VisCPs: Visualizing the Execution of Command Patterns

Zainab Saeed Wattoo

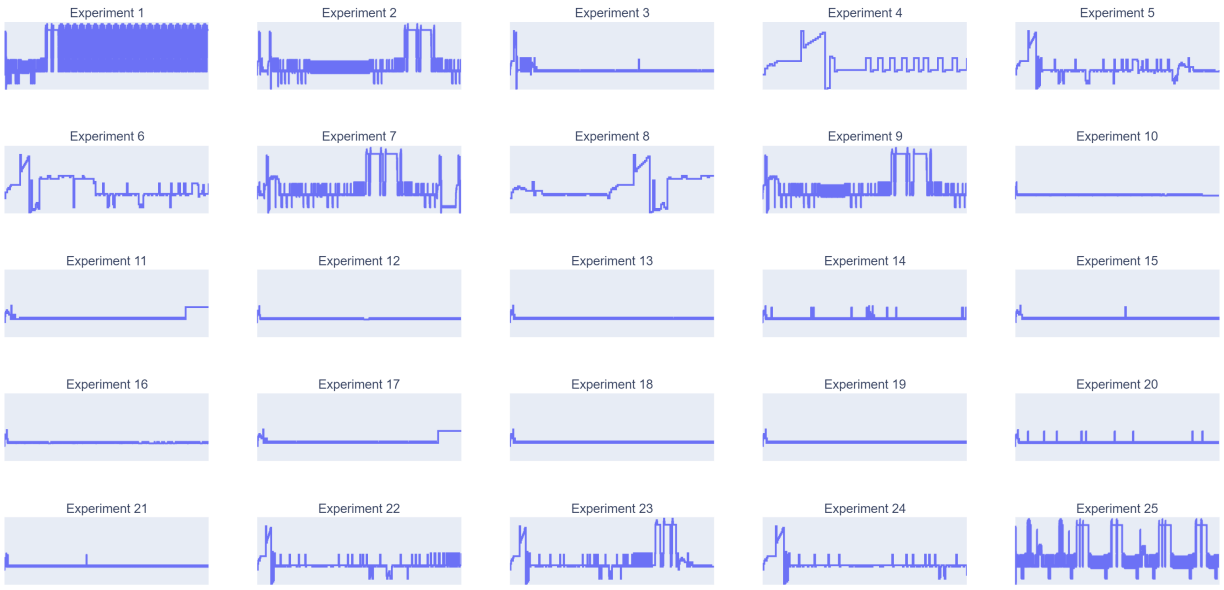


Fig. 1. VisCPs: displays the command patterns of different organic chemistry experiments carried out at a self-driving laboratory (Hein Lab at University of British Columbia).

Abstract— Self-driving laboratories are transforming the way experiments are performed in labs. They integrate automated robotic platforms, cyber-physical systems (CPS) and artificial intelligence (AI) to perform automated experiments, make autonomous discoveries and carry out mundane tasks performed by the researchers. However, these robotic platforms and CPS used as a setup in self-driving laboratories are vulnerable to attacks over the Internet. To protect this particular setup, we are working towards building an Intrusion Detection System (IDS) for a self-driving laboratory in chemical sciences domain - Hein Lab at University of British Columbia (UBC). Designing an effective IDS requires some understanding about the different experiments being performed and data analysis to understand the patterns of the commands sent by the cyber-physical systems used in the labs. To bridge this gap, we present a visualization web application - VisCPs that displays the execution of command patterns of the experiments performed at Hein Lab. This application helps in comparing different experiments and visualizing the signatures of each experiment being performed in the lab. To evaluate our web application, we took a survey from three users from diverse backgrounds who ran our web application. Our results show that the web application helped them in finding a pattern in the performed experiments. Moreover, the responses demonstrate that the application is easy to use and interactive with a limitation of a feature related to usability of the web application.

Index Terms—Self-driving laboratories, cyber-physical systems, intrusion detection system, visualizing command patterns.

1 Introduction

Self-driving laboratories [2, 3] are emerging over time that use robotic platforms, computer vision and artificial intelligence to perform automated operations and work towards autonomous discoveries. Moreover, they are used to automate the routine tasks and automate the tasks where manual labor is involved. We specifically collaborate with a self-driving laboratory - Hein Lab [2] at University of British Columbia (UBC) for this project. Figure 2 shows automated solubility experiment setup at Hein Lab where a lab computer running python scripts sends commands to the Robot Arm (N9) and other cyber-physical system (CPS) modules such as Arduino Augmented Quantos (Solid Dosing), Tecan Cavro (Liquid Dosing) and IKA Magnetic Stirrer (Stirring and Heating) connected to it to carry out different automated experiments.

Unfortunately, the automated robotic platforms and the CPS setup in self-driving laboratories are at risk from an intruder or a hacker that could intercept the network communication. For example, in the Hein Lab CPS setup the hacker who has gained control over the lab computer can manipulate the python scripts and send malicious commands that change the behavior of the robot arm or the CPS modules; leaving them in an unwanted state or manipulating them to perform dangerous experiments. To defend these laboratories from various attacks, we are in the progress of developing an Intrusion Detection Systems (IDS) specifically for Hein Lab so that safe experiments are performed correctly and accurately. As the initial stage towards building an IDS, we have deployed a middlebox that sits between the lab computer and the robot arm along with the CPS modules. This middlebox is a computer that does not need to be connected to the internet connection, shown in Figure 3 is connected to the lab computer via Ethernet and connected to the robot arm along with the other CPS modules using different communication channels. The middlebox collects the traces of the experiments that are being done in the lab and also forwards the commands to the robot arm and the other CPS modules.

To design an effectual IDS on the middlebox, we need to analyze the

• *Zainab Saeed Wattoo is M.Sc. Computer Science student at University of British Columbia. E-mail: zswattoo@cs.ubc.ca.*

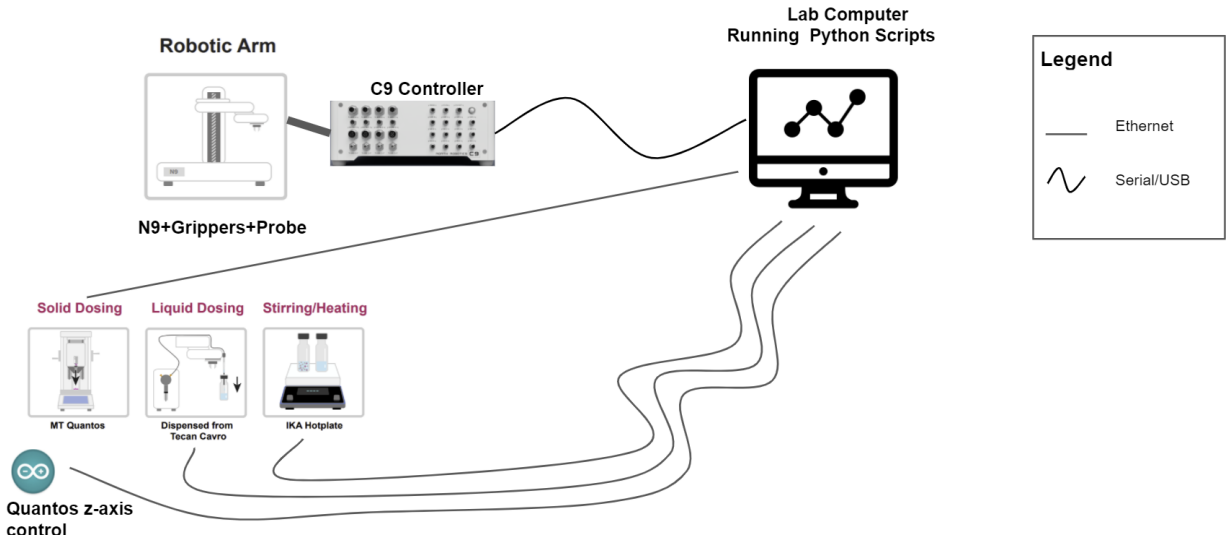


Fig. 2. Hein lab setup for Automated Solubility experiment. A lab computer running python scripts is sending commands to the robot arm and its assisting CPS modules

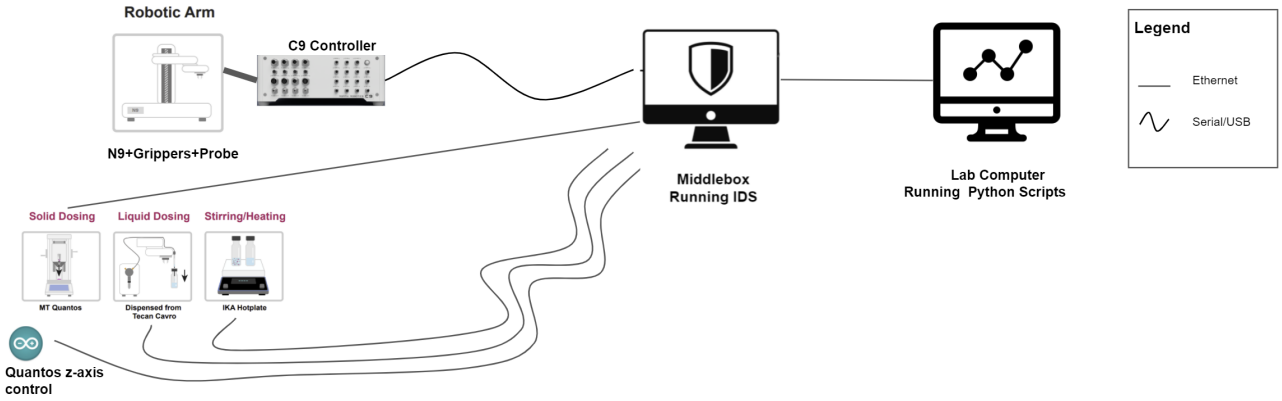


Fig. 3. Middlebox will be running Anomaly Based Intrusion Detection System which is connected to the robot arm and its assisting CPS modules. Further, it is connected to the lab computer via Ethernet.

command data sent from the lab computer to draw comparisons between the different automated chemical experiments performed at Hein Lab. Further, at a lower granularity, we need to recognize the different command patterns of the commands that are sent via different cyber-physical systems (CPS) deployed in the lab while performing a specific experiment. To address this challenge of data analysis, this paper presents VisCPs, a web application that displays the execution of the command patterns over time for different automated experiments. The main goal of this application is to help in understanding the similarities and differences between different experiments. To achieve this goal, Figure 1 displays the overall view where all traces in the form of a step graph of the experiments are shown together. Another goal of this application is to present the signature of each experiment to give an understanding of how the command patterns change over time for a particular experiment. Towards this end, the user selects a specific experiment that they want to visualize and the VisCPs plots the step graph along with the information regarding that experiment. Further, the user can also zoom, pan, auto-scale and download the plot.

For our evaluation, we asked three users from different education backgrounds to run our application and perform different tasks. Later, they filled a survey that has some questions regarding the interpretability, usability and interactivity of the application. All three of them were able to recognize a pattern from the overall view of the experiments. In addition, most of them found it easy to use and an interactive application. However, all of them found a feature related to usability of the web application of going back to the main page as a challenge.

The remainder of the paper is organized as follows. Section 2 discusses the related work in the domain of visualization that was helpful in building an IDS and visualizing patterns. Section 3 describes the data and task abstractions. Section 4 presents our solution - VisCPs followed by the implementation details of VisCPs in Section 5. Section 6 presents the results of the survey conducted to evaluate the application. Section 7 and 8 concludes our work with describing the limitations, future work and a conclusion.

2 Related Work

We review the work where visualizations have helped to build an Intrusion Detection System and Intrusion Detection System based Tools. Further, we explore the work that has been done to visualize different patterns in different areas of research.

2.1 Intrusion Detection Systems and Tools

Several works [5, 6] have been done to create Intrusion Detection System with Visualizing Capabilities. Luo et al. [6] proposed a four-angle-star based visualized feature generation approach (FASVFG) to evaluate the distance between samples in a 5-class classification problem that is used for building an IDS. The four-angle-star approach is used to classify an unknown point to a class where the four vertexes

Field Name	Description	Data Type	Cardinality
Timestamp	Time of the command	Ordered	12th October 2021 - 22nd October 2021
Module	Name of the Module	Categorical	5
Command Name	Name of the command	Categorical	45
Arguments	Arguments executed with the command	Categorical	5747
Responses	Responses received once the command executed	Categorical	0
Exceptions	Exceptions got once the command executed	Categorical	0

Table 1. Dataset Abstraction

represent four anomaly classes and from which new features are generated. It is a feature reduction approach that is used before the data is fed into the classification model. Moreover, Karami [5] worked on a novel anomaly-based IDS where visualization capabilities uses modified Self-Organizing Map (SOM) in the presence of benign outliers. Self-Organizing Map - an unsupervised learning algorithm is used to convert high dimensional data to low-dimensional data that is similar to a 2D discrete map with nodes and connections. It detects the attack and anomalies correctly and also provides useful insights to the end users.

LogicMonitor [1] is an Intrusion Detection Based tool that uses advanced machine learning algorithms to visualize the expected data patterns for data points, so one could see the data that falls outside of these patterns. The LogicMonitor provides dashboards and widgets to display the data the way users wants in order to catch the issues before hand in order to avoid further more severe events.

Their approaches [1,5,6] visualized and analyzed the network traffic data and helped with feature reduction of the dataset or building an IDS using visualization capabilities. However, our data consists of the command traces being used for carrying out the chemical experiments. The main goal is to visualize the command patterns for data analysis which is the stage before building an IDS.

2.2 Visualizing Patterns

People have worked on visualizing different patterns in different research areas to help them get an understanding of the data. Towards this end, BubbleNet [7] is a first complete visualization design study for cyber security where a dashboard is created to help the network analysts understand the patterns within the data. This work is used as a design study and uses network security dataset. Whereas, our work uses the commands data of the cyber-physical systems not the network packets that are used for the communication.

Further, Singh [8] analyzes and visualizes system calls that captures useful information of system call logs to detect malicious activity from benign activity. They find patterns by visualizing their data via scatter plot and line chart. Cadez et al. [4] proposes a new methodology for observing the patterns of navigating through Web Sites. They divide the users with similar navigation paths into one cluster. They create a tool called WebCANVAS (Web Clustering Analysis and VisuAlization of Sequences) for visualizing clusters of users with same behavioral pattern.

Both works [4, 8] are used to visualize the patterns in their own domain such as observing system calls or user behavior on Web Site. However, our work is concerned with visualizing the commands patterns in the cyber-physical systems security domain.

3 Data and Task Abstraction

The following section describes the data and task abstraction.

3.1 Data Description

In the Hein Lab, the traces are collected by our middlebox for every experiment that runs on the lab computer. These traces are collected from five different CPS modules: (i) C9 Controller controls the Robot Arm N9 and a Centrifuge. (ii) Robot Arm UR3e. (iii) Arduino Augmented Quantos is used for dosing the solid in vials. (iv) Tecan Cavro is a pump that is used for dosing the liquid in vials. (v) Magnetic Stirrer is used for stirring and heating the solution in vials. Once these traces are collected, they are processed and stored in the form of a json file, csv file and further stored in MongoDB. For building our solution, we use only the supervised experiment csv files that are in table format to visualize the data. A supervised experiment is the one where the name of the experiment was known when the trace was collected for that particular experiment.

The current dataset contains normal and anomalous traces with a size of 2.32 MB with 25 csv files. Within these 25 csv files, there are 6 attributes (fields) and 30126 items (commands) in total that have been executed. The data description is shown in Table 2. These 25 csv files were supervised with a total of four experiments : (i) Automated Solubility with N9. (ii) Automated Solubility with N9 and UR3e. (iii) Crystal Solubility. (iv) Joystick Movements. The number of csv files for each procedure runs is shown in Table 3. From these csv files, we label three of them as anomalous as they resulted in a crash of a robot arm with another device. The rest of the 22 csv files were labeled as benign as the experiment ran successfully or it was stopped midway by a lab researcher for example if they did not use the correct gripper for the robot arm to pick up a vial.

Data Type	Table
Data Size	2.32 MB
Total Number of CSV files	25
Total Number of Fields	6
Total Number of Commands	30126

Table 2. Dataset description

Procedure Name	No. of csv files
Automated Solubility with N9	5
Automated Solubility with N9 and UR3e	4
Crystal Solubility	4
Joystick Movements	12

Table 3. Number of csv files per experiment

Each csv file known as a trace for an experiment contains the timestamp, commands and arguments sent to the robot arms and the assisting CPS modules and contains the responses and exceptions that are received while the experiment is running. Table 1 shows the data abstraction of the tracing dataset containing the description, data type and the cardinality. There are 6 attributes: Timestamp, Module, Command Name, Arguments, Responses and Exceptions from which we will use the first four attributes (Timestamp, Module, Command Name, Arguments) for the filtered dataset. The ordered Timestamp contains the timestamp when the command was executed, the categorical Module contains the name of the module, the categorical Command Name contains the name of the command that was executed, the categorical Arguments, categorical Responses and categorical Exceptions field contains the arguments, responses, exceptions of the command that was executed.

3.2 Task Description

For designing an effective IDS, the command data produced by the middlebox traces needs to be used for understanding the similarities and differences between different automated experiments. From these experiments, a specific experiment needs to be selected to understand its signature in detail along with the description containing the name of

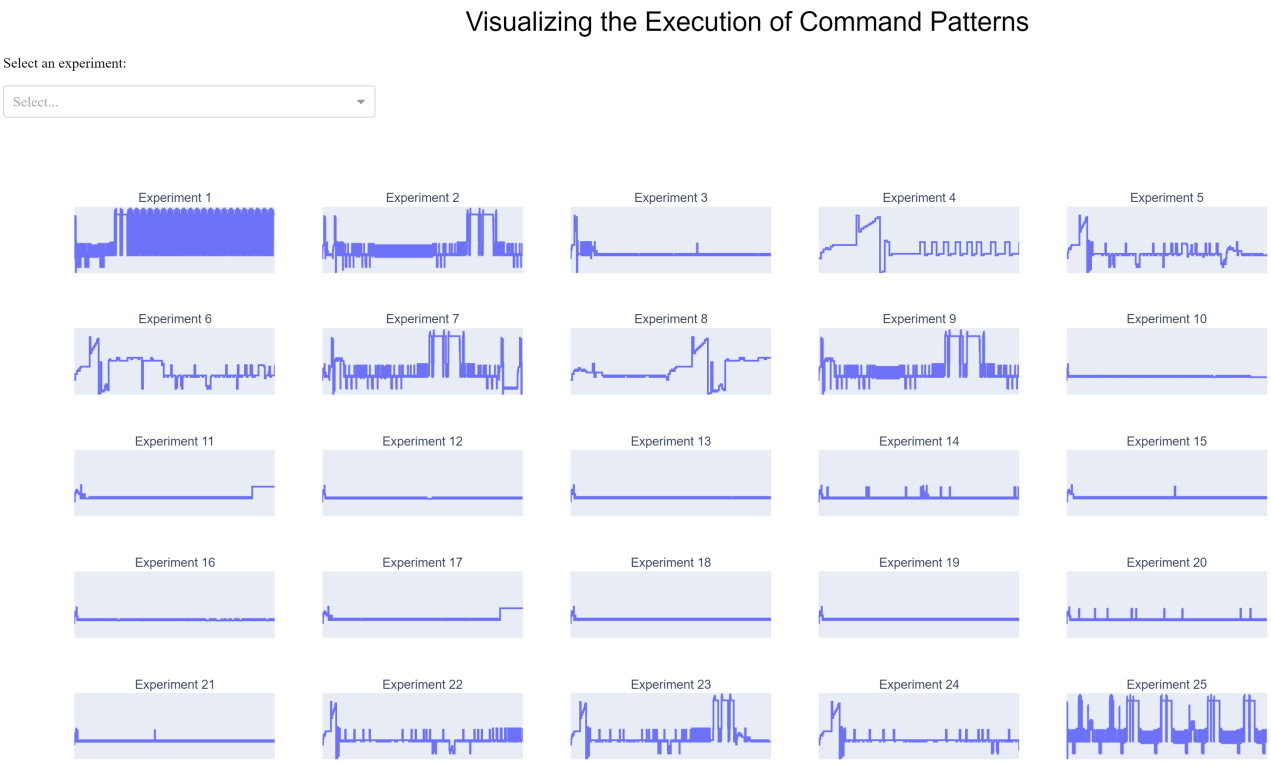


Fig. 4. Main page of VisCPs displays all the subplots of the experiments and also includes a drop down box at the top of the page to select a particular experiment.

the experiment, the type of experiment (anomaly/benign) and the description on how that particular experiment was performed. Further, the command patterns needs to be also presented for a particular timestamp and arguments for a particular command as well.

In the abstract language, we are creating a web application that helps to analyze and search within the tables presented by the csv files. For analyzing, we want to consume the data to present the overall view of all the different experiments showing the similarities and differences within them and also present the trace of each experiment along with its information using subplots and plots respectively. The web application uses two types of search : one where location is known - "Lookup" and one where the location is unknown - "Explore". The user can lookup from the drop down menu, the specific experiment that they want to visualize. Further, using the hover, zoom, pan and auto-scale they can explore the plots for a particular timestamp that they want to visualize the commands pattern for and also see the arguments of a particular command.

4 VisCPs

VisCPs, our proposed web application, presents the execution of the command patterns over time. According to the user tasks and requirements, our application caters to the following : (1) Provides a holistic overall view where the command patterns of different experiments are at the main page. (2) Presents the trace of a specific experiment that is selected by the user from the drop down menu on a different page. In the following sections we describe the idioms used and the design choices considered to create the web application. Moreover, we present the what-why-how framework and a use case scenario of the application.

4.1 Experiments Overall View

The main page presents time-varying step graphs as subplots of all experiments using the command names from the dataset. On these step graphs, the x-axis is the time step and the y-axis is the command name. We decided to use step graph as the idiom for this particular dataset because the command names are categorical and they help in demonstrating the length of time it takes for each command to be executed. Other graphs such as scatter plots are unsuitable as they do not accurately display the amount of time it takes for a command to be executed. Further, a line graph could be another alternative but it adds noise to the data as the attribute chosen for the y-axis is categorical not quantitative, making it also unsuitable for this scenario.

There are 45 total commands on y-axis that are consistent for all the subplots, helping in understanding the different signatures of different experiments. These commands are ordered according to their order in the experiments that are conducted. For example, the solid dosing in vials using Arduino Augmented Quantos (Quantos) is done before the liquid dosing in vials using Tecan Cavro (Tecan), placing the Quantos commands at the top compared to the Tecan commands at the bottom on the y-axis. The robot arms (C9 and UR3e) commands are placed in the middle of the y-axis command order as they are either the most frequent commands or can be used in any order at any time during the experiment.

The overall view including the drop down to select a particular experiment on the main page is shown in Figure 4. We used lines as the mark of the step graph to encode the command used at a particular time. One of the channel used was the length of the lines of the step graph that were used to encode the amount of time it takes for a particular command to be executed. Another channel was the same color used throughout the experiment encodes the whole command pattern of the experiment.

4.2 Individual Signature View

The individual signature view is displayed on another page when the user selects a specific experiment from the drop down menu as shown in Figure 6. This drop down menu is used to reduce the dataset and uses only the dataset for the one required for the experiment selected. This individual view displays a step graph that is the zoomed in version

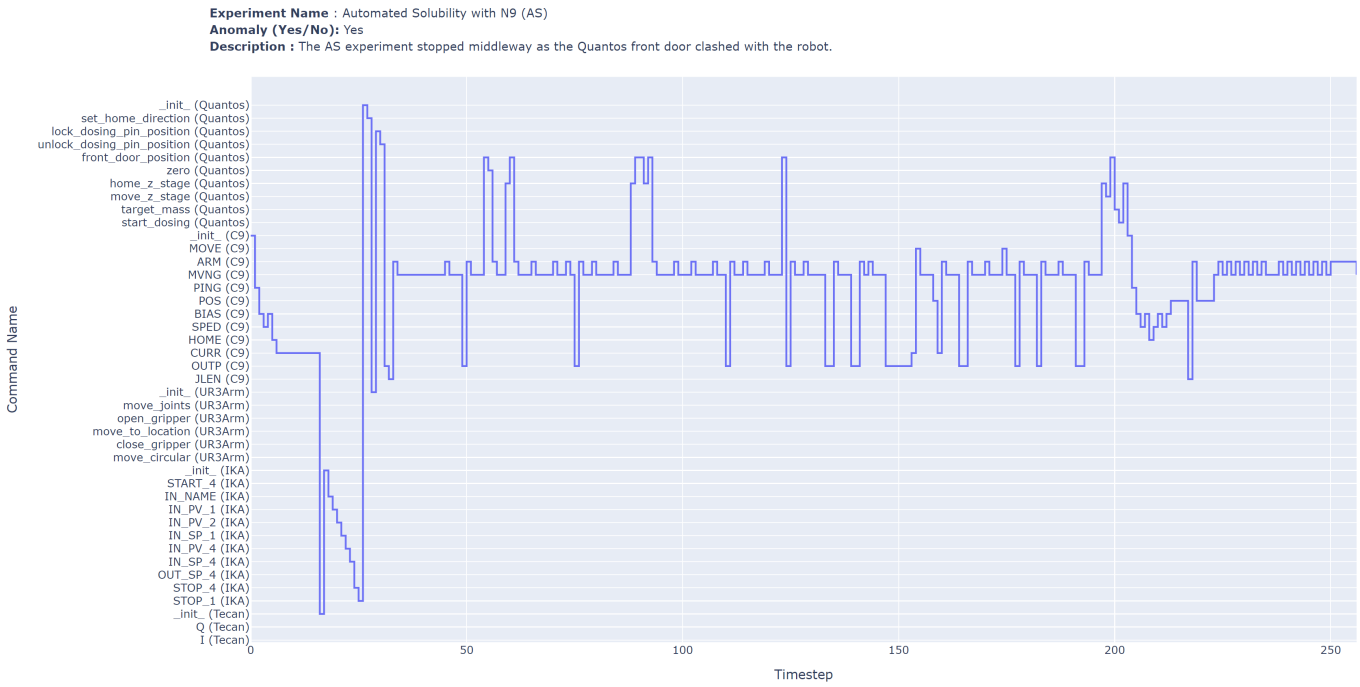


Fig. 5. Individual signature of Experiment 5.

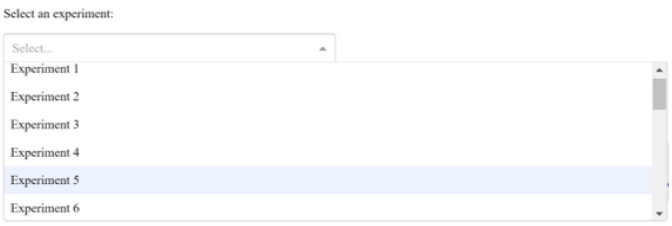


Fig. 6. Drop down menu in VisCPs to select the experiment.

Fig. 7. Toolkit shows up at the top of the graph that helps to zoom, pan, auto-scale and download.

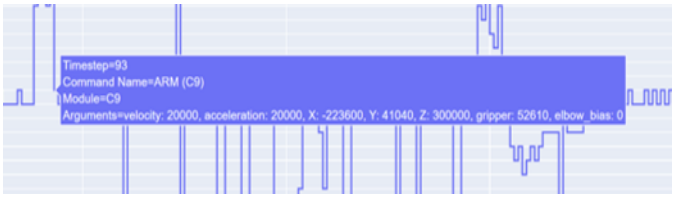


Fig. 8. Pop up appears when one hovers over a particular command at a particular time step.

Fig. 9. Cross button inside the experiment drop down menu switches to the overall view.

of the subplot on the main page. This step graph also has time step on the x-axis and command names on the y-axis.

Data pre-processing allowed to filter out the commands that do not appear in the experiment and displays only the commands that do appear on the y-axis. Further, the y-axis command name labels are processed in a way that it displays command names along with its module name as well.

Similar to the subplots, the line is used as a mark to encode the commands at a specific time and the length of the line is used as a channel to encode the amount of time it takes for individual command to be successfully completed. All commands used the same color channel to encode the command pattern of an experiment. Figure 5 shows the individual view of Experiment 5. When the user hovers over the lines of the step graph that represent the command, we can see the time step, the command name, the module name and the arguments associated with that command name as shown in Figure 8.

Further, the step graph of the experiment has a title that displays the name of the experiment, the type of signature (benign/ anomaly) and the description of the experiment that the signature is for. The options on the top right hand side of the graph displays a toolkit along with the pop up displaying its purpose. This toolkit allows multiple functionalities such as it allows the user to zoom in to the step graph, zoom out of the step graph, zoom at a particular time step, pan the trace so that the user to drag along the trace to observe the zoomed in version for all time steps, auto-scale to original view of the graph and download the plot. This toolkit is shown in Figure 7. To go back to the main page that displays the overall view, the user needs to select the cross button inside the drop down menu box as shown in Figure 9.

4.3 What-Why-How Framework

Table 4 shows the what-why-how framework that is used to analyze the web application.

What: Data	Table data; items and attributes; ordinal and categorical
Why: Tasks	Understanding comparisons between different experiments; Understanding individual signature of a particular experiment
How: Encode	Step graph; Line encodes the command at a particular time; Length of the line encodes the time it took for a particular command to be executed
How: Manipulate	Zoom; Pan; Auto-scale
How: Reduce	Filter according to a specific experiment

Table 4. what-Why-How Framework

4.4 Use Case Scenario

The researcher at the Systems and Security Lab at University of British Columbia runs the web application in their web browser. The web application displays the overall view on the main page where all the experiments are displayed. The researcher is able to determine a specific pattern such as from experiment 10 to experiment 21 have a similar pattern indicating that it was a joystick movement experiment. Further, they want to visualize experiment 3 as the pattern for most of the time steps looks similar to joystick movement experiments. From the drop down menu, they select Experiment 3 which takes them to another page displaying the zoomed in version of Experiment 3. They get the information and the description related to the experiment. Later, they observe that the time step from 1 to 100 looks different from the rest of the command pattern. To observe it closely, they select the zoom button from the toolkit and drag from time step 1 to 100 which displays the zoomed in version for that particular time step. To view the rest of the trace, they select the pan button from the tool kit to drag along the rest of the trace. In order to move back to the original view of the trace, they select the auto-scale button from the toolkit. Moreover, they hover over the commands to get an idea of the arguments used with the commands. The researcher also wants to download the plot to better understand it later offline. For that purpose, they select the camera button from the tool kit to download the plot. Lastly, in order to go back to the main page, they select the cross-button from the drop down menu box that takes them back to the overall view.

5 Implementation

In this section, we will discuss the implementation details used for reading the data, processing it and for visualizing it using the web application. Later, it lists down the time it took to implement different tasks.

5.1 Data Reading and Processing

We used Python as the programming language that was used to read the supervised csv files. Python was also used for filtering the commands that are not part of a particular experiment for the individual signature view. Further, it was also used to change the y-axis labels where the module name was embedded with the command names.

5.2 Visualizing

Our solution was implemented in Python and uses Plotly Express and Dash libraries. The plotly library was used to create the step graphs for subplots and the plots for individual experiments. It comes with the zoom, pan, auto-scale and download feature. Plotly also allows to integrate the hover feature to display the modules and arguments of the commands. The dash library is used to create the web application that integrates the html components. It gives the functionality to add the drop-down menu along with moving from one page to another making it interactive. Before choosing these libraries, we worked with matplotlib and plotly go library that had their own limitations and flask library that did not seem compatible with some functionalities of plotly express.

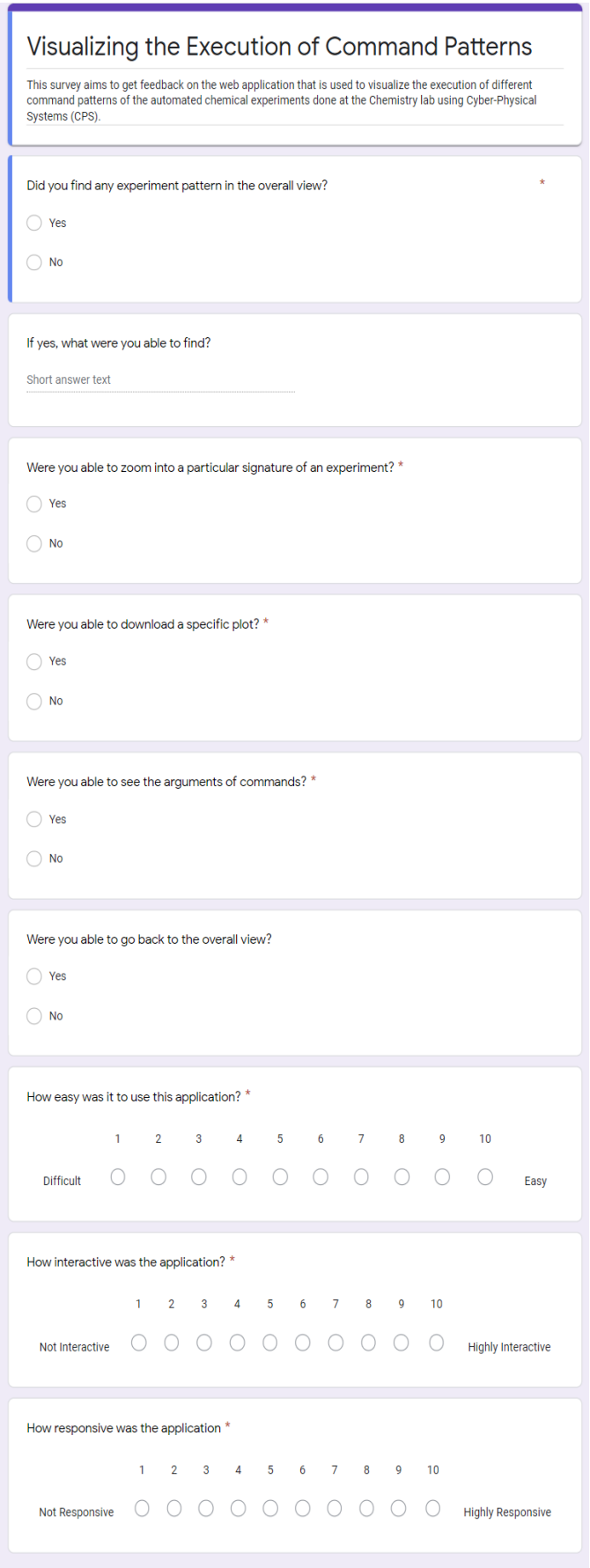
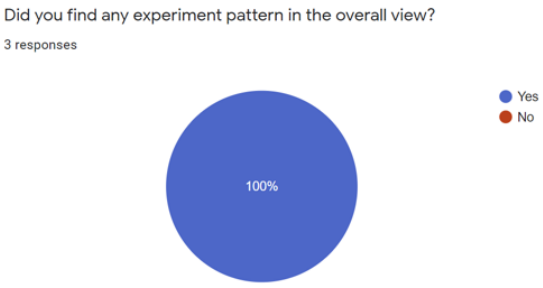
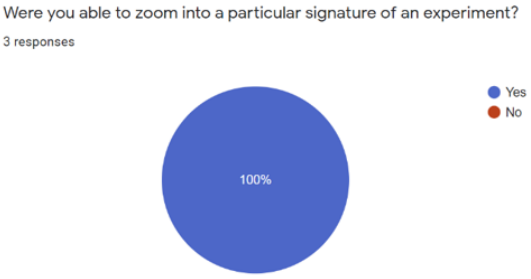


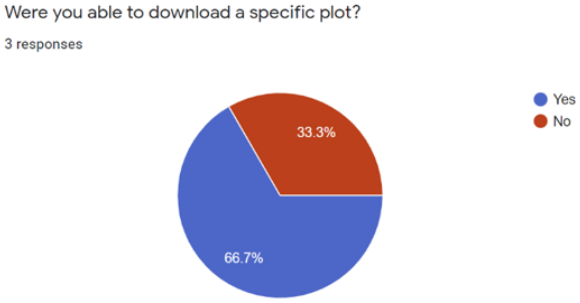
Fig. 10. Survey used for evaluating the web application.



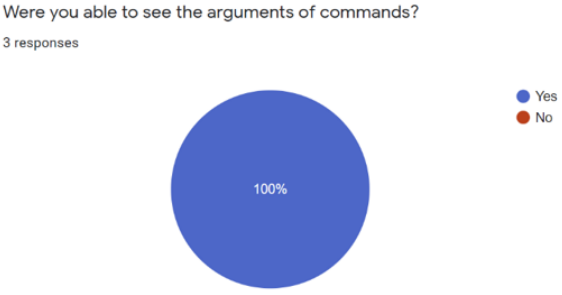
(a) Experiment Pattern



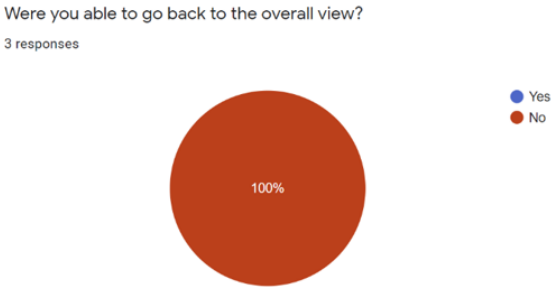
(b) Zoom Feature



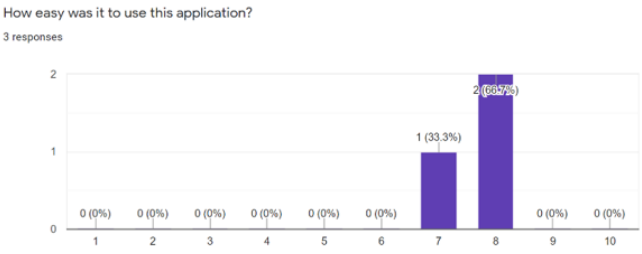
(c) Download Feature



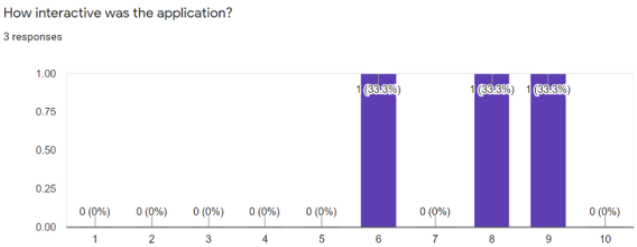
(d) Hover Feature



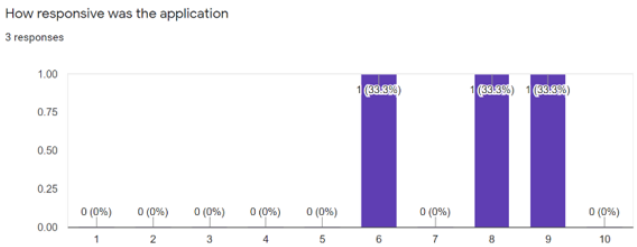
(e) Going Back to Main Page



(f) Easy to Use



(g) Interactive to Use



(h) Responsiveness

Fig. 11. Results of survey filled by three users. **(a)** The pie chart shows 100% response of the users finding experiment pattern. **(b)** The pie chart shows 100% response of the users finding the zoom feature in the web application. **(c)** The pie chart shows 66.7% response of the users finding the download feature in the web application. **(d)** The pie chart shows 100% response of the users finding the hover feature in the web application. **(e)** The pie chart shows 100% response of the users not finding the feature of going back to the main page feature. **(f)** The bar chart shows 66.7% response of the users found it easy to use. **(g)** The bar chart shows a positive response of above 5 for finding the web application interactive to use. **(h)** The bar chart shows a positive response of above 5 for finding the web application responsive.

5.3 Milestones

This section lists down the tasks that were performed along with the number of hours it took to achieve those tasks.

Tasks	No. of Hours
Pitch	3
Project Proposal	10
Data Collection and Finalization	12
Deciding on visualization tool	3
Project Update	4
Initial Implementation and learning tools	8
Implementing overall view	8
Implementing individual view	16
Finalizing Code and Artifact	3
Teaser Image	1
Final Report	20
Total No. of Hours	80

Table 5. Milestones

6 Results

For our evaluation, we created a survey that lists some questions regarding the interpretability, interactivity and usability as shown in Figure 10. Some of the questions included : if the users were able to find any specific pattern in the overall view on the main page, how interactive did they find the application on a scale from 0 to 10 or how easy to use did they find the application on a scale from 0 to 10. Three users were asked to run VisCPs, our web application and fill the survey after performing tasks on the application. The users were from diverse education backgrounds : (1) Sauder Business School at UBC. (2) Electrical and Computer Engineering Department at UBC. (3) Computer Science Department at UBC.

The results are illustrated in Figure 11. Overall, the response was positive and encouraging. All of them were able to find a specific pattern as shown in Figure 11(a). Most of them were able to find the features over the web application as shown in Figure 11(b), Figure 11(c), Figure 11(d). The users mostly found the application easy to use, interactive and responsive as shown in Figure 11(f), Figure 11(g) and Figure (h). However, there was one limitation that the users were not able to go back to the main page and found it a challenge to spot the cross button inside the drop down menu as shown in Figure 11(e).

7 Discussion and Future Work

VisCPs is a problem-driven approach where it not only integrates the feature requirements but also tries to solve the underlying problem as well. It is indeed successful in the problem it is trying to solve. However, the web application is still incomplete and has some limitations that needs to be considered as future work.

Considering the design choice, it will be easier for the user to select the subplot directly in order to choose the experiment for which the command pattern they want to visualize, rather than selecting from a drop down menu. Further, when the individual signature view shows up for different experiments, the step graph dimensions are not consistent for all the step graphs. Moreover, the users found it a challenge to go back to the main page. Therefore, another design choice would have been better that needs to be considered as future work where we can add a home button and a help button to help them navigate to the overall view. Currently, the design choice to display the overall view is perfect for limited dataset. However, as the number of experiments being done in Hein Lab increases, the dataset will grow over time and a different design choice would need to be considered in a way that the users are able to find comparisons between the experiments.

Further, as a future work we want to integrate the real-time execution of the command patterns as the live experiment is performed in Hein Lab. In addition, to expand our work we will generate sub-sequences from data mining techniques. These sub-signatures are selected on the basis of the most frequent commands that appear together in all of the dataset. The sub-signatures can be encoded in the step graph via different colors to represent the sub-sequences. Lastly, as another future work we want to use the anomalous data points to be represented on our design solution to get an idea of the anomaly data points and raise a red flag at run-time when the experiment is being performed in Hein Lab.

8 Conclusion

With the advent of self-driving laboratories and use of cyber-physical systems (CPS) in these labs have raised security concern. To protect these systems, Intrusion Detection Systems (IDS) needs to in place which requires some understanding of the data patterns. We specifically collaborate with a self-driving laboratory in chemical sciences domain that carries out automated chemical experiments. In this paper, we present VisCPs, a web application that visualizes the execution of command patterns where the commands are sent via the CPS used to carry out automated experiments. This application is helpful in providing the comparisons between these experiments in an overall view displaying the subplots. Further, if the user wishes to understand the pattern of a particular experiment they can select the experiment from the drop down menu leading them to another page. This page displays the individual signature of the selected experiment along with its details as a step graph helping the user to understand how the command pattern varies over time for a specific experiment. The user is able to zoom, pan, auto-scale and download the step graph to analyze it offline. For our evaluation, we asked three users to run our application and fill a survey containing questions related to the design and interactivity of the application. The web application was helpful in determining the command patterns, achieving its goal and was considered easy to use and interactive. As the dataset increases, choosing a different design choice is still a question that is left for the future work.

Acknowledgments

The authors wish to thank Dr. Tamara Munzner for her guidance and support. This project was sponsored by the Hein Lab at University of British Columbia that is the self-driving chemistry lab at University of British Columbia and Systopia Lab (Systems and Security Lab) at University of British Columbia that is working on designing the Intrusion Detection System.

References

[1] Anomaly Detection Visualization. `https://www.logicmonitor.com/support/forecasting/anomaly-detection/anomaly-detection-visualization`.

[2] Hein Lab at University of British Columbia. `https://groups2.chem.ubc.ca/jheints1/`.

[3] Matter Lab at University of Toronto. https://www.matter.toronto.edu/basic-content-page/ai-for-discovery-and-self-driving-labs.

[4] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of Navigation Patterns on a Web site Using Model-Based Clustering. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 280–284, 2000.

[5] A. Karami. An Anomaly-Based Intrusion Detection System in Presence of Benign Outliers with Visualization Capabilities. *Expert Systems with Applications*, 108:36–60, 2018.

[6] B. Luo and J. Xia. A Novel Intrusion Detection System Based on Feature Generation with Visualization Strategy. *Expert Systems with Applications*, 41(9):4139–4147, 2014.

[7] S. McKenna, D. Staheli, C. Fulcher, and M. Meyer. Bubblenet: A cyber security dashboard for visualizing patterns. In *Computer Graphics Forum*, vol. 35, pp. 281–290. Wiley Online Library, 2016.

[8] A. Singh. System Call Analysis and Visualization. 2019.