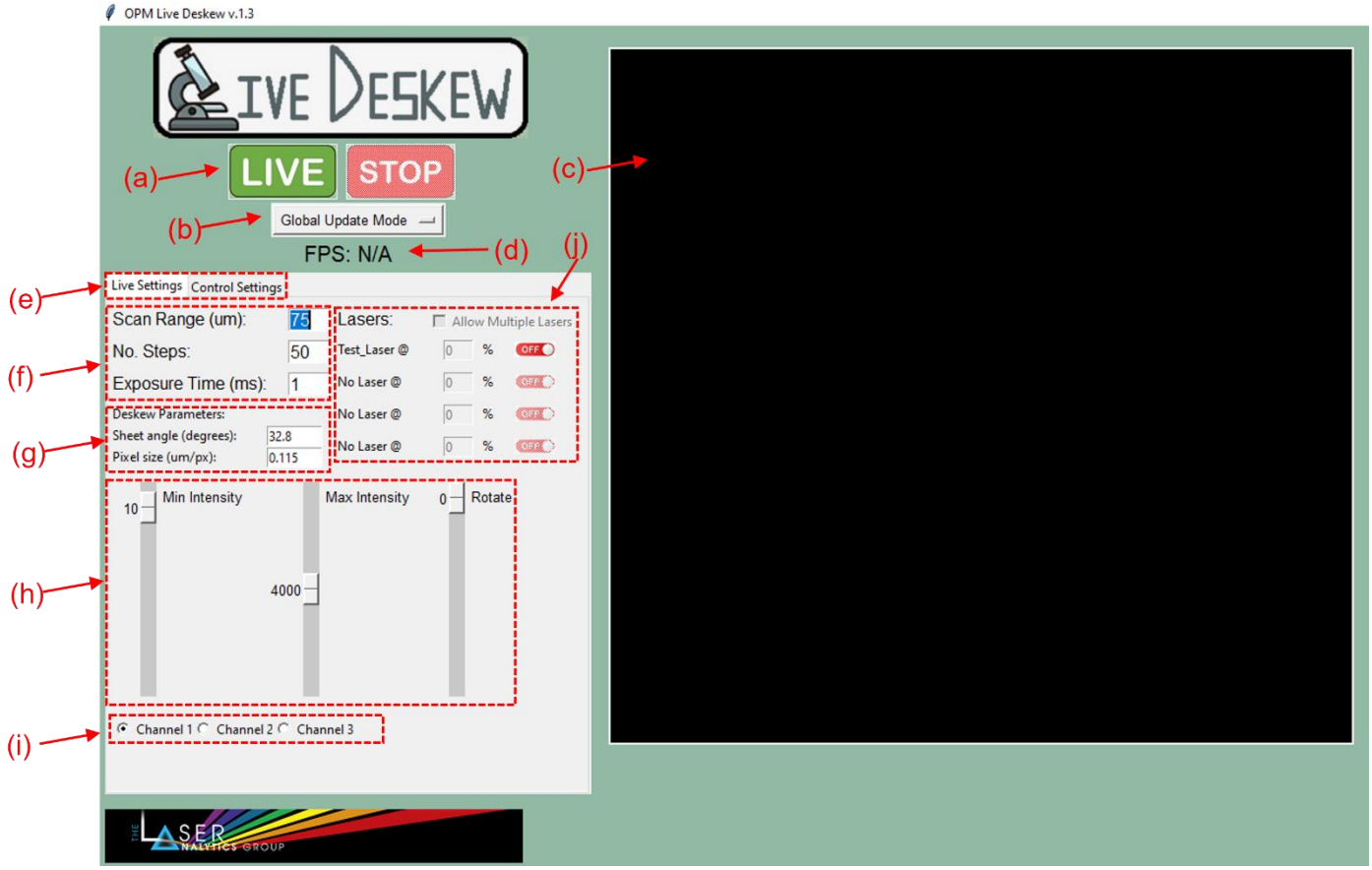
Live Deskewing Software User Manual

GUI Overview:



a) **Live/Stop Buttons:** These buttons start and stop the live deskewing. 'Live' starts the software and 'stop' stops is. Note that the scan parameters are disabled whilst the software is running.

b) **Update Mode Selection:** Here the user can choose between the global and rolling update modes. Global mode updates the output image after each complete stack and is suitable for experiments with low exposure times. Rolling mode updates the output images after each camera frame and is suitable for long exposure time experiments. (We would consider a 'long' exposure to be > ~20ms)

c) **Image display:** This is where the output deskewed image will be displayed. Note that the size of this window will vary depending on the camera region of interest and the sample scan range used.

d) **Frame rate indicator:** Here the output frame rate of the image display will be plotted. This is how quickly the output image in the GUI is updated thus will be a lot higher in rolling mode.

e) **Tab selector:** The 'Live settings' tab contains all the controls needed to operate the live deskewing software and changing the imaging characteristics. The 'Control settings' tab is where hardware can be initialised.

f) **Scan parameters:** Here the scan range, number of steps per volume and camera exposure time can be set. These can only be varied whilst the deskewing is not running.

g) **Deskewing parameters:** Here the sheet angle and pixel size can be set. These must be accurately set for the software to correctly deskew the raw images. Sheet angle is the angle between the sheet and the axis of sample scanning. Pixel size is the size of each camera pixel in the sample.

h) **Live image control:** Here the contrast of the image can be adjusted whilst the software is running by changing the minimum and maximum pixel intensity values. The rotate slider dynamically adjusts the skew factor to give a rotated perspective of the sample. 0 corresponds to a top-down view and 90 gives a side on view.

i) **Channel selection (Only for Optosplit):** When a triple colour optosplit is used the live image controls (h) will control the image of the channel selected by these radio buttons. E.g., if channel 1 is selected the controls will change the displayed output of channel 1.

j) **Laser control (Only when lasers are initialised):** Here the lasers can be turned on and off. Power is controlled by changing the value in the edit box and then they can be turned on and off using the buttons to the right. Unless the 'Allow Multiple Lasers' button is checked (which can only happen if an optosplit is installed) then turning a laser on will turn all other lasers off.

1. Running the software:

1. Open the camera in micromanager and select the ROI where the sheet is formed. NB: Smaller ROIs will have a higher final framerate.
2. Run the script Live_Deskew_GUI.py
3. Enter the correct scanning and deskewing parameters. (If using your own hardware control make sure the scan range and No. steps match the values set in your code)
4. Press the start button to begin displaying a live deskewed output

2. Hardware control modes:

Note that when the software is first opened no hardware is initialised and the software will only control the camera in an internal triggering mode.

2.1 Your own software:

The live deskewing software can be used in conjunction with your own hardware control software. In this case, your software will have to handle synchronisation of sample scanning with the camera exposures. The live deskewing software will continually take images from the camera and then process and plot them. Your software will have to move to the next sample position (either move the stage or move the sheet) after each camera image. To do this we recommend setting your system to be externally triggered to move to the next scan position by a TTL output line from the camera. We use an exposure output line on the camera which is high whilst the camera pixels are exposing and low otherwise. The falling edge, when no pixels are exposing, is used as the trigger for our NI DAQ card to then move to the next galvo voltage. We have this set up in LabView software

which is running whilst the live deskewing software is running. We use a Photometrics Kinetix 22 so you will need to check the operating manual of the camera you use to find the appropriate triggering line.

Note that your software will have to be initialised with the same parameters as the live deskewing GUI. This means using the same scan range and number of steps, set in box (f), in both the live deskewing software and your hardware control. E.g., if the live deskewing software has Scan range = 75um and No. steps = 50, your software will have to scan a range of 75um in 50 steps.

2.2 Using our software to control hardware:

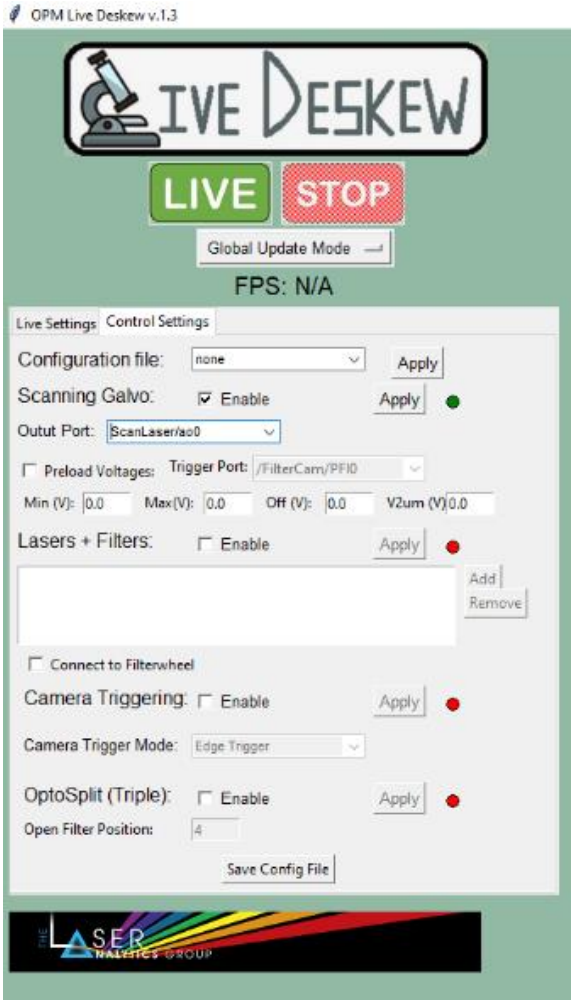
Our software can handle the lasers and hardware synchronisation, or any subset of the hardware you wish to use.

2.2.1 Scanning Galvo control: (Galvo Scanning OPM):

The movement of the scan galvo can be controlled by an analogue output of a NIDAQ card. The voltage can either be written before each camera frame or can be preloaded into the DAQ card and externally triggered by an output line from the camera. The latter will increase the imaging rate.

1. To initialise the galvo first check the enable button. The status indicator (right of the apply button) will now change orange to indicate the settings have been change but not applied.
2. Select the output NIDAQ card port from the drop-down list. This is the voltage port that controls the galvo position. The software will automatically scan the computer and find the available ports.
3. If you want to externally trigger the voltages, select the ‘Preload Voltages’ check box and choose the NIDAQ card line which will receive the trigger from the camera. If this is not selected the code will default to writing the voltages before each camera exposure.
4. Input the minimum allowed galvo voltage ‘Min (V)’, maximum allowed galvo voltage ‘Max (V)’, galvo offset voltage ‘Off (V)’ (this is the voltage the software will scan around) and the volts to um conversion ‘V2um’.
5. Click the apply button. The status indicator should now be green showing that the galvo is initialised and ready to use.


OPM Live Deskew v.1.3
LIVE DESKEW
LIVE
STOP
Global Update Mode
FPS: N/A
Live Settings
Control Settings
Configuration file:
none
Apply
Scanning Galvo:
Enable
Apply
Outut Port:
ScanLaser/ao0
ScanLaser/ao0
ScanLaser/ao1
Preload V
erCam/PFI0
Min (V):
0.0
Max(V):
0.0
Off (V):
0.0
V2um (V)
0.0
Lasers + Filters:
Enable
Apply
Add
Remove
Connect to Filterwheel
Camera Triggering:
Enable
Apply
Camera Trigger Mode:
Edge Trigger
OptoSplit (Triple):
Enable
Apply
Open Filter Position:
4
Save Config File
LASER
ANALYTICS GROUP


OPM Live Deskew v.1.3
LIVE DESKEW
LIVE
STOP
Global Update Mode
FPS: N/A
Live Settings
Control Settings
Configuration file:
none
Apply
Scanning Galvo:
Enable
Apply
Outut Port:
ScanLaser/ao0
Preload Voltages:
Trigger Port:
/FilterCam/PFI0
Min (V):
0.0
Max(V):
0.0
Off (V):
0.0
V2um (V)
0.0
Lasers + Filters:
Enable
Apply
Add
Remove
Connect to Filterwheel
Camera Triggering:
Enable
Apply
Camera Trigger Mode:
Edge Trigger
OptoSplit (Triple):
Enable
Apply
Open Filter Position:
4
Save Config File
LASER
ANALYTICS GROUP

2.2.2 Writing software control for your own hardware scanning:

If you want to use our software to control the scanning, however, are not using galvo mirror scanning controlled by an NI DAQ card analogue voltage we have written the software in such a way that it can be easily changed to incorporate different scanning hardware. All that needs to be changed is the file utils.sampleScan.py.

Here we will describe the process of how to add functionality to control scanning using alternative hardware. In this example we show the procedure for an ASI motorised translation stage, but the process can be easily extended to other technologies:

For the code to run properly the class named sampleScan **must** have the functions set_scanrange(), set_steps(), createWaveform(), connect(), initPos(), nextPos() and exit(). Aswell as the standard __init__ function. To help with writing the new file we provide a template in the Github file Live-Deskewing/utils/AltHardware/Template/sampleScan.py

- The __init__ function:

  This function is fairly self-explanatory, here some basic variables that will be used in later functions are initialised. In the case of the ASI stage we define the serial port name and the baudrate. Also we define the settle time needed between each stage position. This settle time is not only time needed for the stage to move but a delay to prevent serial commands being sent to the stage too quickly which we have found can cause issues.

  Example Code:

```
def __init__(self):
    self.asi_stage_serialport = 'COM3'
    self.asi_stage_baudrate = 9600
    self.currentPosNo = 0
    self.stepsize = 0 # 10 = 1um for asi commands
    self.scanrange = 0
    self.steps = 0
    self.settleTime = 200 #Settle time for stage in ms
```

- The set_scanrange() function:

  This function is used to set the scan range that has been chosen. This value comes from the user set value given in block (f) in the GUI.

  Example Code:

```
def set_scanrange(self,scanrange):
    self.scanrange = scanrange # in um
```

- The set_steps () function:

  This function sets the number of steps per volume to be used during operation. This value is also set by the user in block (f) in the GUI. Note that in the software set_scanrange() is called before set_steps(). Thus in the set_steps() function we can also calculate the size of each scan step between each image.

Example Code:

```
def set_steps(self,steps):
    self.steps = steps
    self.stepsize = (self.scanrange/steps)*10
```

- The createWaveform() function:

  This function is used to create the range of voltage values for the scan range. This function is used for devices that are controlled using an analogue voltage. In the case of a stage where we move the stage by a set amount each time this function is not needed thus, we set it to do nothing.

  Example code:

```
def createWaveform(self):
    print('Doing nothing here')
```

- The connect() function:

  As the name suggests this function is used to connect to the scanning hardware. Here we open a serial connection to the stage controller.

  Example code:

```
def connect(self):
    print('Connecting to ASI Stage Control')
    self.ASI_StageSer = serial.Serial(self.asi_stage_serialport,self.asi_stage_baudrate)
    print('stage baudrate is ', self.asi_stage_baudrate)
    self.ASI_StageSer.bytesize = serial.EIGHTBITS # bits per byte
    self.ASI_StageSer.parity = serial.PARITY_NONE
    self.ASI_StageSer.stopbits = serial.STOPBITS_ONE
    self.ASI_StageSer.timeout = 5
    self.ASI_StageSer.xonxoff = False #disable software flow conrol
    self.ASI_StageSer.rtscts = False #disable hardware (RTS/CTS) flow control
    self.ASI_StageSer.dsrdtr =  False #disable hardware (DSR/DTR) flow control
    self.ASI_StageSer.writeTimeout = 0 #timeout for write

    if not self.ASI_StageSer.isOpen():
        try:
            self.ASI_StageSer.open()
        except Exception as e:
            print('Exception: Opening serial port: '+ str(e))
```

- The initPos() function:

  This function is used to move the scanning hardware to the first position. Note that the software calls the function to move to the next scan position once before the first image is taken. Thus, here the scan hardware should actually be moved to one position before the

first position. In this example we create a variable that keeps track of what scan position the stage is currently in which we set to be -1.

Example code:

```
def initPos(self):
    self.pos = -1
```

- The nextPos() function:

  This function is used to move to the next scan position after each camera frame. The deskewing function is expecting the scan direction to change after each volume. Thus, in this function the scan direction needs to change at the end of each volume. Here we update a variable called pos after each movement and when this reaches the number of steps per stack we invert the direction of scanning. Note that if the initial scan direction is opposite to what the deskewing is expecting the final result will look wrong. Thus, if the result looks incorrect this is a good first place to start.

  Example code:

```
def nextPos(self):
    ASI_ZCommand = 'movrel x=' + str(self.stepsize) +'\r\n'
    self.ASI_StageSer.write(str.encode(ASI_ZCommand))
    time.sleep(self.settleTime/1000)
    self.pos += 1
    if self.pos == self.steps:
        self.pos = 0
        self.stepsize = -1 * self.stepsize
```

- The exit() function:

  This function closes the communication with the scanning hardware.

  Example code:

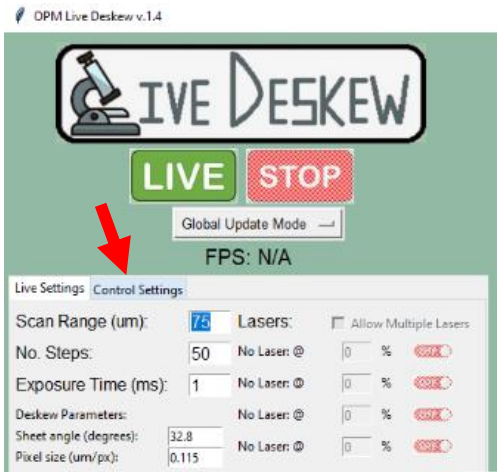
```
def exit(self):
    try:
        self.ASI_StageSer.close()
    except Exception as e:
        print('Exception: Opening serial port: '+ str(e))
```

Once the new code has been written, this file (which must be called sampleScan.py) should be put in the utils folder so it can be read by the OPMLiveDeskewing.py code. Once your new code is in the right folder start the live deskewing GUI as normal. Now once the GUI has started you will need to set up scanning control in the software and tell it that you are using alternative hardware to a NIDAQ card analogue voltage.

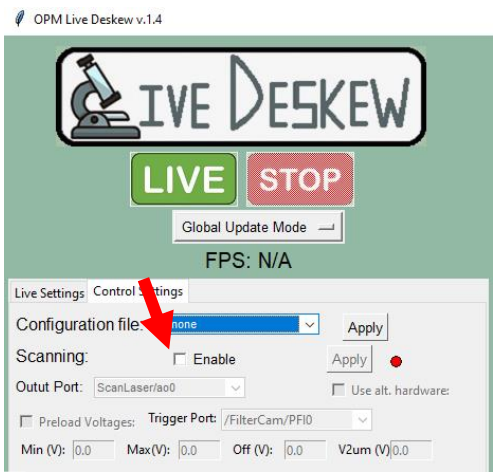
To do this:

a) Navigate to the 'Control Settings' tab
b) Check 'Enable'
c) Check 'Use alt. hardware'
d) Click 'Apply'
e) The indicator switch should now show green to show. Your new hardware should now be ready to use and you can start live deskewing.

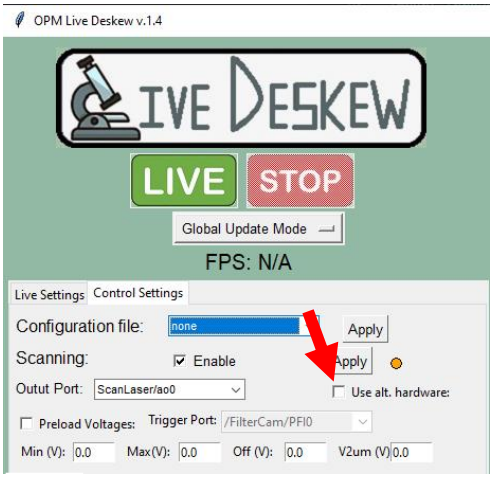
(a)



(b)



(c)



(d)

(e)

2.2.3 Running the software without installing the nidaqmx library:

The software requires the use of the python nidaqmx library. However, if you do not want to install this library it is possible to run the code without using this library by making a few small edits to the file Live_Deskewing_GUI.py Note that these are instructions are correct as of 25/01/2023. Any future updates may change the corresponding line numbers.

1. Comment out line 15: import nidaqmx
2. Comment out line 166: self.local_system = nidaqmx.system.System.local()
3. Comment out lines 566 - 579: def find_daqaolines(self): **…** self.daqdilines.append(line)
4. Comment out line 196: self.find_daqaolines()
5. Comment out line 200: self.find_daqdolines()
6. Comment out line 202: self.find_daqdilines()
7. Change line 195 from: self.daqaolines = [] **to** self.daqaolines = ['']
8. Change line 199 from: self.daqdolines = [] **to** self.daqdolines = ['']
9. Change line 191 from: self.daqdilines = [] **to** self.daqdilines = ['']

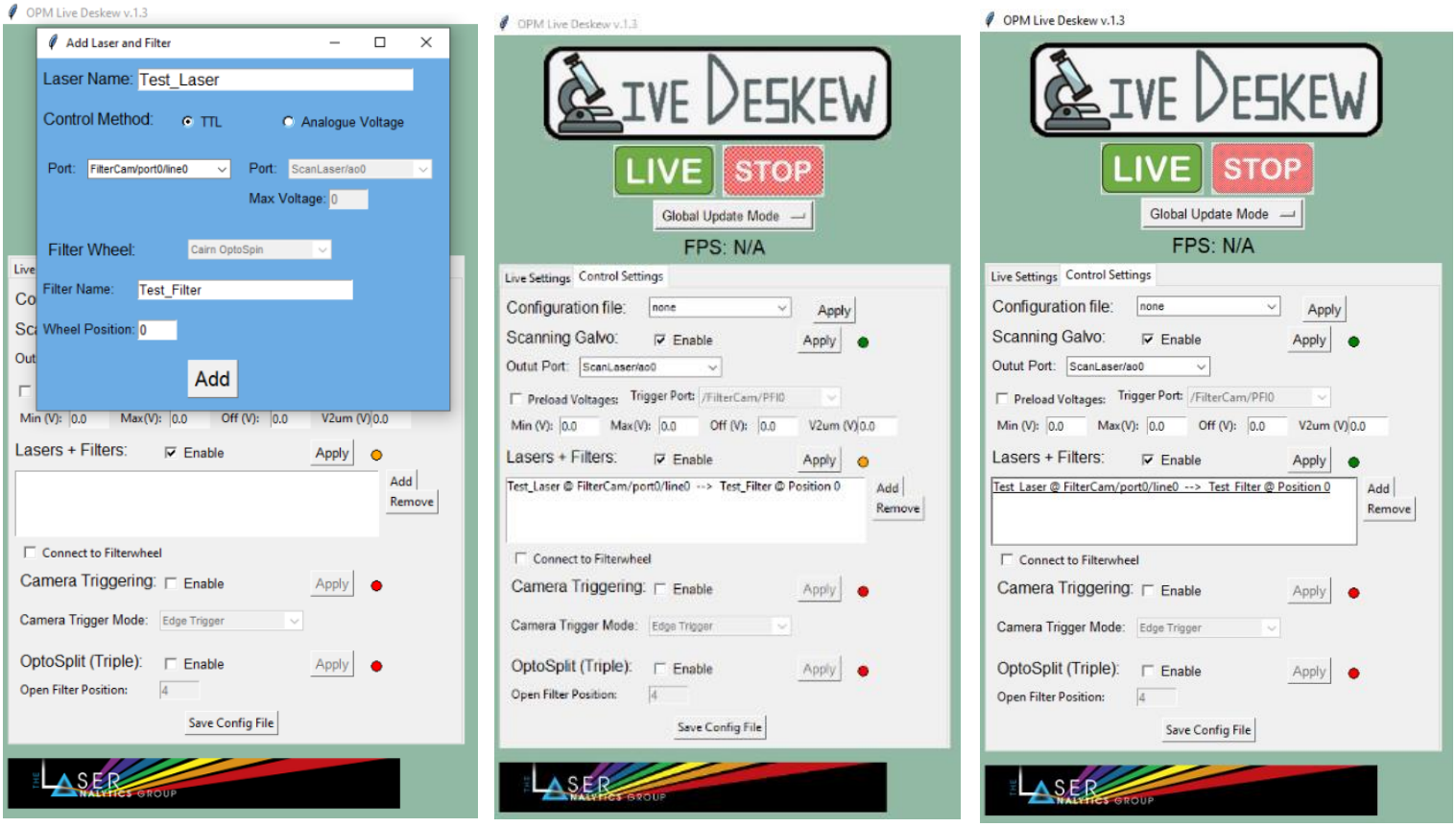
The software will now run without nidaqmx.

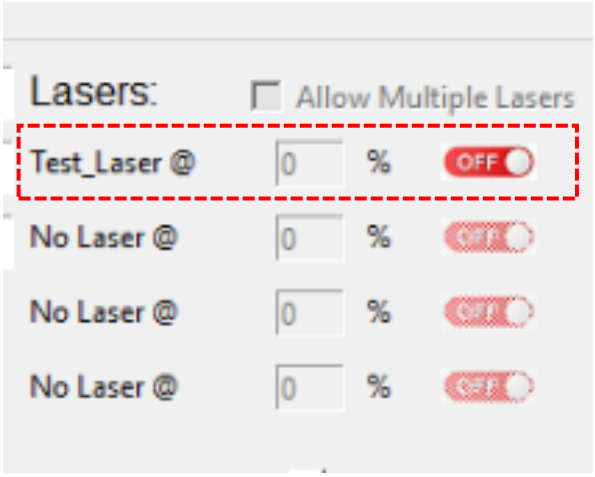
2.2.4 Laser Control:

Lasers can be controlled from within the GUI via digital (on or off) or analogue (voltage) signals from NIDAQ cards.

1. To initialise the lasers first check the enable button. The status indicator (right of the apply button) will now change orange to indicate the settings have been change but not applied.
2. To add a laser, click the 'add' button and a blue window will pop up to allow a laser to be added.
3. Fill in the form. 'TTL' means that the lasers will be turned on when the output is high and off when the output is low (we use this for a shutter in front of our 561nm laser which doesn't have any power modulation). 'Analogue voltage' is where the laser is controlled by an analogue voltage. The max voltage edit box is the voltage at which the laser is at full power. Each laser will need a corresponding filter wheel position. Currently only the Cairn OptoSpin is supported but it should be fairly easy to extend the code to support other filter wheels.
4. Once the form is complete select add and the laser will be added.

5. Once all the lasers you want are added then select apply and the lasers will be initialised in the software. At this point the lasers names in the Live settings tab will be updated.





2.2.5 Saving configuration files:

In order to speed up initialising all the hardware that you want to use you can save configuration files which can then be used to rapidly initialise all the hardware you want at once.

1. Install all the hardware that you want to use.
2. Click on the ‘Save Config File’ button
3. Insert the name that you want to use and click ok one happy.
4. The configuration file dropdown box will contain all the .txt files in the directory where the *Live_Deskew_GUI.py* file is run.
5. Select the appropriate file from the dropdown box and select apply to initialise all the hardware.