ROBOTICS DASHBOARD WEB APP FOR LABORATORY MONITORING

Amaan Khan[1], Rory Butler[2,3], Carla Mann[3], Arvind Ramanathan[3]
[1]University of Illinois Urbana-Champaign, Champaign, IL,
[2]University of Chicago, Chicago, IL,
[3]Data Science and Learning Division, Argonne National Laboratory, Lemont, IL

MOTIVATION

- Well-functioning self-driving labs require numerous independent robotic systems
- Difficult for lab operators to monitor, analyze and debug their data and functionality, such as video streams, ROS topics, etc.
- A dedicated web app that provides real-time monitoring of video streams, annotated camera views from neural network prediction, and log outputs from ROS topics will improve productivity, debugging, and ease of use.

TOOLS

- Python
- ROS (Robot Operating System)
- plotly.dash & Flask (Python packages for developing web applications)

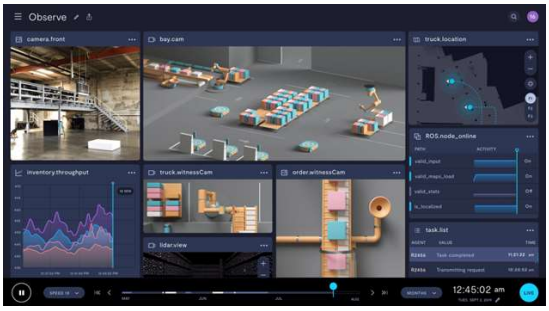

Image 1: Example of a dashboard app similar to what we aim to develop. Top-left box shows live-stream camera display, bottom-left box shows data visualization of the lab, and middle-right box shows statuses of ROS nodes. Source: https://formant.io

MAJOR ACCOMPLISHMENTS

- Developed software to live-stream camera from Intel NUC (computer) to a Flask app (residing in a separate computer)
  - Developed ROS publisher/subscriber model
  - Established socket connection between NUC and Flask App to send camera frames wirelessly over the network (*Fig. 2, top*)
- Developed two-way communication system between Flask App and NUC (*Fig. 2, Bottom*)
  - User can send messages from Flask app to NUC by clicking a button, typing a message, etc.
  - Data is sent through a socket connection for wireless communication
  - ROS publisher/subscriber model also developed
- Developed interactive graph display (*Fig. 1, right*) to visualize data

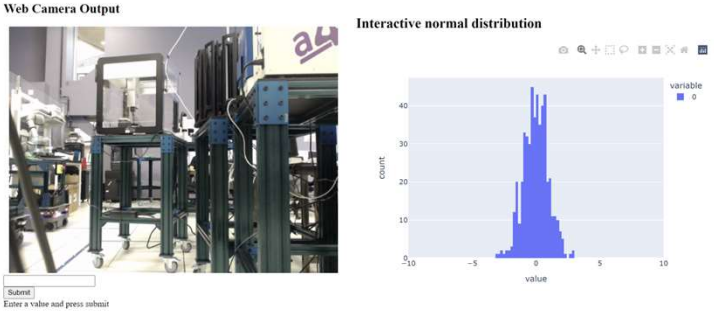

Fig. 1, Left: Live-stream camera output from Intel NUC displayed on web app
Right: Interactive graph to visualize data being sent from a robot to web app.

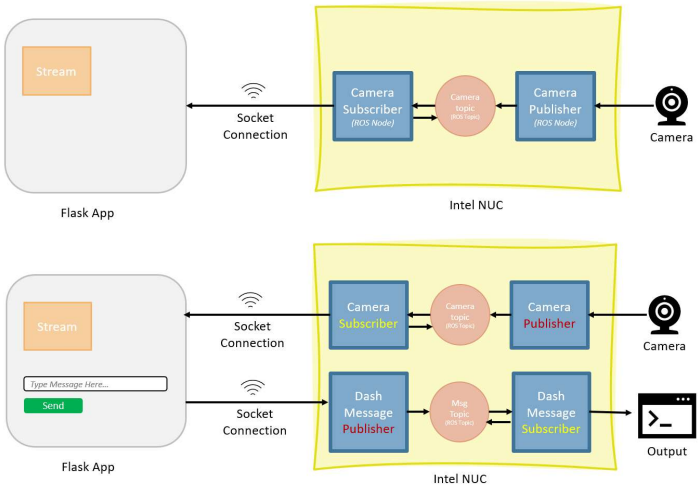

Fig. 2. Top: Demonstrates how camera connected to a computer, such as an Intel NUC, streams to the Flask App. Data is sent to a ROS publisher, which publishes data to a ROS topic. Then a subscriber listens to that topic and sends any data to the Flask app (residing on separate computer) over a socket connection.
Bottom: Demonstrates the two-way communication. User can type in a message in the Flask app and click send. Message is sent over socket connection, in which the "Message Publisher" publishes the message to a ROS topic and a subscriber fetches that message and outputs it.

IMPACT

- Dashboard enables flexible and easy access to debugging information from outside development environment and will increase productivity for robotic developers
- Set up initial camera communications infrastructure through ROS
- Can easily add functionality between blocks in the future, such as adding processing, analysis, etc. before displaying to user

FUTURE DIRECTIONS

- Remove socket connection layer and integrate Flask app with ROS
- Integrate feature to have multiple camera streams sent to and displayed in the Flask app
- Integrate window that displays log outputs from various user-chosen ROS topics
- Improve web application appearance using CSS

ACKNOWLEDGEMENTS

- This work was supported by the U.S. Department of Energy (DOE) Office of Science and DOE Office of Workforce Development for Teachers and Scientists (WDTS) through the Science Undergraduate Laboratory Internships (SULI) program. I also want to thank Rafael Vescovi for providing guidance and clarifications on ROS and the program architecture, as well as Casey Stone for setting up weekly tutorials and meetings.