NASA ULI Shared Simulation Platform

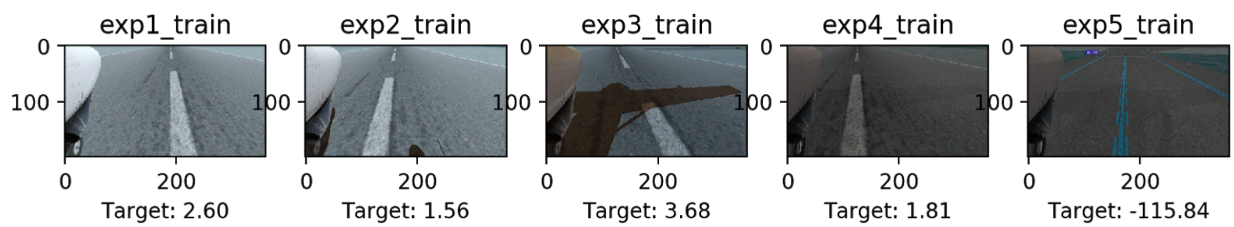
Authors: Sandeep Chinchali (csandeep@stanford.edu), Apoorva Sharma (apoorva@stanford.edu), Sydney Katz (smkatz@stanford.edu), Marco Pavone (pavone@stanford.edu)

Overview

The purpose of this document is to specify a shared simulator and dataset for all groups in the NASA ULI project. The simulator, based on X-Plane, will simulate autonomous taxiing and landing on a runway. Development will be spearheaded by MIT Lincoln Labs. The goal is for each group to be able to retrieve (i) perception data from cameras on-board a simulated aircraft, (ii) state information, and (iii) implement closed-loop control policies.

Technical Specifications

Simulation Platform



We will collect data from the X-Plane 11 simulator. Our first scenario should have a Cessna taxiing on-board a runway lane, where the goal is to simply track the center-line of the lane using an on-board camera placed on the plane's wing. A useful package to store data from the simulator is XPlane-connect, which is cited in the references section. Furthermore, the X-cam renderer toolbox can be used to access camera images.

Dataset and Logging

The dataset should log the following information as a continuous video sequence, stored as arrays in a Python pickle file:

1. Image Information
    a. RGB image
    b. Camera number
2. Timestep information
    a. Simulator Absolute UTC Time

    b. Relative time-step from start of simulation
3. Plane state information
    a. Heading angle relative to runway center-line
    b. Position along runway
    c. Lateral Distance from runway center-line
4. Environmental Conditions Information
    a. Period of day
        i. Number values between 0 and 2
        ii. Morning (5 AM - 12 pm): 0
        iii. Afternoon (12 pm - 5 pm): 1
        iv. Night (5pm - 5 AM): 2
    b. Weather conditions - fog, rain, etc.
        i. Analogous to Step 4

The individual images and corresponding state information should be compatible with deep learning data-loaders, such as a Pytorch Dataloader or Tensorflow dataloader. A common use case is to learn a convolutional deep neural network (DNN) to map from the camera image to predict the lateral distance from the runway center-line.

Using X-Plane connect, the dataset can be generated by driving the plane along straight-line and sinusoidal patterns around the center-line and saving the data.

Closed-Loop Control

Finally, we want to show a simple PID (proportional-integral-derivative) controller can move the plane to the center-line. Given the current state and the desired reference, the PID controller should send commands through X-Plane connect to move the aircraft to the center-line.

Software Development

Development should be done in Python 3.x. As a unit-test, we want a Jupyter IPython notebook to be able to visualize images and corresponding state information at any given time-step.

Further, we want a Pytorch 1.x dataloader to be able to load a batched dataset of images and corresponding labels of heading angle information so that users can train CNNs directly from the data.

Prior Experience in DARPA grant

- Used a similar framework built by Boeing for the DARPA grant.
- Using mss to capture image feed from xplane display
- Collected 30,000 frames from sinusoidal trajectories in 5 groups
    - Morning, clear conditions

- All day (morning - afternoon), clear conditions
- Afternoon (clear conditions)
- Afternoon (cloudy conditions)
- Afternoon (cloudy conditions, different airport)

References

Pytorch data-loader:
https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

X-plane connect:
https://github.com/nasa/XPlaneConnect

X-cam renderer:
https://www.stickandrudderstudios.com/x-camera/download-x-camera/

Boeing wayfinder:
https://acubed.airbus.com/blog/wayfinder/how-wayfinder-is-using-neural-networks-for-vision-based-autonomous-landing/