ARO Project 2021
USER MANUAL

Author: Andrew O'Kins
Date: 01/24/2022

Table of Contents

Introduction

The ARO Project is a program to find a sufficiently optimal series of images for spatial light modulators (SLMs) to generate a high intensity signal image from a camera. The program offers three algorithms to achieve this; sequential (brute-force), simple genetic algorithm, and micro genetic algorithm. Through the graphical user interface (GUI), a user can configure various parameters for the camera, SLMs, optimization algorithm (type, how many generations, bin size for images, etc.), and the kind of expected outputs during/after the optimization. The program also offers multithreading to increase the efficiency and speed towards finding results.

Requirements for this program are libraries needed for interacting with the hardware and image data. The following libraries/SDKs are Spinnaker and Blink_PCIe.

The User Interface

Section 1: Startup & Main GUI

When starting up the application, two windows will be generated. The first is a console window which will provide verbose text information and also will always provide updates during optimization. The second, smaller, window is the GUI interface which will be where you can configure the setup and start/stop the optimization process for given boards. This GUI is structured so that some prominent options are always available while more specific parameters are contained in subject-based tabbed windows. The rest of this section will go over the main GUI that is not encapsulated in one of these tabbed windows.

On the top row are “Save Settings” and “Load Settings”, when configuring a setup you can use these two options to save or load them. When saving, the parameters are saved in a readable configuration file that can be edited with a simple text editor if needed. When loading from a file, it will attempt to assign all LUT files to all the boards that existed when being saved and will give a warning if there are failures to do so. The configuration files will not contain the LUT file data, so be sure to have those in the same file path as when saving. The console window will also provide updates during the loading as well. When making changes, be sure to save before closing as there are no warnings to save them.

Next to the settings options is a checkbox for enabling multithreading. When enabled, running a genetic algorithm will utilize multithreading with the exact number of the threads dependent on the parameters within Optimization Settings. Generally, enabling multithreading will lead to a faster run of the algorithm. The iterative algorithm does not use any multithreading however, so for that choice there will be no difference.

Below these options are 4 buttons, with 3 of them being labeled for choosing an optimization algorithm to use and the right-most labeled “Start Optimization”. uGA and SGA are the two provided genetic algorithms, and IA is the iterative algorithm. If none of the algorithms have been selected yet, the start button is disabled. After selecting an algorithm, you may start the algorithm by pressing the start button. Once started, the start button becomes the abort button in case that is needed during the run to safely stop.

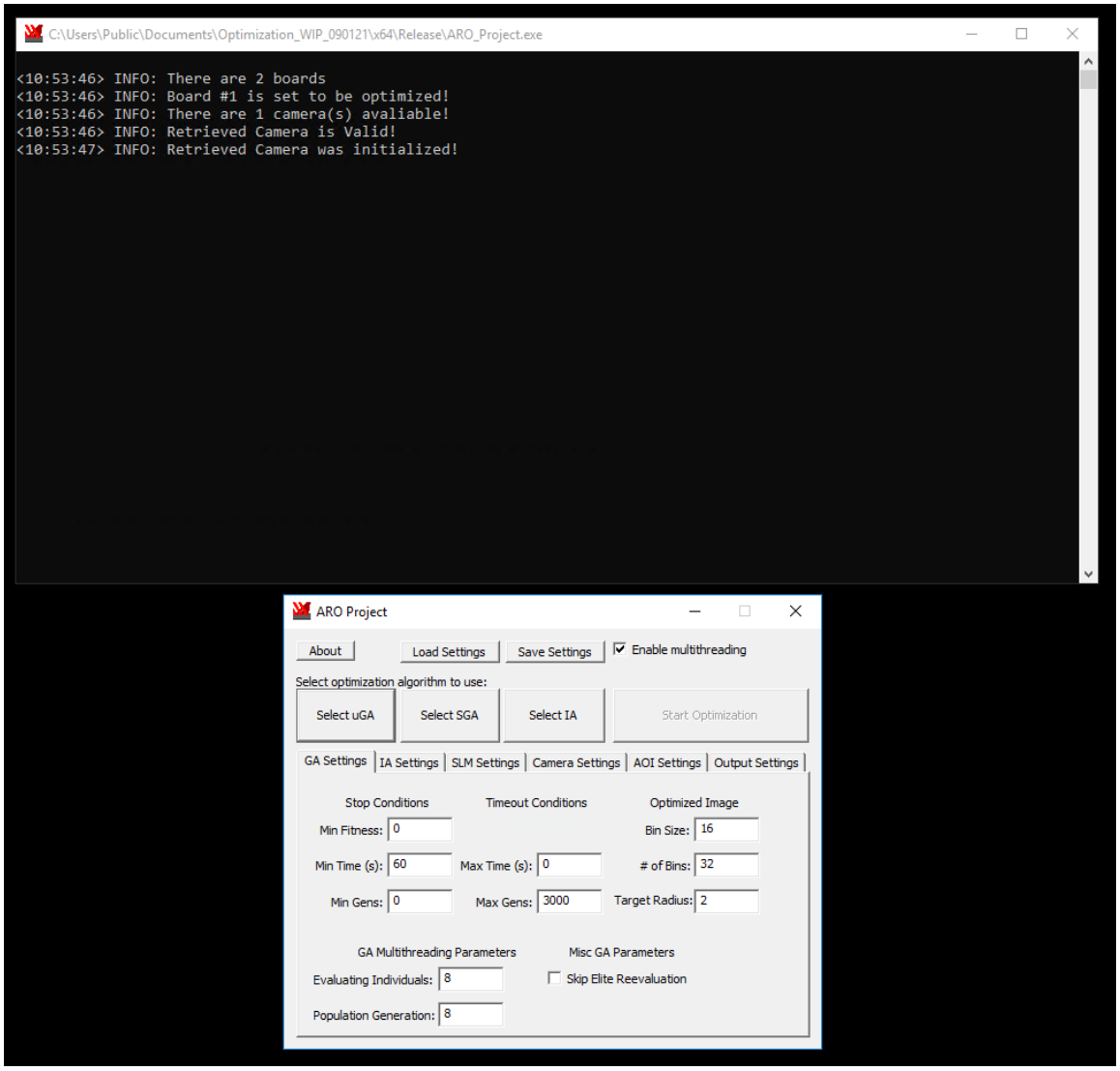

Figure 1. Start up with 2 SLM boards and 1 Camera connected with the tabbed window open to GA Settings.

Section 2: Optimization Settings (GA & IA)

Parameters are given within their categories

Stop Conditions: All these conditions must be met before ending the algorithm's run.

- *Min Fitness* - The minimum average intensity within the target radius of the camera image to achieve.
- *Min Time* - The minimum amount of real-world time the genetic algorithm must run in seconds.
- *Min Gens* - The minimum number of generations the genetic algorithm must perform.

Timeout Conditions: If any conditions are reached, the algorithm will end regardless of results.

- *Max Time* - The maximum amount of real-world time the genetic algorithm is allowed to run in seconds. Set to 0 for indefinite.
- *Max Gens* - The maximum number of generations the genetic algorithm is allowed to perform. Set to 0 for indefinite.

Optimized Image: These determine the characteristics of the optimized SLM image(s.

- *Bin Size* - Size of a given bin in the SLM image for optimization.
- *# of Bins* - The square dimension of the resolution of the SLM image for optimization (example: a value of 10 would have 100 bins).
- *Target Radius* - The radius from the center of the camera image (in pixels) that will be considered for increasing intensity, the rest of the camera image will be ignored in the evaluation.
- *Phase Resolution (IA Only)* - The step-size within a bin to find an optimal value.

GA Multithreading Parameters: Determine the number of threads used if multithreading.

- *Evaluating Individuals* - Number of threads to use when performing the evaluation process of the individuals in a pool.
- *Population Generation* - Number of threads to use when performing the genetic crossover/breeding process in a population pool in a generation.

Misc GA Parameters: A section for other parameters, currently only has one.

- *Skip Elite Reevaluation* - If toggled, during evaluation of individuals in a generation it will skip those that have already been assigned a fitness (the elites that were carried over).

Section 3: Camera Settings

Two subwindows will be described in this section

Camera Settings: General parameter settings for the camera, some are currently supported in the Spinnaker version

- *Frames Per Second (Spinnaker Version)* - The rate the camera acquires images.
- *Initial Exposure Time* - The initial exposure time the camera uses when getting images (in microseconds). If the resulting image is too high during the run of the algorithm, it may reduce it during runtime.
- *Gamma Value* (*Spinnaker Version)* - The gamma intensity of the image data, separate from the exposure time.

AOI Settings: A subset given in a separate subwindow, for the Area of Interest in the camera.

- *AOI Control Parameters* - Four fields to set the width/height and offsets for the AOI
- *Center AOI* - Takes current width/height and sets to have contents in center of camera's larger view window.
- *Max Image Size* - Sets the AOI to be the entire camera image view window.

Section 4: SLM Settings

Within SLM Settings window, in the top-left the user may choose from a list a connected SLM to configure. As an option, checking “Set All” will have any further adjustments within SLM Settings be applied to all the boards.

- *Optimize this board* - Check this to include this board to be used in optimization.
- *Turn power ON/OFF* - Power button for the selected SLM
- *Set LUT* - Browse and select a LUT file (default filter for .lut extension), updates current assignment path in SLM settings.

Section 5: Output Settings & Details

Output Folder: A field-box and browse button to set a path to a folder where all file outputs will be made to.

Runtime Displays:

- *Display Camera* - Output to the screen as a separate window the best camera image during optimization runtime (for GAs, this will be updated in every generation).
- *Display SLM* - Output to the screen as separate window(s) the best SLM image(s) during optimization runtime (for GAs, this will be updated in every generation).

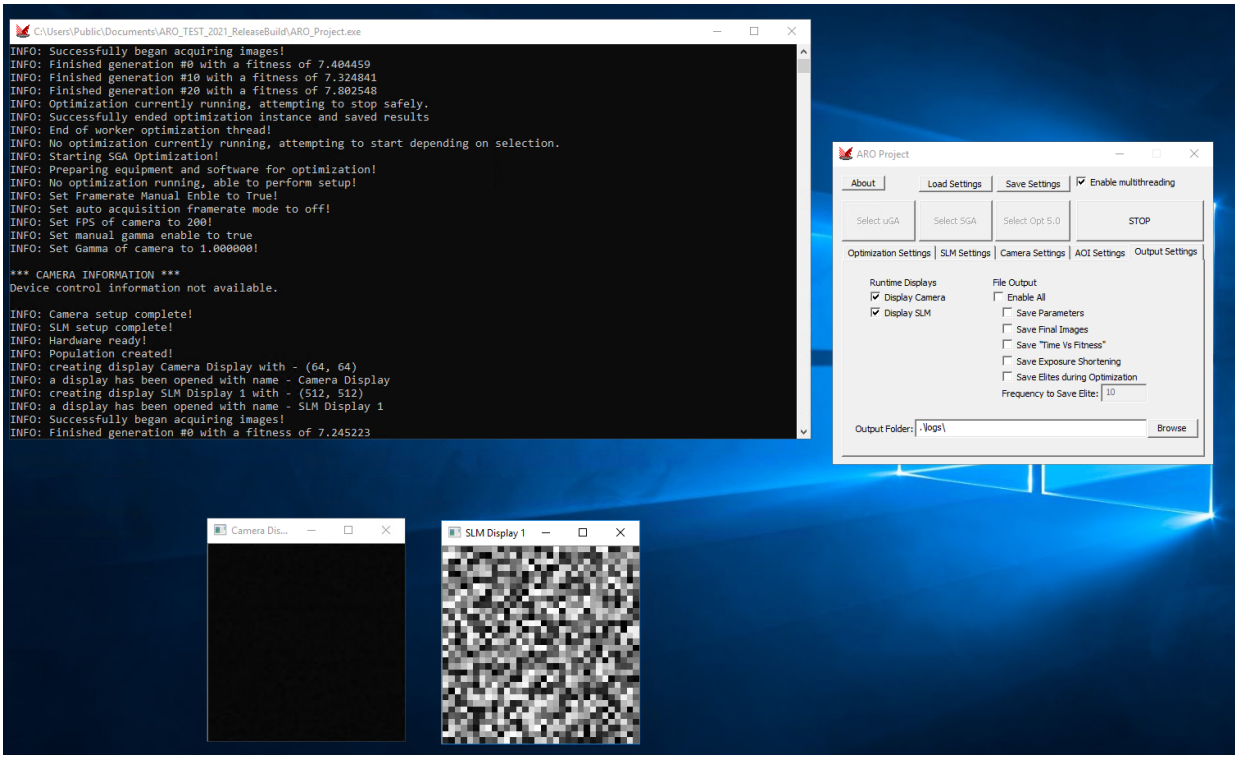

Figure 2. Test Demo of Optimization in Process with toggled Image Displays

File Output:

- *Enable All* - Shortcut to enable all file outputs that also are listed as separate toggles.
- *Save Parameters* - When running optimization will save the current setup as both a config file that can be loaded/edited and an additional verbose text file.
- *Save Final Images* - Save the resulting optimized camera and SLM images in bitmap format. If a timeout condition occurs, it will not save the resulting images.
- *Save "Time Vs Fitness"* - Records the time performance of the program during optimization run.
- *Save Exposure Shortening* - Logs when exposure was shortened from initial during the optimization run.
- *Save Elites during Optimization* - Save camera and SLM images during optimization run to observe change over optimization run at a later time.
- *Frequency to Save Elite* - If saving elites during optimization, the frequency by generation to save the progress (example: input of 10 would save every 10 generations).

File Format Details

All files are prefixed with a timestamp containing date and time down to seconds, followed by algorithm name (for example the simple genetic algorithm will have files prefixed with "SGA" and the micro-genetic algorithm with "uGA").

If *Save "Time Vs Fitness"* is toggled, 2 files will be produced after running the optimization algorithm.
The first is entitled "time_vs_fitness.txt" and contains in a row the following:

- Time elapsed since start in milliseconds
- Fitness of current individual that has been evaluated
- Exposure time, and the ratio of it compared to the initial exposure time.

The second is focused on performance of parts of the algorithm and is entitled "timePerformance.csv".

- In the top row along with header info is also a record of the thread counts given by the GUI parameters (as described in Optimization Settings), this is to help record the difference multithreading has on performance between runs.
- Each row has the following info:
    - Current generation
    - Time (in microseconds) to evaluate the pool of individuals
    - Time (in microseconds) to perform genetic crossover/breeding to produce new pool.
    - Overall time (in microseconds) to process the entire generation. This is expected to be approximately (but slightly greater as it includes the other outputs) equal to the sum of the previous recorded times.
- At the bottom of the file is a record of the overall time of the run in microseconds.

The Configuration File

Briefly mentioned within Section 1 is the ability to save and load settings. When saving, it is saved in a text-based configuration file that, if desired, can be accessed with any text-editor to adjust parameter values. It is ordered such that the parameters are separated out by the dialogs (GUI regions) that the values affect and have names comparable to what is in the GUI.

As long as they are parseable, it is possible to directly modify these values in the file and then load the result through the GUI's "Load Settings" button. The available parameters are as given, omitting some will result in them not being assigned (nor reverting to any "default") and adding unrecognized ones will bring warnings within the console window.

The syntax for successfully being parsed is the following layout:

<variable name>=<value>

With support for commenting either after the value (separated by a space) or in a new line with the `#` character. Some comments are provided when saving and may be useful if wishing to note particular values or other important details regarding a particular configuration setup.