CMSC 35360 Autonomous Laboratories HW2

Kevin Wu, Irene Madejski, Bayard Walsh, Lucas Mantovani

Introduction

Sterling Baird's light-mixing SDL-Demo is a self-driving lab demonstration that showcases the capabilities of autonomous laboratories in the context of a simple experiment involving light mixing. We ran the experiment eight times on a local Raspberry Pi, with four having different RGB target configurations and four being initialized with the same target (R25_G25_B25) and distinct random seeds. We included graphs (see Figures section at the end) comparing the results on the local and remote machines for some of the results obtained. We also implement a multi-device system, combining two local devices and one remote device, and perform two sets of experiments testing the accuracy and speedup of the multi-device system.

Running the Experiments

Full results, including CSV data and graphs for each experiment, can be found in `light-mixing.zip`.

To run all experiments, run the shell script `experiments.sh`. Note that you will need two Pico W devices, and replace Pico ID's with you device.

```
PICO_LOCAL_1="..."
PICO_LOCAL_2="..."
```

You can also run individual experiments with one device, or with batch optimization on multiple devices (see `README.md`).

Single Local Device

Setup

To perform our experiment, we used a self-driving lab composed of a Maker Pi Pico base, a RPi Pico W and a AS7341 Light Sensor. We set up our Raspberry Pi Pico W with the steps outlined on the Raspberry Pi website, and used MicroPython with Thonny to control the device and to retrieve the PICO ID. We inputted the PICO ID into the provided SDL-Demo script to run the demo locally and obtain the desired results.

Results and Analysis

First, we performed an experiment on the local device that was identical to the experiment performed with the remote device – which varied both seed and target RGB colors. We expected very similar results to those collected from the remote device. The most notable change between the two experiments was the change in environment in which the experiment was run. The sensor is sensitive to changes in background light, and although our local device experiments were performed in a consistent setting, we did not eliminate all surrounding light. When looking at the local results, we noticed that the Bayesian search method provided end-results that varied +/- ~10 for each of the three RGB values from the target. This is different from the previous remote results which provided and end result closer to the actual target, with a variation of only +/- 2 for each of the three RGB values from the target. The only exception was for the target R25_G25_B65, which was significantly worse for the remote than the local (Figure 3). However, this was an outlier for both the local and remote experiments, and we attribute this descrepancy to the fact that differences in environmental lighting across both cases led to an exacerbation of the distance to target on the remote but not local case. Nonetheless, it is worth noting that the Bayesian search method still performed consistently better than the random and grid search methods across the remote and local cases for both varying seed and varying target.

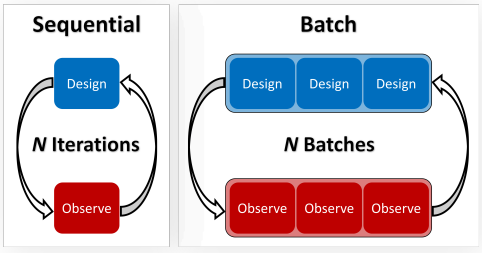
In terms of the search iterations for both varying seeds and varying targets, we did not notice any significant difference in the best estimates obtained across multiple iterations for the two experiments besides the expected variability in the initial value, which was expected given that different seeds were used on the first assignment than on this one. However, we noticed one difference that we do not attribute to the difference in seed used: on our third random seed experiment with the target R25_G25_B25 (Figure 4), the Bayesian method took much longer to converge on the local device than on the remote one. We attribute this difference to distinct lighting conditions that may have affected the local experiment but were not present in the remote one. For instance, it is possible that a flash of light near the device was turned on and off while running it locally, a variation in lighting that likely made convergence take longer on one of the 8 experiments.

Lastly, when comparing the Frechet distances between the local and remote experiments for both varied seed and for varied target, we notice that the grid and random search methods consistently display the exact same layout of the datapoints in the tridimensional space, with the only difference being the distance assigned to each of the points. On the other hand, the Bayesian optimization method consistenty indicated a difference in both the tridimensional layout of the points in the RGB space and in the Frechet distances assigned to each of them. The same layout of the datapoints across the grid and random search processes can be attributed to the fact that both methods used the same parameters in the search process. On the other hand, we attribute the smaller scale of distances obtained in the local experiment (0 to 7k) than the one executed on the remote machine (0 to 18k, nearly twice the scale of the local one) to different lighting conditions across both settings. More specifically, we were surprised to see that the scale of Frechet distances of the remote experiment was larger than the local one, since this seemed to contradict our assumption that the local experiment was operated under more variable lighting conditions than the remote one. However, upon further reflection, we concluded that an explanation compatible with all of our results is that the remote condition is consistently lit (which explains the lack of variation in the remote server for the iterations shown in Figures 1) and 2), but that the total lighting was higher for the remote server. On the other hand, we believe that the difference in the layout of the datapoints for the Bayesian optimization process was caused by both the difference in lighting (which might have generated worse previous estimations used to obtain the subsequent ones through the Bayesian method) and in the initial seed used (which differed between the remote and local cases, since a random seed was generated in every experiment across the two scenarios).

Combining Devices

Batch Optimization

We follow the batch optimization example provided by Sterling Baird: https://self-driving-lab-demo.readthedocs.io/. Batch optimization chooses multiple points at once, choosing points that would likely be good together in each step. These points are evaluated in parallel on seperate devices, then sent to the model for fitting. If we have a computationally intensive evaluation of the objective (which is not necessarily the case in this experiment), this can reduce the total experiment duration.



Implementation

See `sdl_light_mixing_batch.py` in the code, which performs batch optimization for a light mixing experiment using Meta's Ax optimization library and Ray for parallel computation. We combine two Pico W microcontroller devices and the remote Utah device, for a total of three devices.

We initialize a batch of `SelfDrivingLabDemoLight` instances, one for each PICO device, with the target color and other settings. Like the single-device experiments, the search space is defined by the bounds of the RGB values, and the objective of the optimization is to minimize the 'frechet' distance.

Ray is initialized for parallel computation, and an evaluation function is defined to evaluate a set of parameters (RGB values) on a SelfDrivingLabDemoLight instance. This function is decorated with `ray.remote` to allow it to be run in parallel on different cores or machines.

An Ax client is created and an experiment is set up with the defined parameters and objective. The script then enters a loop where it repeatedly asks the Ax client for the next set of trials (sets of parameters to evaluate, three in our case), which are generated using SOBOL for the first six trials and Bayesian Optimization for the rest. The trials are evaluated in parallel using Ray, and reports the results back to the Ax client, which updates the Bayesian Optimization model.

Experimental Setup

We repeat each of the following experiments ten times with different random seeds on the target RGB = (25,25,25).

1. 27 total iterations
   - Since we have three devices, this results in 9 batches
   - Motivation: we can compare the same number of iterations as we did with one device, expecting faster completion with similar results
2. 81 total iterations
   - 27 batches
   - Motivation: we expect the same experiment completion time with better results (lower frechet)

Results and Analysis

See `results.ipynb` and Figure 5 for two example search plots using three devices.

Experiment	Mean Frechet	Standard Deviation
Single device, 27 iterations	458.0	205.9
Three devices, 27 iterations	713.5	174.5
Three devices, 81 iterations	384.0	270.4

We see that the mean frechet with the same number of iterations is higher for three devices (albeit with a lower standard deviation). This could be because the Bayesian Optimization model is only trained after every three observations rather than after every observation, so the model outputs higher quality predictions when the experiment is sequential.

The mean frechet with 81 iterations is slightly lower than with 27 iterations on a single device but with a higher standard deviation. This result makes sense, as we see that the line plots in Figures 2 and 3 show that the trend of best frechet already flattens out by 27 iterations. However, running 81 iterations took around 5:50 minutes per experiment, while a single device took around 2:30 minutes per experiment for 27 iterations. Ideally, we should see a similar amount of time for these

experiments. This is because in both cases, we are trying to fit one model. Parallelizing experiments can increase speedup when the experiment itself takes a long time, because the part of the code that is parallelized is only the experiment. In our case, the experiment is just a flash of light, so the total time is dominated by the model fitting and trial generation, which are not parallelized.

Figures (single local device experiments)

Configurations

We tested eight RGB targets:

R25_G25_B25,R25_G25_B65,R25_G65_B25,R65_G25_B25
R25_G65_B65,R65_G25_B65,R65_G65_B25, R65_G65_B65

We present the results for selected targets below.

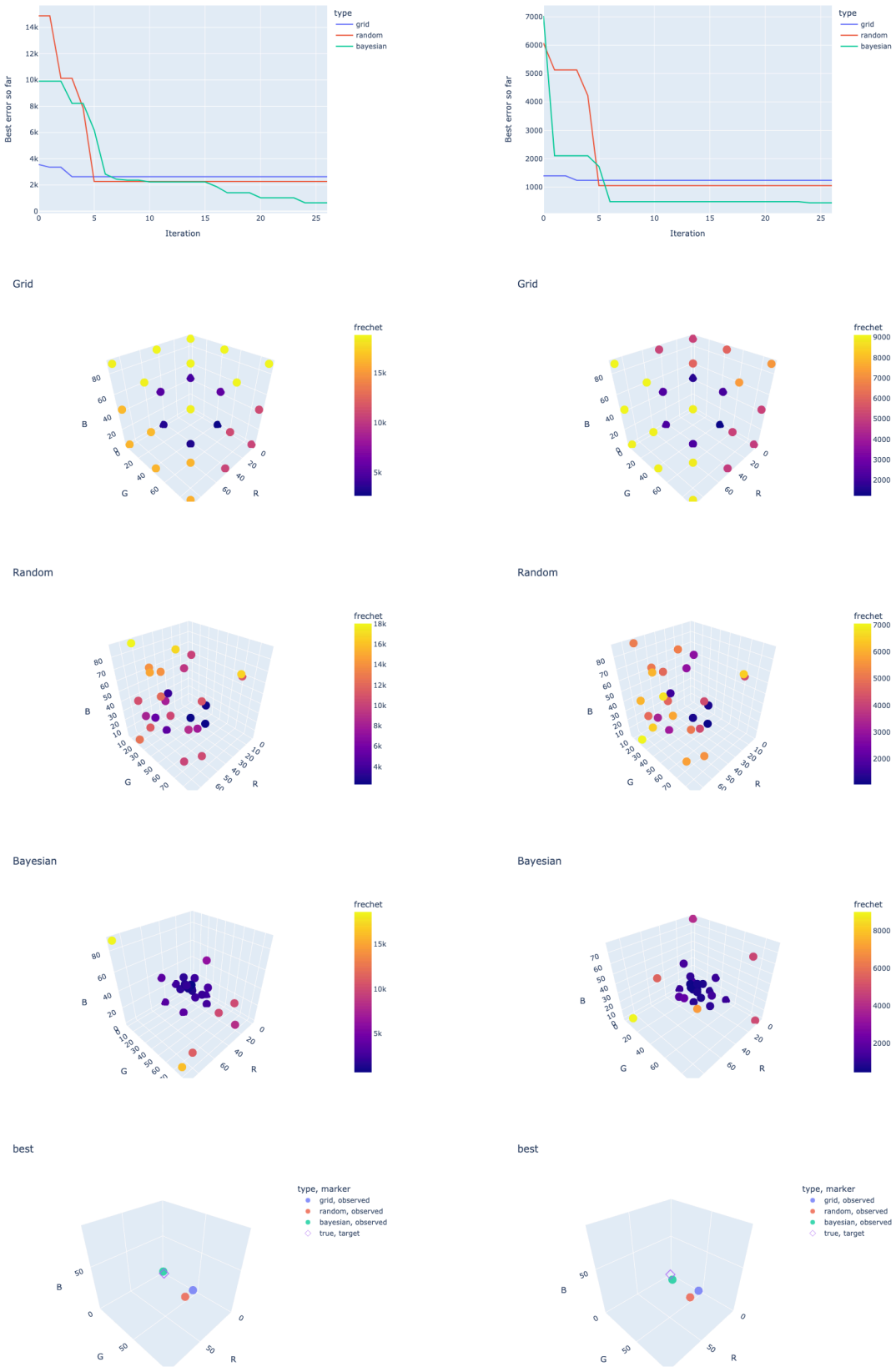


Figure 2: Comparision: remote (left) and local (right) R25_G25_B25 (random seed 3)

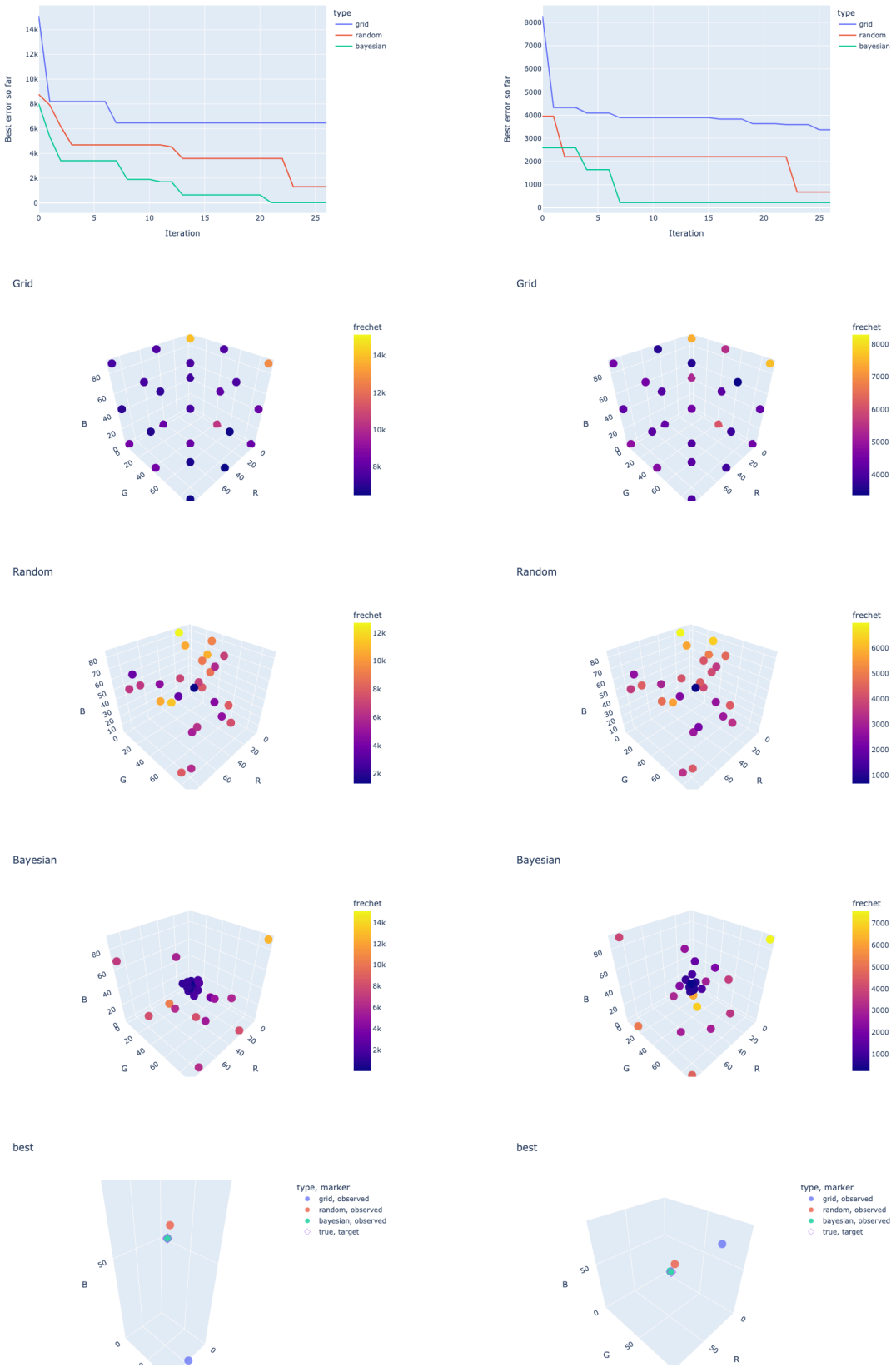


Figure 3: Comparision: remote (left) and local (right) R65_G65_B65

Figure 4: Best performance comparision: remote (left) and local (right) R25_G25_B65

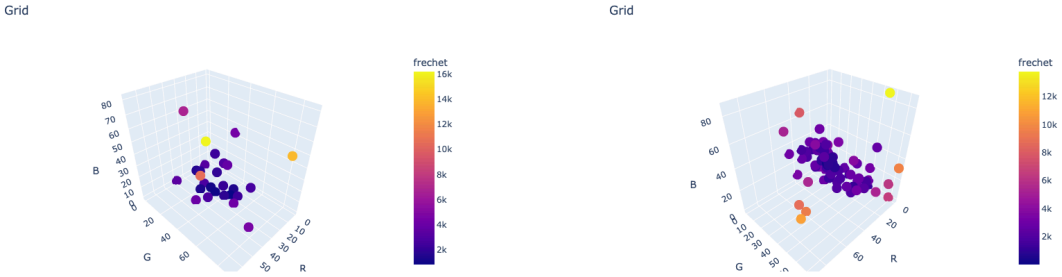


Figure 5: Three devices, 27 iterations (left) and 81 iterations (right) R25_G25_B25