests using fiducial marker-based pose estimation with an Azure Kinect DK camera. For these tests, the camera was mounted at a fixed position separate from the robot wrist. The sample holders were systematically positioned at 10 cm intervals along the workspace, with varying yaw angles relative to the camera. Successful detections and grasp attempts depended on two primary criteria: (1) the fiducial tag must remain visible to the camera within its field of view, and (2) the sample holder must be oriented within an angle range that allows the robotic arm to approach and grasp it effectively. Grasp failures primarily occurred when samples were placed more than 65 cm from the camera [Fig. S2†]. In such cases, inaccurate pose estimations of occluded fiducial markers resulted in a 50% success rate (2 out of 4 grasps). We also measured the acceptable yaw range that ensured an orthogonal approach for the robot grasp through manual variation, and logged these ranges for each position above [Table S1†]. While fiducial markers provided reliable pose estimation under most conditions, certain fundamental limitations must be noted. Detection accuracy is influenced by occlusion, extreme viewing angles, distortions in the marker's surface, and lighting conditions, which can impact the consistency of marker recognition. On accurate pose estimation, a successful grasp still requires an orthogonal approach by the gripper, thus imposing limits on the sample orientation.

3.5 Discussion of impact and limitations

This campaign demonstrated the full capacity of the architecture described above, and the utility of integrating open-source robotics tools into *Bluesky* orchestrated experiments. We established in previous work that such a deployment can be re-deployed to a beamline in under 2 person hours.[16] By integrating computer vision into the approach, we have further reduced the re-deployment effort required, as samples can be placed arbitrarily within the environment with need no individual alignment, therefore allowing constant time scaling of human effort. We prescribed a form factor for our samples that is flexible and precise, asserting only that the sample has a 25 mm square kinematic base mounted opposite the ArUco tag. This ensured that the samples can be placed in arbitrary positions in the robot workspace, and precisely loaded and unloaded. As each of our samples is designed to hold a sub-library of capillaries or 2-d sample morphologies (*e.g.*, battery coin cells), this system can be extended through effective packing of the robot workspace to conduct a campaign over thousands of measurable samples. The precision afforded by the kinematic base is limited by the consistency of the placement of the fiducial markers, and we suggest embedding the markers directly in the manufacture of the sample holders. Nonetheless, the combination used in our experiments was robust to variation in positioning in the 3-d workspace as well as mounting angles. In summary, this demonstrates a capability for full shifts (8 hours) of self-driving experiments with minimal setup required, providing an opportunity for researchers to run overnight high-throughput or adaptive experiments without limiting the more manual involved research they might want to accomplish during the day.

Our approach to pose estimation and integration into the ROS2 service provides a clear path for extensibility to variable environments and engineered sample morphologies. Nonetheless, there are several opportunities for enhancement in this approach. Firstly, the requirement for a local physical server to execute the camera nodes produced a lethargy in our overall system infrastructure, and a networked camera would be an immediate improvement. We chose to use the Azure Kinect DK to exploit its ToF depth sensing capabilities; however, found that the repeatability in pose estimation using available foundation models was insufficient for our application.[33,34] We expect that as the field advances, there will be new opportunities to replace the fiducial markers. Extending the computer vision capabilities would only involve replacing the algorithm in the Pose Estimation service. At present, the reliability requirements for our systems led us to engineer sample holders and fiducial markers. Given the routine capabilities in additive manufacturing, we see ample opportunities to extend this system to use diverse sample morphologies, and increase the precision of the fiducial marker relative to the grasp point of the sample by printing the marker directly into the sample holder.

Furthermore, as the experiment environment becomes more densely populated with samples, there will be a need to integrate the object Pose Estimation service with the obstacle registration services for dynamic collision avoidance.

While not innovative from an AI algorithm standpoint, our use of *Bluesky Adaptive* created an extensible closed-loop system between samples and experiments that paves way for increasingly complex self-driving measurements at user facilities. The *Bluesky Adaptive* harness is built to be extended to arbitrary levels of decision making complexity, and provides a framework for higher order decision making in *Bluesky* orchestrated experiments [Fig. 1]. The agents built using this harness can also be automatically deployed with an HTTP interface, opening up opportunities for designing effective human-agent interaction, introspection, and control with web tools. Prototype self-driving experimental campaigns have been accomplished using only *Bluesky* orchestrated automation before,[27] albeit this integration of sample databases, computer vision, and robotic sample management with *Bluesky Adaptive* provides a framework for long running campaigns in the BaaS model. Future work at facilities leveraging this technology should focus on extending the diversity of experimental environments that can be integrated with this kind of solution. Currently, the effort required to modify or extend this architecture to new tasks depends on the degree of modification. Performing the same pick-and-place task in a different environment or with a different sample morphology is a trivial change that only involves parametrization. Adjusting the states in the FSM or adding additional states and transitions would require adjustment of the server source code. Nonetheless, there are opportunities to make this approach more easily extensible to completely novel tasks without substantial reprogramming.

4 Conclusions

This study demonstrates the successful integration of robotics into the *Bluesky* experimental orchestration ecosystem, achieving adaptive, high-throughput experimentation at NSLS-II. By harmonizing ROS2 with *Bluesky*, we introduced a client-server architecture for robotic sample management, incorporating real-time computer vision, adaptive decision-making, and modular design principles. The architecture's flexibility enables rapid deployment and re-deployment across diverse experimental setups, reducing human intervention and enhancing experimental efficiency. Robustness tests validated the system's ability to handle extended experimental campaigns, achieving overnight, error-free operation. The implications of this work extend beyond the PDF beamline or NSLS-II, offering a scalable framework of open-source technologies for integrating robotics and AI in scientific research. Future enhancements could include networked camera systems, advanced pose estimation techniques, and the incorporation of diverse sample morphologies to further enhance adaptability and precision. This progress represents a critical step toward realizing the vision of collaborative and [semi]-autonomous laboratories, empowering researchers to conduct more efficient, reproducible, and innovative experiments.

Data availability

The code for this manuscript can be found at **https://github.com/maffettone/erobs/releases/tag/digitaldiscovery** under the BSD-3 license. The version of the code employed for this study is tagged as 'digitaldiscovery'. This code also includes .stl files to describe all 3-d printed components in this study. A built container image for this work is hosted at **https://ghcr.io/nsls2/erobs-common-img:digitaldiscovery**. These resources have been deposited in a persistent repository, Zenodo, under DOI: **https://doi.org/10.5281/zenodo.15025185**.

Author contributions

Chandima Fernando: software (lead); writing – original draft (lead); writing – review and editing (equal). Hailey Marcello: software (supporting); writing – review and editing (equal). Jakub Wlodek: resources (equal); writing – review and editing (equal). John Sinshimer: resources (equal); writing – review and editing (equal). Daniel Olds: resources (equal); conceptualization (supporting); writing – review and editing (equal). Stuart I. Campbell: resources (equal); conceptualization (supporting); writing – review and editing (equal). Phillip M. Maffettone: software (supporting); writing – original draft (supporting); conceptualization (lead); writing – review and editing (equal).

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This research is supported in part by Brookhaven National Laboratory (BNL), Laboratory Directed Research and Development (LDRD) Grant No. 23-039, "Extensible robotic beamline scientist for self-driving total scattering studies". This research also used resources of the PDF (28-ID-1) Beamline and resources of the National Synchrotron Light Source II, a U.S. Department of Energy Office of Science User Facility at BNL under Contract No. DE-SC0012704. This work is supported by the U.S. Department of Energy (DOE) Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences Advanced Scientific Computing Research for DOE User Facilities award "ILLUMINE – Intelligent Learning for Light Source and Neutron Source User Measurements Including Navigation and Experiment Steering". The authors would like to knowledge Edwin Lazo for his insightful conversations surrounding this project, John Trunk for his engineering support, and Richard Faussete for assistance with 3-d printing.

References

1 Y.-H. Lin, D. A. Joubert, S. Kaeser, C. Dowd, J. Germann, A. Khalid, J. A. Denton, K. Retski, A. Tavui, C. P. Simmons, S. L. O'Neill and J. R. L. Gilles, *Sci. Robot.*, 2024, **9**, eadk7913.
2 Y. Yang and P. Jiao, *Mater. Today Adv.*, 2023, **17**, 100338.

3 Y. Jiang, H. Fakhruldeen, G. Pizzuto, L. Longley, A. He, T. Dai, R. Clowes, N. Rankin and A. I. Cooper, *Digital Discovery*, 2023, **2**, 1733–1744.
4 P. Nadan, S. Backus and A. M. Johnson, *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 18480–18486.
5 M. Walker, G. Pizzuto, H. Fakhruldeen and A. I. Cooper, *Digital Discovery*, 2023, **2**, 1540–1547.
6 T. Ha, D. Lee, Y. Kwon, M. S. Park, S. Lee, J. Jang, B. Choi, H. Jeon, J. Kim, H. Choi, H.-T. Seo, W. Choi, W. Hong, Y. J. Park, J. Jang, J. Cho, B. Kim, H. Kwon, G. Kim, W. S. Oh, J. W. Kim, J. Choi, M. Min, A. Jeon, Y. Jung, E. Kim, H. Lee and Y.-S. Choi, *Sci. Adv.*, 2023, **9**, eadj0461.
7 T. Dai, S. Vijayakrishnan, F. T. Szczypiński, J.-F. Ayme, E. Simaei, T. Fellowes, R. Clowes, L. Kotopanov, C. E. Shields, Z. Zhou, *et al.*, *Nature*, 2024, 1–8.
8 A. Angelopoulos, J. F. Cahoon and R. Alterovitz, *Sci. Robot.*, 2024, **9**, eadm6991.
9 D. Caramelli, J. M. Granda, S. H. M. Mehr, D. Cambié, A. B. Henson and L. Cronin, *ACS Cent. Sci.*, 2021, **7**, 1821–1830.
10 S. Back, A. Aspuru-Guzik, M. Ceriotti, G. Gryn'ova, B. Grzybowski, G. H. Gu, J. Hein, K. Hippalgaonkar, R. Hormázabal, Y. Jung, *et al.*, *Digital Discovery*, 2024, **3**, 23–33.
11 R. L. Greenaway, K. E. Jelfs, A. C. Spivey and S. N. Yaliraki, *Nat. Rev. Chem*, 2023, **7**, 527–528.
12 T. Konstantinova, P. M. Maffettone, B. Ravel, S. I. Campbell, A. M. Barbour and D. Olds, *Digital Discovery*, 2022, 413–426.
13 S. Campbell, D. B. Allan, A. Barbour, D. Olds, M. Rakitin, R. Smith and S. B. Wilkins, *Mach. Learn.*, 2020, **2**, 013001.
14 A. Barbour, S. Campbell, T. Caswell, M. Fukuto, M. Hanwell, A. Kiss, T. Konstantinova, R. Laasch, P. Maffettone, B. Ravel, *et al.*, *Synchrotron Radiat. News*, 2022, **35**, 44–50.
15 P. M. Maffettone, S. Campbell, M. D. Hanwell, S. Wilkins and D. Olds, *Cell Rep. Phys. Sci.*, 2022, **3**, 101112.
16 C. Fernando, D. Olds, S. I. Campbell and P. M. Maffettone, *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9578–9584.
17 E. O. Lazo, S. Antonelli, J. Aishima, H. J. Bernstein, D. Bhogadi, M. R. Fuchs, N. Guichard, S. McSweeney, S. Myers, K. Qian, D. Schneider, G. Shea-McCarthy, J. Skinner, R. Sweet, L. Yang and J. Jakoncic, *J. Synchrotron Radiat.*, 2021, **28**, 1649–1661.
18 A. R. Round, D. Franke, S. Moritz, R. Huchler, M. Fritsche, D. Malthan, R. Kläring, D. I. Svergun and M. Roessle, *J. Appl. Crystallogr.*, 2008, **41**, 913–917.
19 D. Y. Ozgulbas, D. Jensen Jr, R. Butler, R. Vescovi, I. T. Foster, M. Irvin, Y. Nakaye, M. Chu, E. M. Dufresne, S. Seifert, *et al.*, *Light: Sci. Appl.*, 2023, **12**, 196.
20 P. M. Maffettone, P. Friederich, S. G. Baird, B. Blaiszik, K. A. Brown, S. I. Campbell, O. A. Cohen, R. L. Davis, I. T. Foster, N. Haghmoradi, *et al.*, *Digital Discovery*, 2023, **2**, 1644–1659.
21 R. Vescovi, T. Ginsburg, K. Hippe, D. Ozgulbas, C. Stone, A. Stroka, R. Butler, B. Blaiszik, T. Brettin, K. Chard, *et al.*, *Digital Discovery*, 2023, **2**, 1980–1998.
22 H. Fakhruldeen, G. Pizzuto, J. Glowacki and A. I. Cooper, *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6013–6019.
23 B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, *et al.*, *Nature*, 2020, **583**, 237–241.
24 Y. Jiang, H. Fakhruldeen, G. Pizzuto, L. Longley, A. He, T. Dai, R. Clowes, N. Rankin and A. I. Cooper, *Digital Discovery*, 2023, **2**, 1733–1744.
25 J. X.-Y. Lim, D. Leow, Q.-C. Pham and C.-H. Tan, *IEEE Trans. Autom. Sci. Eng.*, 2020, **18**, 2185–2190.
26 I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik and K. Sreenath, *Sci. Robot.*, 2024, **9**, eadi9579.
27 P. M. Maffettone, D. B. Allan, S. I. Campbell, M. R. Carbone, T. A. Caswell, B. L. DeCost, D. Gavrilov, M. D. Hanwell, H. Joress, J. Lynch, B. Ravel, S. B. Wilkins, J. Wlodek and D. Olds, *Workshop on AI for Accelerated Materials Design (AI4Mat) at NeurIPS*, 2022.
28 D. Allan, T. Caswell, S. Campbell and M. Rakitin, *Synchrotron Radiat. News*, 2019, **32**, 19–22.
29 P. M. Maffettone, C. Fernando, T. Caswell, D. Gavrilov, P. Shafer, D. Allan and S. I. Cambell, *Workshop on Accelerating Discovery in Natural Science Laboratories with AI and Robotics at ICRA*, 2024.
30 S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas and M. J. Marín-Jiménez, *Pattern Recogn.*, 2014, **47**, 2280–2292.
31 E. Olson, *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.
32 H. R. Kam, S.-H. Lee, T. Park and C.-H. Kim, *Telecommun. Syst.*, 2015, **60**, 337–345.
33 B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Muller, A. Evans, D. Fox, J. Kautz and S. Birchfield, *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 606–617.
34 A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár and R. Girshick, *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 3992–4003.