The Frontier of Visualization for Large-Scale Robotics Datasets (Roboviz)

Given random data without context, looking at the raw data doesn't provide much insight about what you could learn from the data.

Our visualization tools reveal different properties, characteristics, and patterns, so that users can infer the quality of the dataset as well as what the data is even about in the first place. Our software currently only works with the franka robot model.

[Link to Live Demo]
[Link to Research Poster]

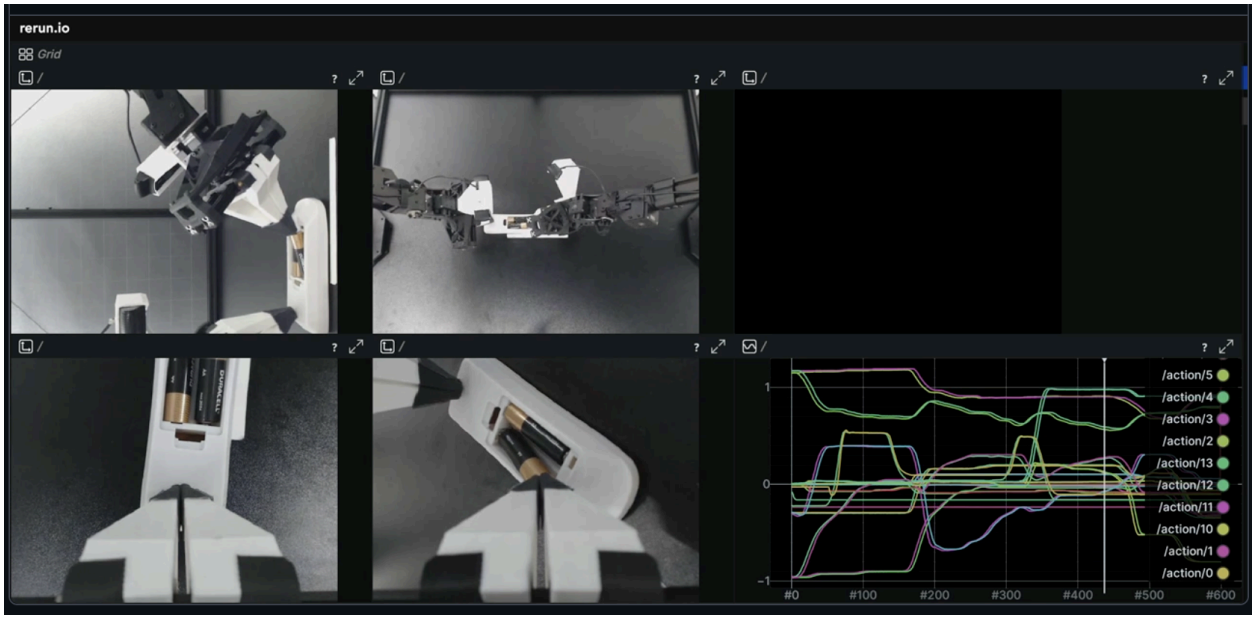
Description/short summary:
Our interactive visualization tool can process large-scale robotic datasets and reveal properties, characteristics and patterns that otherwise are difficult to determine by looking at the raw demonstration data. The tool currently supports LeRobot and HDF5 formats where it can classify play vs expert trajectories, cluster data to find behavioral hotspots, segment trajectories, remove partial demos, and compute KDE displayed on a 2D plane. Our goal is to assist researchers who work with complex robotics data and transform it into a more interpretable form.

Background/Motivation

Existing implementation only offers trajectory level data visualization. Some other work includes foxglove or [rerun.io]

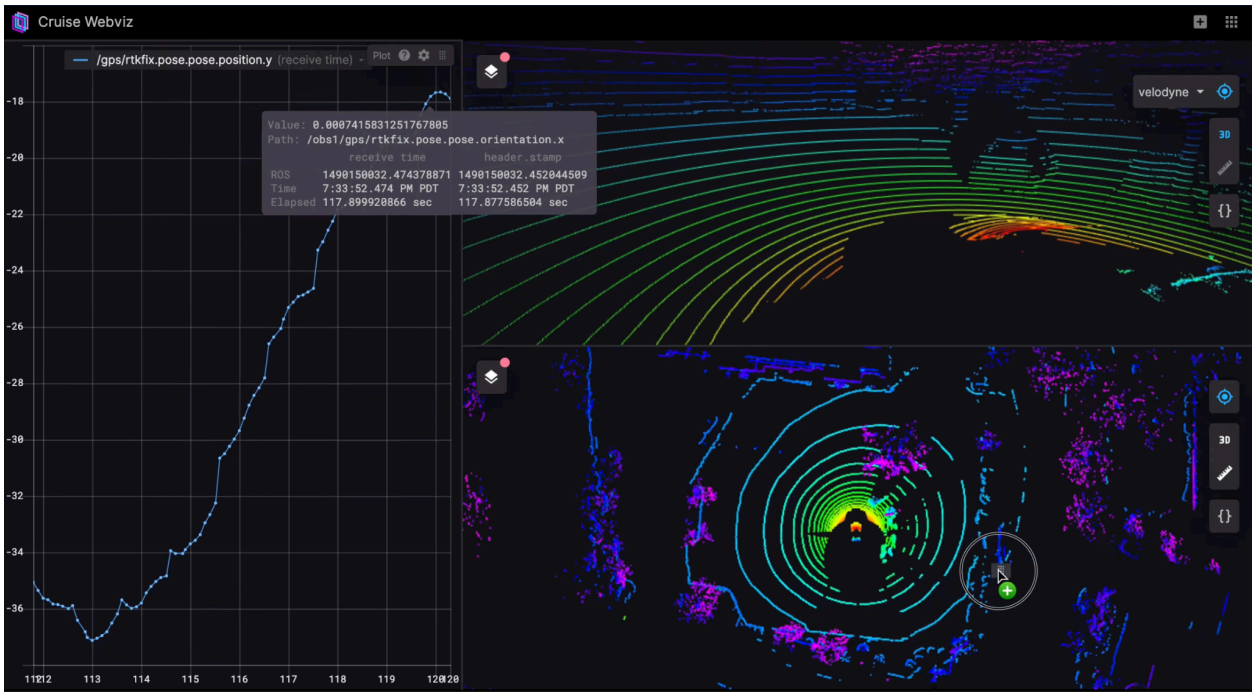
Some other robotic visualizations people have done:

[Online LerRobot Visualizer]



[LeRobot Command Line Visualization]

This only has visualization per episode, which only describes the actions of the robot.



https://webviz.io/

Again, only shows the trajectory data but not enough analytics on the dataset.

More work:
https://huggingface.co/spaces/lerobot/visualize_dataset
https://foxglove.dev/product/visualization
https://rerun.io/examples/robotics/droid_dataset

Rationale

Robot learning tasks largely depend on the quality of the dataset. It is hard to evaluate the quality of a dataset for robot learning tasks just by visualizing singular trajectories in isolation. We plan to expand on existing implementations by proposing novel analytical and visualization methods for robotics datasets.

Our approach breaks down the dataset into 3 levels.

- L0, which is the raw dataset
- L1, the dataset split on vastly different trajectories (usually task-specific expert data vs undirected play data)
    - One notion is to split on entropy
- L2, further breaking down expert trajectories into multiple sub-trajectory segments for more in-depth analysis
    - Clustering by meaningful edges or “primitive skills” like “reach toward object”, “grasp object”, “transport object”, “insert object”, “reset to home”, etc.

L0, L1, L2 can be seen on the example plots in our demo.

Methods/Our implementation

Data format

HDF5
LeRobot

Types of tasks

- Separating play and expert data in a combined dataset
- Identifying partial and full trajectories
- Visualizing high-density regions in the trajectory
- Segmenting a trajectory into multiple parts at high high-density region

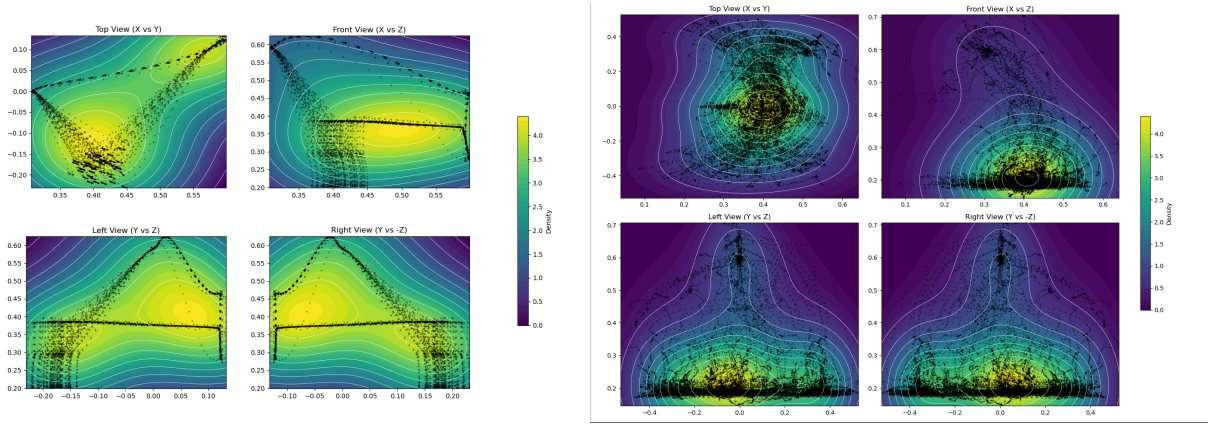
Scripts

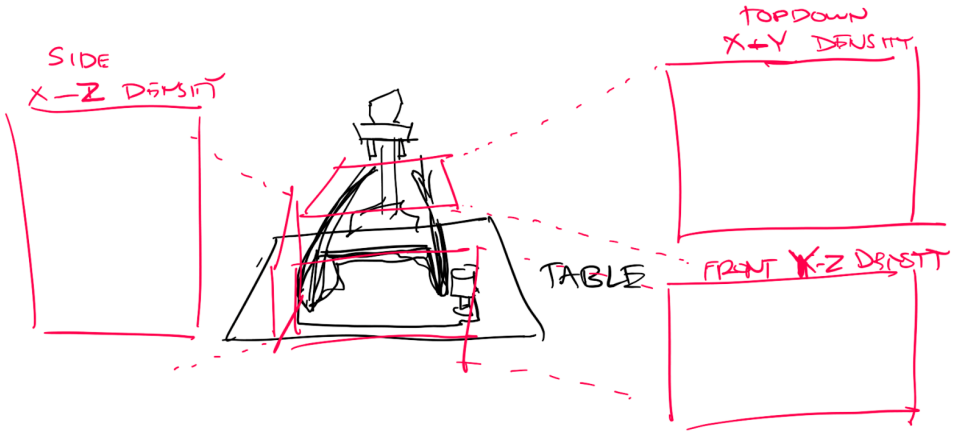
Show the implementations as a flow chart of different before and after plots.

- **Expert and play trajectory classifier**

- We take advantage of the inter-trajectory state entropies to separate play and expert data at the L0 level to produce L1 level dataset. Expert data generally has a lower inter-trajectory state entropy
- Initial evaluation of the dataset can also be made by evaluating the state entropy across the expert trajectories at the L1 level.

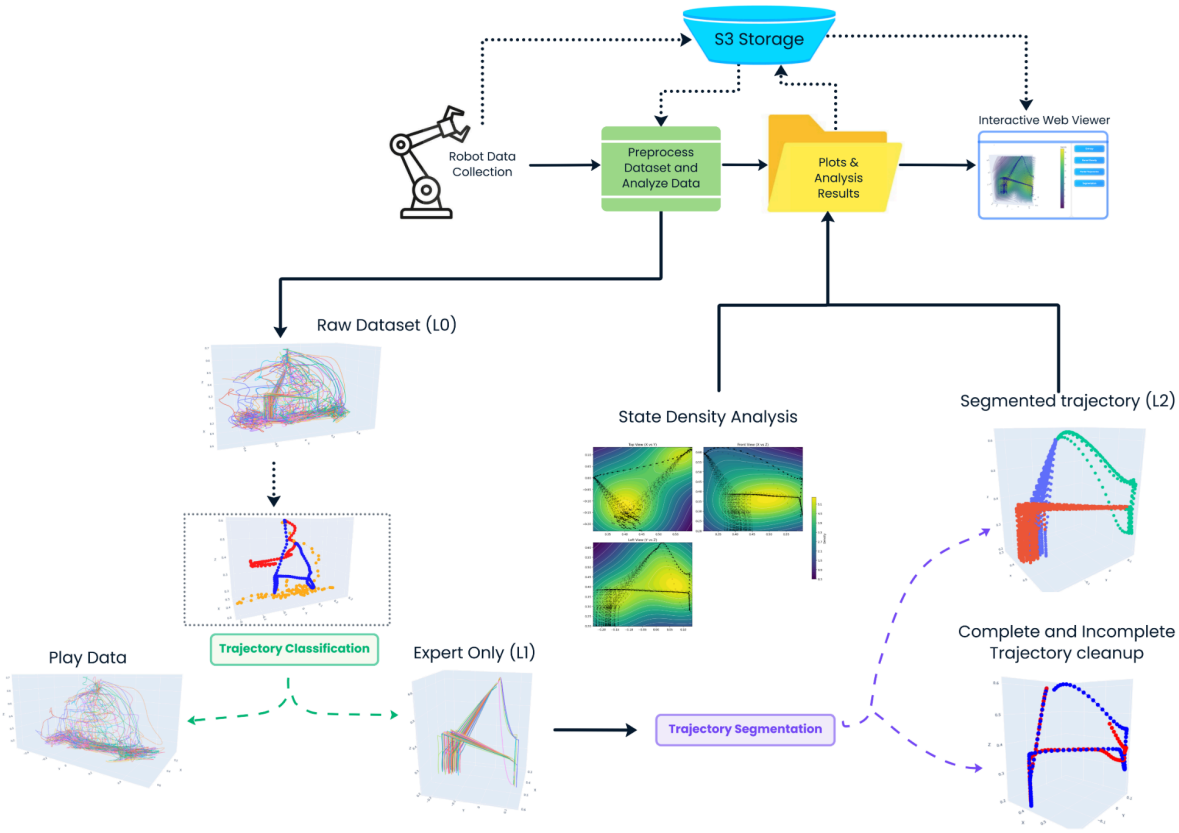
- **Kernel Density**
  - Evaluate the density of the dataset at the L1 level.
  - Visualize the highly dense regions in the dataset based on all the trajectories in it.





- **Full Trajectory Identification**
  - Extracts features for each trajectory, which includes the total travelled distance, mean & std of instantaneous speed, start-position and end-position
  - Uses HDBSCAN to identify dense clusters in this feature space
  - Labels all demos in the largest cluster as **Full**

    - Applys a median-distance ± tolerance heuristic to split the remaining clusters into **Partial** and **Overshoot**
    - Treats those heuristic labels as weak ground truth, train a shallow Decision Tree on the same features, and predict final classes to smooth boundary cases
    - Use case: We are able to identify complete trajectories in the dataset as opposed to partial/ incomplete paths, or overshoot paths for the task being completed
- **Segmentation**
    - This takes L1 level expert dataset to create L2 level dataset, where each trajectory is split into multiple sub-trajectories based on dense regions
    - Uses HDBSCAN to identify dense regions (clusters) of trajectory states
    - Uses cluster centers as a segmentation point
    - Uses the segmentation points to separate trajectories into multiple parts.
    - Use case: Automatically separate trajectories into multiple areas of interest so that users can evaluate the dataset only in regions that matter the most. For example: moving the lampshade from the initial position to the position of the lightbulb, putting the lampshade on, etc



Application

We created a web-based viewer to view plots generated by our scripts. Plots range from trajectory density plot, entropy estimation, averaged trajectories, and segmented trajectories.

Future Work

We aim to use a more robust general representation for our trajectories states and actions, that can work with any type of robot. One method is to use Representation Learning to generate learned representations of the robot's states and actions using Variational Autoencoders (VAE). Furthermore, we also aim to implement methods to generate metric that directly evaluates trajectories based on their contributions to the learned policy such as the *Demonstration Information Estimation* (Hejna, et al 2025).

We also want to extend to a second platform (like the SO-100 arm from HuggingFace LeRobot)

Potential areas of weakness for future work:

- We cluster by states
- Because of ML based models we need a significant amount data (must be large-scale, at least 40ish demos)
- Hyperparameters are hard-coded with a reasonable benchmark for entropy, HDBSCAN with minimum number of datapoints as 20%
- As the dataset gets bigger, the script takes longer to generate the visualizations
    - Our attempt at resolving this issue (introduced pipeline to generate everything as soon as the user uploads the data in an interactive web viewer)
- We currently can't do online learning or parametrize HDBSCAN