Table of Contents

Setup

a. MATLAB GUI

i. Raspberry Pi Cameras

To set up the MATLAB GUI so that it is ready to use there are a few things that need to be edited. Make sure to set up Raspi connections to all cameras at the top of the GUI as “app.pi_n = raspi('ip', 'username/hostname', 'password')”. Uncomment the Pis that are being used and comment the ones that are not being used. Then make sure that all lines containing the array of cameras (cameras = {app.pi1, ..., app.pi_n};) contains all cameras currently being used. As of right now, the array can be seen around lines 91, 139, and 319. Also, the order that the Raspi objects are placed in the camera array matters. When selecting the camera number the index of your Raspi object is used. So camera #1 in the GUI will correspond with the first camera in your cameras list.

ii. Image Folder

Make sure that you have created a folder (it can be anywhere) that you want to store the images in. You will need this folder path for storing the images taken during the execution of calibration/experiment runs.

iii. Python Script

Make sure that the computer being used has the Python from the Microsoft store downloaded. Make sure that the opencv library is installed on the computer. This can be done in the command prompt with 'python' (to install python from the MS store), and once python is installed run 'pip install opencv-python'. Opencv is a library that the python script uses for unpacking images. The script should be named 'showImg.py' and in the same directory as the cameras.mlapp (MATLAB GUI). If this is not the case, please provide the custom file location under the ‘py_script’ variable around line 66. showImg.py converts RAW image files containing multiple images into individual TIFF image files.

iv. On Startup

Provide the image folder path under the "image directory" box and press "Go". Do not have any whitespace or quotation marks surrounding your file path. If the app successfully launched and you have entered an image directory then the app is ready to go.

b. Raspberry Pi

i. Setting Up Pis

Getting the Pis to work with MATLAB and the Raspi library might provide difficulties at first if it is a new pi. Just keep on installing those libraries when it throws errors until it doesn't anymore. Run 'pip install {library name}' on the Raspberry Pi to install the library MATLAB wants. For the setup that is used with the app, the Raspberry Pi camera should be using the pixel format Y10, with a width of 1280 and a height of 800. There should also be directories named "Live", "Cylinder", and "Experiment" inside of the "/home/afdl/Documents" folder on the Pis. If there are not, log into the Pi then run "cd ~/Documents" followed by "mkdir Live", "mkdir Cylinder" and "mkdir Experiment".

ii. Image Files on Pis

The Pis store images taken from the camera either in the "Live", "Cylinder", or "Experiment" folder based on what is being used. During the live camera view of the app the image is stored in "/home/afdl/Documents/Live" as "test.tif". During checkerboard, the same system is used, but the files are saved after retrieval from the "Live" directory on the Pi. For Cylinder and Experiment, there is a set number of photos streamed to a RAW file. The file is saved as "camera.raw" under the respective directory.

Each pi should contain the file ‘raw_photo.sh’which is edited in the app to have the input desired number of images. The sh file is used to take images on each Pis camera which are then streamed to the Cylinder/Experiment folder as “camera.raw”

Functionality

a. Properties

The app comes with many properties, each stored from lines 4 through 85. There are properties that correspond to app components which just establish all visual aspects of the app. Next, there are all the properties that correspond to the live view popout or underlying components. The first 'matlab.ui' properties establish the popout and the ability to close it. Then there are different default values for the root file node, screen/button/image/exposure/gain/path values. Then image and python script path and all cameras labeled as pi1 -> pi(n).

b. Functions

*For all functions, ignore the app input in the function call. This just allows for properties of the app to be used in the method. Any function that contains "event" means it is a callback function. These are executed when some kind of interaction occurs in the GUI (ex. button pressed).

1. rpi = GetPi(app, index)
   *Used to get a Raspi object.
   Input: index - position of the camera in the cameras list
   Return: rpi - Raspi object of camera at the given index in the cameras list

2. createClose(app)
   *Creates the close button on a live view popout after the "Live View" button has been pushed.
   No input or return values.

3. ClosePopout(app, ~)
   *Deletes the live view popout after the close button has been pushed.
   No input or return values.

4. img = GetImage(app, index)
   *Get a live TIF image. Used for checkerboard test type and live images.
   Input: index - position of the camera in the cameras list
   Return: img - TIF image file representing the TIF image inside of the "Live" directory on the Raspberry Pi.

5. DisplayLiveFeed(app)
   *Runs the GetImage function and displays the image and the intensity while the live view popout is active.
   No input or output.

6. SetExposureAndGain(app, exposure, gain, type)
   *Set exposure and gain for all/one camera.
   Input: exposure - Desired exposure.
   Input: gain - Desired gain.
   Input: type (0 or _) - entering type == 0 will update all cameras while any other entry will only update the camera in the Camera Edit Field (Camera #_).

7. popTree(app, start_point)
   *Populate the file tree with the entered image directory.
   Input: start_point - This is the starting point of the file tree. The start_point is going to be whatever image directory is entered in the "Images Directory" edit field.

8. popDir(app, dirNode, folder, dirName)
   *Get information on directories inside of the parent directory.
   Input: dirNode - the directory you want to populate
   Input: folder - parent folder of directory
   Input: dirName - name of folder

9. dirchk(app, fullpath)
   *Checks if a directory exists and if it doesn't the directory is created.
   Input: fullpath - absolute path of directory

10. EnterImageDir(app)
    *Highlight the image directory edit field in red for 1 second to remind the user to enter an images directory before proceeding with the app.
    No inputs or outputs.

11. run_python(app, saveFolder, nImgs)
    *Runs the python script to unpack images on the host computer from a RAW file with multiple images to individual TIFF files.
    Input: saveFolder - directory on host computer where you store the converted images
    Input: nImgs - number of images in the RAW file

12. redrawTree(app)
    *Redoes the file tree in the middle of the screen. Used after a calibration/experiment has been run.
    No inputs or outputs.

13. send_signal(app, i)
    *Executes the raw_photo.sh file on a given Pi index.
    Input: i - index of Raspberry Pi camera

14. startupFcn(app)
    *Runs on app startup. Sets the default exposure and gain on all cameras (default value found under properties).
    No inputs or outputs.

15. LiveViewButtonPushed(app, event)
    *Creates the popout and displays a live image feed while the popout is active. Only works if an image directory has been entered. The function is called when the "Live View" button is pushed.

16. OnlyLiveEGButtonPushed(app, event)
    *Updates gain and exposure for the current camera #. (set E&G with type != 0)

17. FileTreeDoubleClicked(app, event)
    *Gets the file path of a file in the tree and displays the file in the TIFF image viewer on the right side of the app. Not working right now.

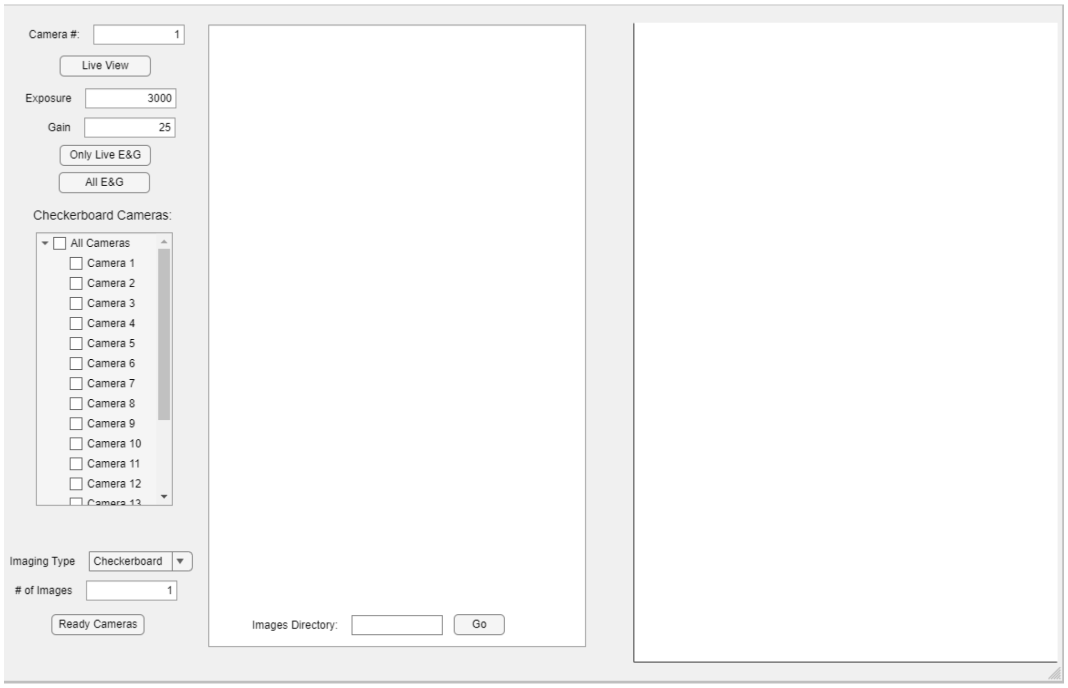
18. ReadyCamerasButtonPushed(app, event)
    *Called when the "Ready Cameras" button is pushed. Executes either checkerboard, cylinder, or experiment for cameras.

19. GoButtonPushed(app, event)
    *Reads in the image directory entered, checks for the cylinder, checkerboard, and experiment folders, and updates the file tree.

20. TreeCheckedNodesChanged(app, event)
    *Updates the "nodes" array to display whether a camera index is on or off.

21. AllEGButtonPushed(app, event)
    *Sets exposure and gain for all cameras.

App Layout



- Camera #: Choose which camera you want a live view of
- Live View Button: start a live view of selected camera
  - Live view started in a popout window. There is a display for the image intensity on the bottom of the screen as well as a close button.
- Exposure: desired exposure
- Gain: desired gain
- Only Live E&G: update exposure and gain for the camera # in the “Camera #” edit field
- All E&G: update exposure and gain for all cameras in cameras array
- Checkerboard cameras: select which cameras you want to do the “checkerboard” imaging type with. The cameras selected will go one at a time, but back to back. I recommend that you just select a single camera at a time.
- Imaging type: choose between doing checkerboard/cylinder calibration or an experiment.
  - Checkerboard: Displayed as a live view on the far right side of the screen in the white box.
  - Cylinder & Experiment: Not displayed anywhere, updates file tree when finished executing.
- # of Images: selects the number of images per camera for an imaging type.
- Ready Cameras: begins execution of an imaging type with the input number of images.
- Images Directory: input the images folder path (ex. D:/MATLAB/images) and hit go. This will update the file tree in the middle of the screen and allow you to see all files inside of the given directory.

Pitfalls

Common errors:

- Checkerboard Cameras - Make sure that the indexes of cameras selected in the checkerboard panel are the indexes of cameras being currently used.

- Connection already exists - This occurs because either a raspi connection already exists in the workspace or because the app is still open. First make sure the app is closed and then go to your MATLAB workspace and type “clear” to remove all variables from the workspace.

- Device is busy - This most likely occurred when some signal is sitting waiting to be executed on the pi. Since the pi’s are in external trigger mode some voltage signal has to be sent to the camera in order for a command to run on the cameras and if this error appears it is because some command is sitting waiting for the voltage signal. Close all MATLAB windows and restart. A way to avoid this is to make sure that you do not close the app while calibration/experimentation is in progress. Run the signal to finish execution and then close.

- Image directory entry - Make sure that the image directory entered has no quotation marks or leading spaces. Enter just the path (Ex. D:\MATLAB\images).

- Python script not working - Make sure that python is installed through the Microsoft Store and that opencv has been installed. Look at “Python Script” under MATLAB GUI for how to install each.