# CPL Imager Manual

## 1 Introduction:

Welcome to the manual for the CPL-Imager, a low-cost, compact circularly polarised imaging system. This system is designed to be quick to assemble and modular enough that it can be adapted to suit a variety of needs (in situ annealing, translation stages, magnets etc). It is primarily designed to image chiral thin-films, but can easily be applied to other contexts, like solutions.

The instrument consists of a Fresnel rhomb (which converts CPL to linearly polarised light), followed by a linear polariser in a piezoelectric rotation mount and a Thorlabs scientific camera. CPL-Imager controls the rotation of the piezomotor, ensuring subsequent images taken by the camera are of alternating handedness (i.e Left-CPL then Right-CPL then Left-CPL etc.). These LCPL and RCPL images are then combined to calculate differential absorbance and Circular Dichroism and these four quantities are then displayed on a live view in the GUI.

Abbreviations used:

- CPL = Circularly Polarised Light. LCPL = Left-CPL, RCPL = Right-CPL
- dA = delta Absorbance
- CD = Circular Dichroism
- GUI = Graphical User Interface
- FR = Fresnel Rhomb
- TL = ThorLabs

## 2 Parts list:

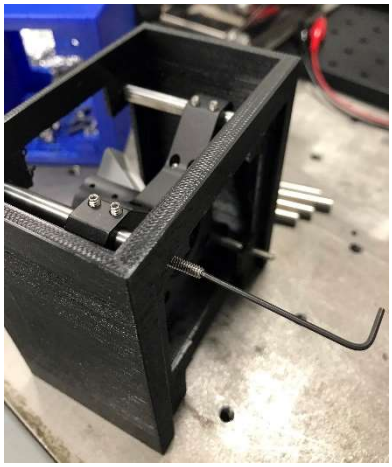
1. Mounted quarter-wave Fresnel rhomb retarder (Thorlabs, FR600QM)
2. Wire grid polarizer (Thorlabs, WP25M-VIS)
3. Piezoelectric rotation mount (Thorlabs, ELL14K)
4. USB controlled monochrome Zelux® 1.6 MP CMOS Cameras are used (Thorlabs, CS165MU/M)
5. 6 mm focal length f = 1.4 Navitar lens (Thorlabs, MVL6WA)
6. Rods, 2", 4 pack (Thorlabs, ER2-P4)
7. Rods, 3", 4 pack (Thorlabs, ER3-P4)
8. 4-40 Capscrews; 3/8", 50 pack (Thorlabs, SH4S038)
9. 4-40 Setscrews; 1/2", cup-tipped, 50 pack (Accu, SSU-4-40-1/2-A2 or Amazon)
10. Black 3D printing ABS or PLA plastic (Amazon)

# 3 Assembly:

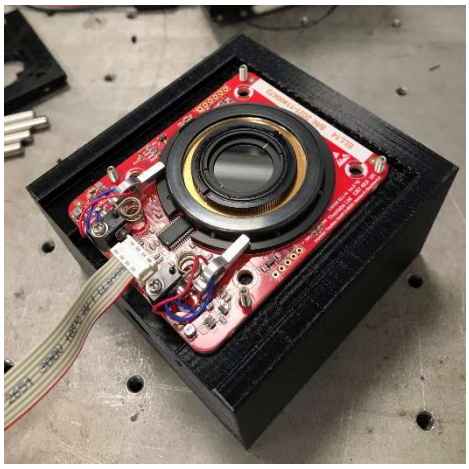
0) Print all the 3D printed parts found in the 'build' directory of this repo

1) Place the FR into the 30-60mm TL cage adapter, ensure it is aligned at 45° from the vertical then fix it in place with 2 slip rings. Screw in the 2" rods in to the 60mm spaced holes, ensuring the rods come in front of the adapter (i.e in the direction of the fresnel rhomb) and are evenly spaced. Leave some amount of the rod behind the adapter so that the adapter is not in direct contact with the rotator when resting on top.

2) Place this construction in the 3D printed front body, then screw the setscrews into the ends of rods facing away from the FR.



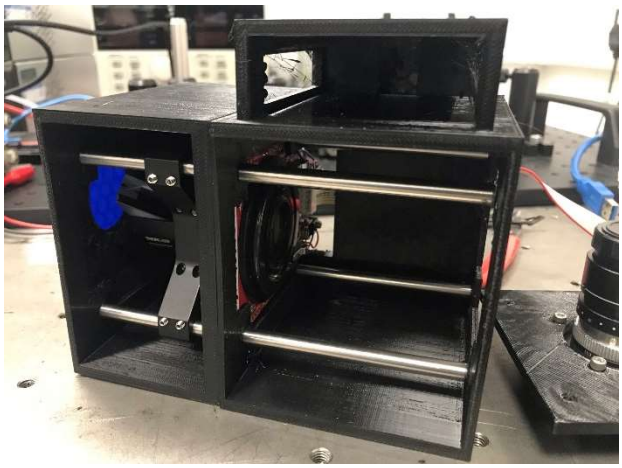
3) Place the linear polariser into the ELL14 rotator. Ensure the linear polariser is aligned at 0° to the vertical, even if the rotator casing is slightly angled (every ELL14 has a slightly different home angle, this is accounted for in rotator.py).



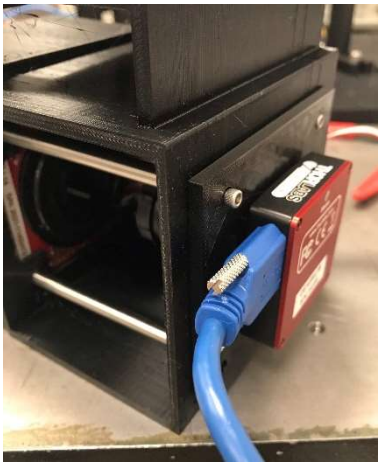
4) Place the ELL14 piezoelectric rotator on the back of the casing such that the setscrews poke through the holes in the rotator.

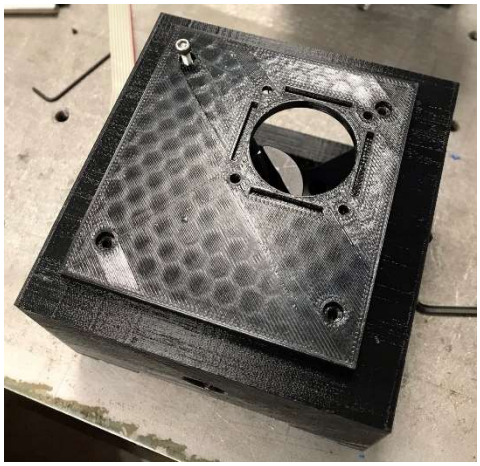
5) Place the 3D printed back body case onto the poking out screws. Be careful not to damage or knock any of the electronics on the piezo board whilst doing this (though the casing is designed to account for this)

6) Starting from bottom to top, screw in the 3" rods to the setscrews in the back body casing by hand. This step is tough - there isn't a lot of give in the casing and there can be slipping. Be careful the rods don't hit the rotator.



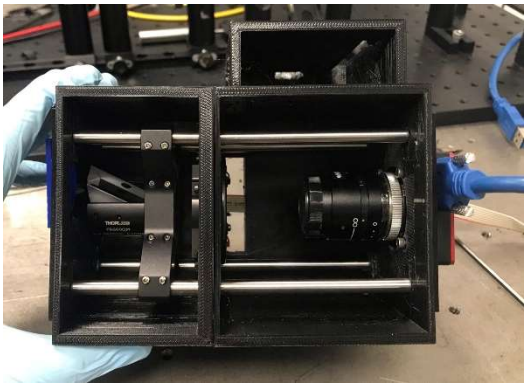
7) Screw the capscrews through the back plate into the camera, then screw more capscrews through the reverse of the plate into the rods so the camera sits enclosed in the casing.

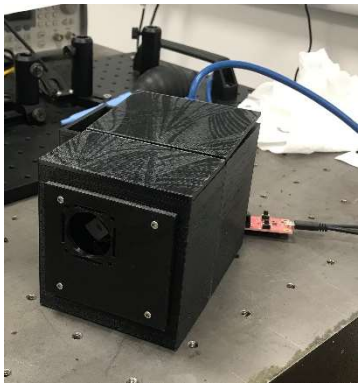




8) Screw the capscrews through the front plate into the rods inside the front body casing.

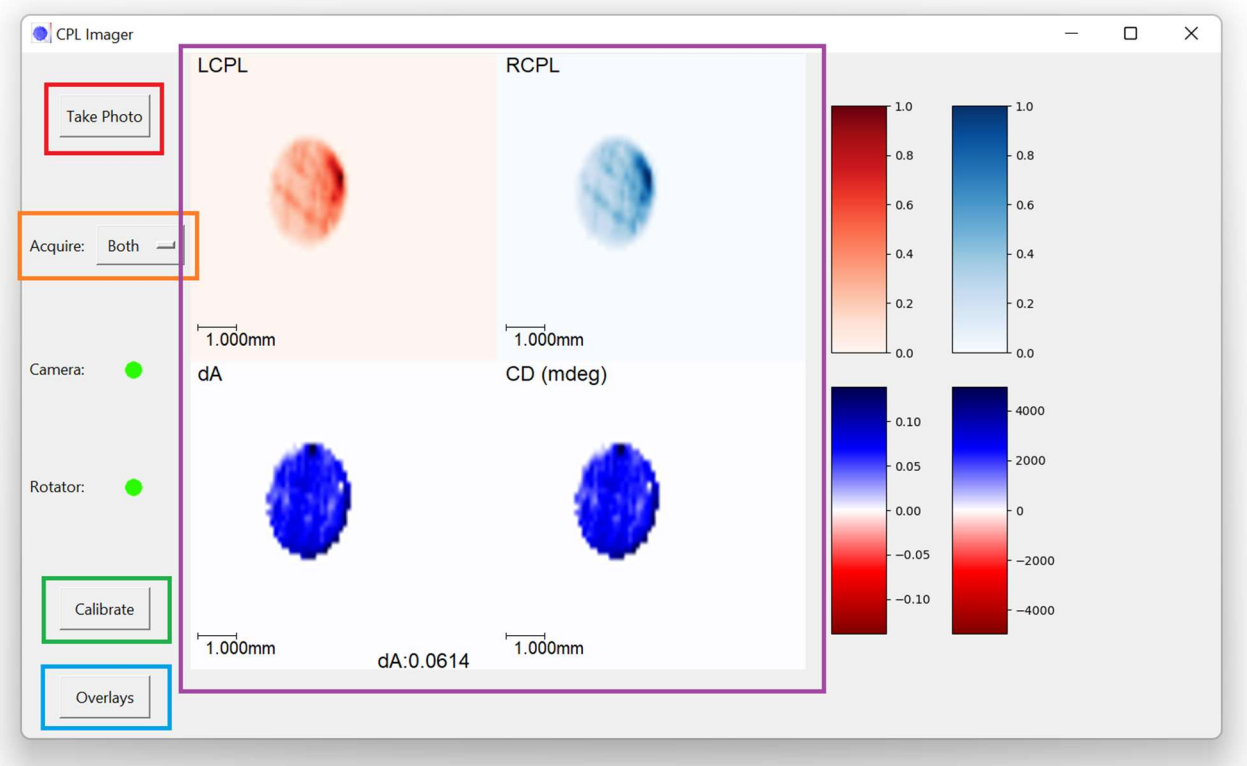
9) Place the piezorotator board ribbon cable into its space on the bottom of the back body, then plug the USBs from the camera and piezorotator board into either your computer or a Raspberry Pi.





There is an alternate setup with 2 cameras and a polarised beamsplitter (rather than the piezorotator), but this is more expensive and has difficulty in aligning the 2 cameras. However, it has no associated rotator delay so can run with a higher framerate.

# 4 Software:




These following sections refer to the control software of the CPL-Imager. Certain sections are colour coded to refer to sections of the above GUI image. A handy quickstart guide is in the `docs` folder under the filename `quickstart.png`, which may be sufficient to understand the software.

## 4.1 Installation:

As with assembly, detailed installation instructions (including automated scripts) can be found in the `install` folder, but to sum up:

1. Download the correct Thorlabs SDK from their website (link!) for your system. If using a Raspberry Pi for the setup, email the Thorlabs tech support team at (email!) and ask for their experimental ARM SDK, where the .so files are compiled for the right instruction set.
2. Run whatever `THORLABS_SDK_INSTALLATION_INSTRUCTIONS.txt` says to do for your system, and install the Python library in the "Python Toolkit" folder via `pip` or `conda` or whatever environment manager you use

3. Install the required Python libraries (`numpy`, `matplotlib`, `Pillow`, `pyserial`, also `skimage` if you want to try the 0 camera demo)
4. Run `CPL_Imager.py` either via the CLI or in an IDE of your choice

After that, you're all set!

## 4.2 Panel Layouts:

The software has 4 panels which show different data. The top two panels show the last readings of LCPL and RCPL intensity taken from the camera. The bottom two show dA and CD, two measures of the differential absorption of LCPL vs RCPL common in the literature. All 4 are colourmapped based on the colourmaps in `config.json`; the values of these colours can be read off the colourbars on the right hand side of the screen, where the colour bar position corresponds to the same data pane (i.e top left colourbar is LCPL).

## 4.3 Taking Photos:

Pressing the "Take Photo" button will take a photo of whatever is currently on the GUI screen and save it to a timestamped directory in the `photos` folder. This will contain a bitmap of the image, a `.txt`, `.csv` and `.npy` of the raw LCPL and RCPL data and a file describing the image metadata like any corrections applied or the pixels per mm of the images. LCPL data will be the left image and RCPL data will be on the right of the image (the same holds true for the data files).

## 4.4 Acquisition modes:

There are 4 possible acquisition modes of the software, available under the "Acquire" dropdown menu:

**- Both:** the rotator will rotate every 0.2 seconds, taking images of alternating handedness and displaying them on the screen. dA and CD will be calculated automatically and displayed as well. This mode has an associated 0.2s delay, and the rotator will burn out if run continuously for too a time, so be careful.

**- LCPL / RCPL:** the rotator will move to either the horizontal or vertical position, and the camera will capture images of a single handedness with a high framerate. As there is no rotator delay, the live view is much more responsive, so these modes are good for calibrating the system before taking images.

**- Pause:** both the rotator and camera will pause and the picture will be static. Useful if you have an image you want to examine or take a photo of.

## 4.5 Calibration:

Pressing the "Calibrate" button will open a sub-menu with a variety of useful calibration options:

- **RPS correction**: remove any samples in the beam path and set up your source to be a Reference Polarization State i.e a state with a 50:50 mix of LCPL and RCPL and where each intensity reading is just barely saturated. Then enter '1' in both forms and press submit then a correction mask will be generated which adjusts any spots that aren't of value 1. This correction then persists, so will apply when imaging a sample. This is intended to correct errors in the setup like optical aberrations.

- **Spatial calibration:** click two points on the screen separated by some reference distance, then enter the reference distance in mm into the popup and press 'Finish'. Then the spatial based overlays like ticks, axes and gridlines will automatically adjust to the new scale when toggled on in the 'Overlays' menu.

- **Reset ROI:** reset camera ROI back to a wide view (1024x1024). Useful for calibrating the ROI later.

- **Calibrate ROI:** click a point on the screen to act as the top-left corner of the ROI, then a rectangle will appear at the current cursor position showing a bounding box. Click a second point to define the other corner and finish the calibration.

- **Intensity calibration:** a popup will appear telling you the percentage of non-zero pixels in the LCPL or RCPL data that are 1 or above (i.e are saturated). When taking measurements it is preferable to have 0% saturation with as high an intensity as possible; you want to be just below the threshold of saturation.

- **Set threshold:** set a threshold intensity value below which any readings will be set to 0. Useful if there's weak stray light. Would recommend using this for calibration or visualising more so than taking data as any masked data won't be recorded, causing you to miss possible real signals. Setting a threshold will also cause the colourbars on the right to be reloaded.

## 4.6 Overlays:

These overlays appear over the data panes when toggled on in the menu (and are not saved to the photos):

- **Change cmaps:** there is a text field for each data pane - enter the name of a valid matplotlib colourmap for each and press finish, then the colourmaps will be updated. I recommend diverging colourmaps for dA and CD panes and converging colourmaps for LCPL and RCPL.

- **Intensity:** when toggled on, text will appear at the bottom of whichever data pane the cursor is on displaying the value of that pane at that point. This means the mouse can be used for quick data reading.

- **Label:** display text which explains which pane is which.

- **Tick:** a tick bar will appear on each pane showing the scale of the image. The size of the tick bar will be determined by your spatially calibrated pixels-per-mm value.

- **Axes:** axes will appear on each pane, again with a scale given by the pixel-per-mm value

- **Grid:** a grid will appear on each pane at the same positions as the axes but across the entire pane.

## 4.7 Source Code:

Source code is of course contained in the repo - modify to suit your needs. If you think it's a useful addition, please send a merge request. I have broadly tried to follow a Model-View-Controller framework where possible whilst also respecting the threaded nature of the Thorlabs camera code, but this has caused some rough edges.

The programme structure is as follows:

- `CPL_Imager.py`: entry point of the program, instantiates the camera, Thorlabs SDK, GUI and live view objects.

- `colourmapper.py`: handles the colourmapping for the data panes and the colourbars on the left hand side. Colourmaps based on matplotlib colourmaps and adjusts based on the max and min values on the screen. Also contains the definitions for delta absorbance and circular dichroism, which probably isn't the best place to hold the central equations but que sera.

- `cameras.py`: classes for the Thorlabs camera, handles the main loop of taking photos, sending the rotate command and dumping photos onto the correct queue.

- `rotator.py`: sends hex commands via pyserial to the rotator.

- `GUI.py`: handles the GUI – mostly buttons etc and logic around that, but also contains a live image display thread that grabs pairs of images from the separate queues, combines them, colourmaps them and places them onto the GUI.

# 5 Troubleshooting:

This section contains some common problems and suggested solutions:

- **COM/USB port can't be found for the rotator**. Usually happens when program crashes during runtime and the port wasn't closed properly. Unplug and plug in the rotator board USB and try again.
- **I can't see stuff on the screen!** Check the camera ROI, shutter etc.
- **I pressed 'left' on the acquisition dropdown but nothing changed!** Probably the rotator getting strange – try changing it to 'Right' then back to 'Left'.
- **The Python Thorlabs library isn't working! It can't find the drivers!** Try following the driver/.so file troubleshooting in the SDK. If still not working, check the `windows_config.py` file path is right (on Windows) or that you've run `ldconfig` properly on Linux.
- **Permission error on /dev/ttyUSB{N} !** Your current user account doesn't have USB permission. Run `sudo chmod 777 /dev/ttyUSB{N}` where {N} is some integer (i.e `/dev/ttyUSB0`)
- **I'm using a Raspberry Pi and I've followed all the driver instructions properly but the Python library still won't work!** Make sure you're using the ARM compiled version of the SDK – this isn't available online, you have to email the Thorlabs support team for it.