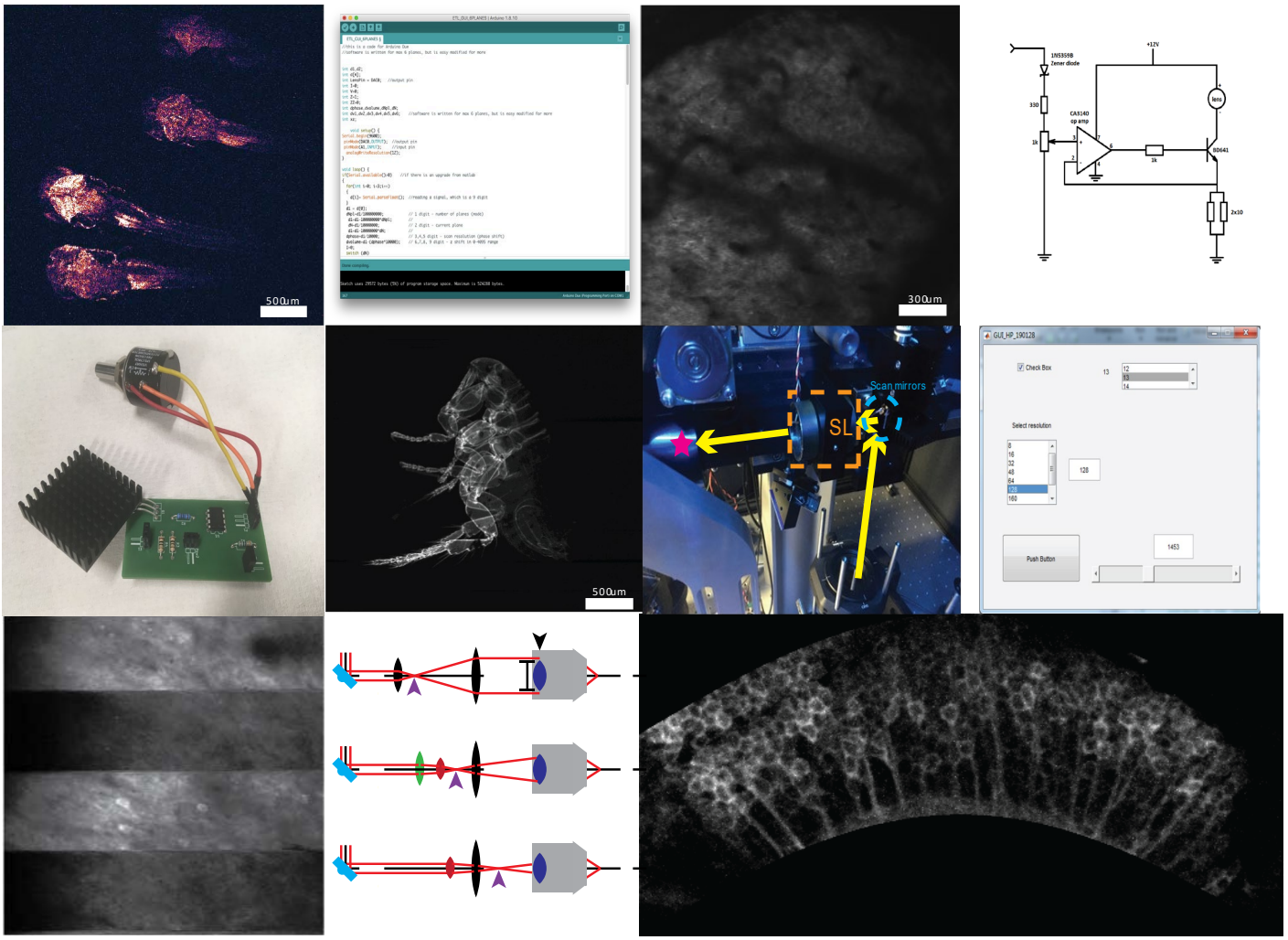
Manual for divergent (non-telecentric) excitation modification of 2-photon microscopy

Filip Janiak[1] and Tom Baden[1,2]

1: Sussex Neuroscience, School of Life Sciences, University of Sussex, UK;
2: Institute of Ophthalmic Research, University of Tuebingen, Germany;



A note on safety

Remember to always wear safety googles when working with lasers! Take off all hand-worn jewellery and watches as these might reflect the beam into your or someone else's eyes. Always work at the lowest possible laser power. And never, even with the safety glasses on, look straight into the laser beam!

What will be useful?

Tape measure, fixed fluorescence sample (i.e. a test-slide), laser viewing card

Definitions and helpful simplifications

Many of the optical elements in 2p microscopes, like the scan lens (SL), tube lens (TL) and objective (objective lens) tend to consist of multiple stacked lenses with different curvatures and thickness. For simplicity, we treat each of these as a single ideal lens.

In detail: There is a large set of parameters describing optical properties of such lenses, however for us important are only the following: principal planes (P, P'), focal planes (F, F'), surface vertices (V, V'), and three focal lengths: effective focal length (EFL), back focal length (BFL) and front focal length (FFL). For details on any of these, see also the following Wiki articles:

https://en.wikipedia.org/wiki/Cardinal_point_(optics)
https://en.wikipedia.org/wiki/Focal_length

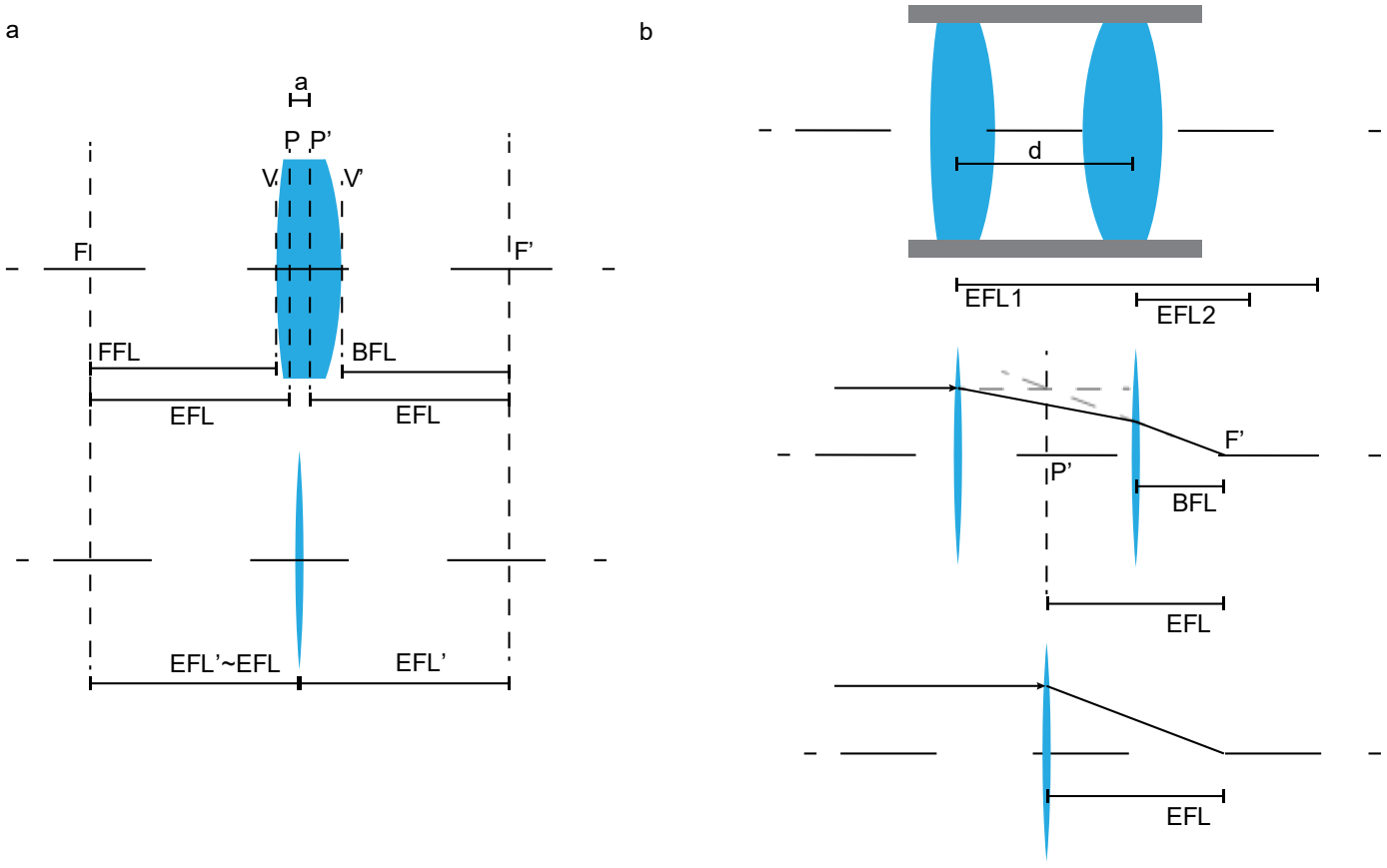


Fig 1. Thin lens approximation of a thick lens (a). Single thin lens approximation of two thick lenses (b).

In optical models, such constituent lenses can be replaced by a single, thin, ideal lens, and for simplicity this is what we are doing in all calculations and diagrams. This ideal lens has a negligible thickness, so principal points are located exactly in a centre of the lens, and effective, back and focal length are the same, and equal to EFL of thick lenses corrected by half of the thickness (EFL' on fig 1). Typically, the thickness can be disregared if EFL>> thickness.

$$EFL' = EFL + \frac{a}{2} \quad if\ (EFL \gg a) \quad EFL' \cong EFL$$

To calculate effective focal length of two thin lenses, we can use:

$$\frac{1}{EFL} = \frac{1}{EFL_1} + \frac{1}{EFL_2} - \frac{d}{(EFL_1)(EFL_2)}$$

Or just simply:

$$\frac{1}{f} = \frac{1}{f_1} + \frac{1}{f_2} - \frac{d}{f_1 f_2}$$

See more on https://physics.stackexchange.com/questions/247617/how-is-focal-length-defined-for-a-two-lens-system-separated-by-a-distance-d

This ideal lens is located in the back principal planes of complex lenses (which is usually in the centre of a given tube).

Initial steps: Obtaining all the key numbers of your specific setup.

This section is intended to help identify all key elements in your laser path between the scan mirrors and the sample plane, and to obtain concrete values for their relative distances and focal lengths etc..

1. Identify scan lens (SL). It should be the first lens ensemble after the scanning mirrors (from point of view of the laser source). Identify Tube Lens (TL), it should be the next one after SL. Measure the distance between them (D_2).

2. If focal lengths and dimensions are unknown, carefully take out both lenses. Use gloves to avoid leaving smear-marks, and do not scratch them. Measure their focal distance (F_{SL},F_{TL}), and their dimensions ($Ø_{SL}$, $Ø_{TL}$), for example as demonstrated here: https://www.youtube.com/watch?v=joQw3_rM_jI

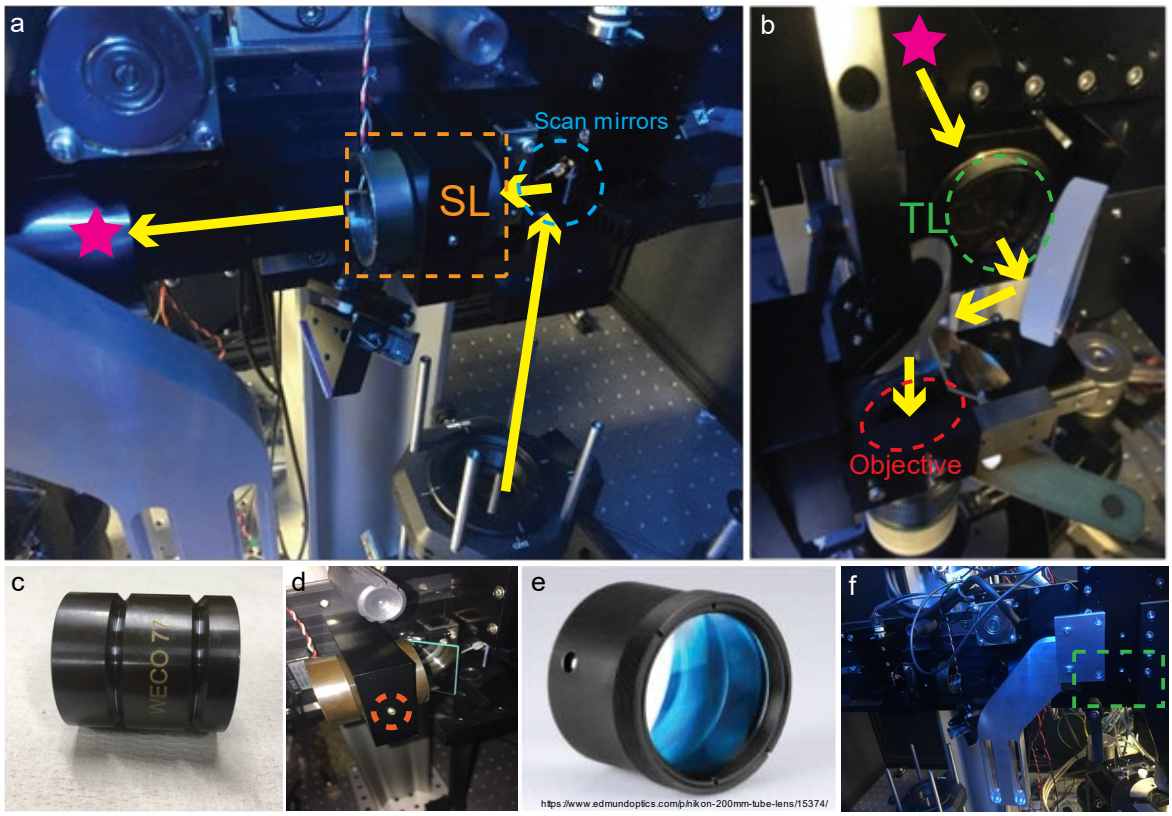


Fig 2. Sutter MOM setup with DL standard configuration. Location of SL (a), TL and objective (b). SL outlook (c) and mounting (d). TL outlook (e) and position from the side (f).

If you use a Sutter MOM setup, the SL model is VISIR 1534SPR136 from Leica (Fig. 2a), which is mounted in a WECO 77 tube (Fig. 2c). It has a focal length of 50 mm, and it is

located 56.6 mm from second scan mirror. The thickness of the lens (which is made up of two biconvex lenses) is 43.1 mm. The external tube diameter is 38.2 mm, and the internal aperture diameter is 28.9 mm. It is easy to unscrew and take out (Fig. 2d). However, in the Sutter MOM the TL (MXA22018, Nikon) is less readily accessible (Figs. 2b,f). It has a 200 mm focal length (Fig. 2e) and is positioned at a distance of 250 mm from the SL. The length of the TL tube is 29 mm, and it has a 36 mm aperture.

3. For your objective, check the magnification, nominal working distance, and size of the back aperture ($Ø_{obj}$). Measure the total distance between the objective's back aperture and the SL.

Note: Microscope objectives are designed to correct for chromatic aberration at a specific distance D_3 between TL and its back aperture O_{BA}. For Zeiss objectives, D_3 = 95 mm. (Philbert S. Tsai and David Kleinfeld, 2009).

In our setup, we use a Zeiss Objective W "Plan-Apochromat" 20x/1.0, 1.8 mm working distance.

In a standard diffraction limited (DL) configuration, the distance between SL and TL is equal to the sum of their focal distances $D_2=F_{SL}+F_{TL}$. D_3 is specified by the objective as discussed above.

4. Based on these numbers, calculate the distance D_1 between the second scan mirror pivot point and SL that minimizes motion of laser beam (intensity fluctuation) on objective back aperture:

$$D_1 = \frac{F_{SL}^{\ 2}}{F_{TL}} + F_{SL} - D_3\left(\frac{F_{SL}}{F_{TL}}\right)^2$$

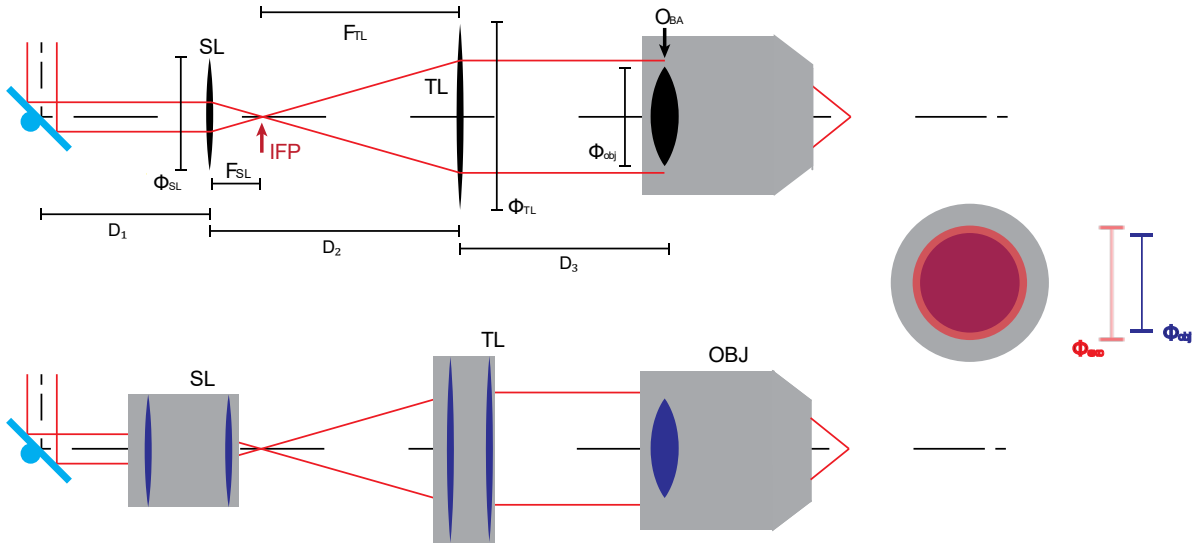


Figure 3. DL configuration. Ideal lenses approximation (top), objective back aperture overfilling (left) and typical configuration with stacked lenses (down).

5. Check objective's back aperture (O_{BA}) filling factor (Øexc/Øobj). To do so, turn on a regular scan mode at "high zoom" (i.e. with minimal movement of scan mirrors) without the objective in place. Use a laser viewing card in the position where the objective's back aperture would be to assess the setup's fill factor (i.e. the size of the laser spot relative to the objective's back aperture) (Fig. 3, 4). Alternatively, if your laser supports an alignment mode at a visible wavelength (typically low 700s of nm) you can also do this procedure with a piece of paper in place of the laser viewing card.

In a DL configuration, overfilling O_{BA} by the excitation laser beam minimizes the lateral and axial size of the excitation spot (point spread function, *psf*), but also decreases effective excitation power delivered by the objective (Helmchen & Denk, 2005; Philbert S. Tsai and David Kleinfeld, 2009).

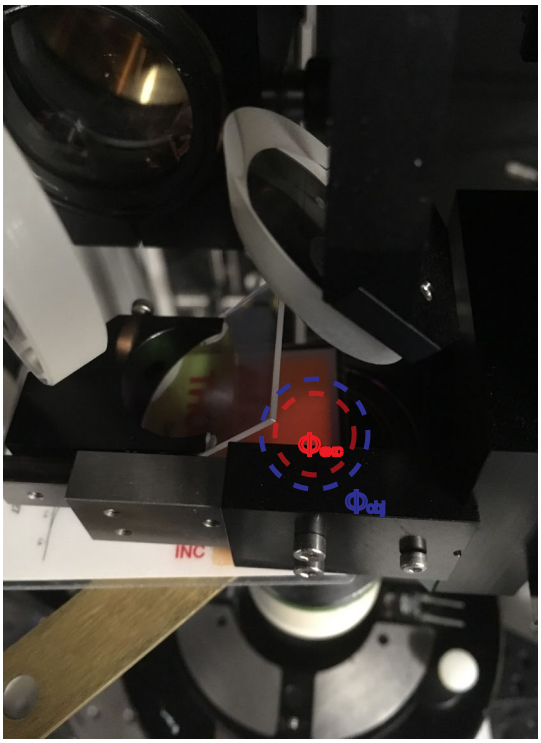


Fig 4. Objective back aperture filling factor showed on laser viewing card.

This manual describes how to change a setup from a DL configuration. Therefore, based on your data, check and calculate if your setup operates on DL settings.

If it is, go ahead to the next section. If it is not, try to understand why. There are two main possibilities. First, unintentional misalignment during a construction/mainlining or a using of the setup, which can be easily fixed to come back to DL configuration or apply a modification. Second, intentional changes from DL configuration. In this case you need to check why your setup is designed in this specific manner (e.g. if there are specific advantages/limitations). In this case, application of our modifications could be more complicated, but may still be possible.

We present two modifications to obtain a divergent excitation 2-photon setup leading to an expanded field of view (FOV) and excitation spot (point spread function, *psf*).

Throughout, we will be focussing on two parameters: Intermediary focal point (IFP) and the objective's back-filling factor (Fig. 5):

Intermediary focal point (IFP) position Objective back aperture filling factor

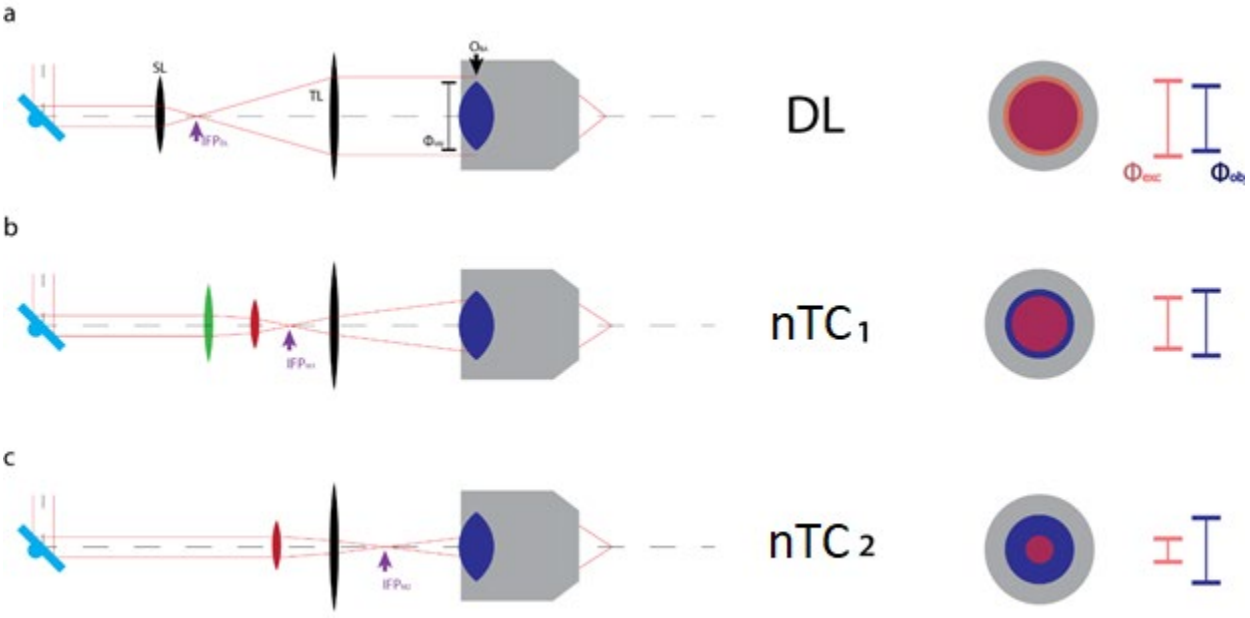


Fig. 5. IPF positions (left) and objective back aperture filling factor (right) for 3 configurations DL(a), nTC$_1$(b) and nTC $_2$(c).

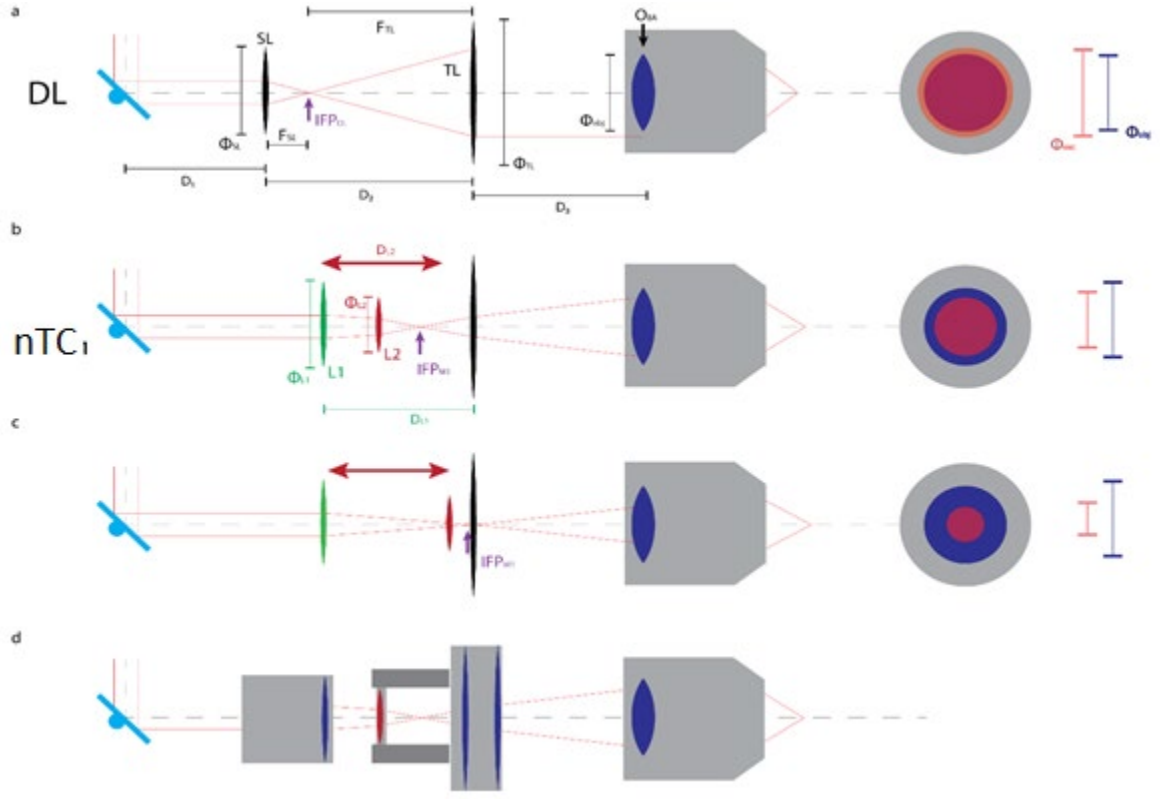
nTC_1 configuration

1. For “non-telecentric modification 1” (nTC_1), the goal is to move the IFP closer to the TL, yet without passing it, while simultaneously underfilling O_{BA} (Fig. 6).

2. To assess the extent of any changes, use a fixed fluorescence sample. Prepare your setup for a standard experiment with DL configuration. Using lowest possible power find a suitable sample plane (one with obvious structure) and determine the original maximal FOV and working and working distance (e.g. by moving the sample side to side and checking when a particular image structure leaves the field of view).

3. Prepare setup to measure objective back aperture (O_{BA}) filling factor (Øexc/Øobj) – see initial steps.

4. Completely remove SL from the system. **Before implementing any modifications, remember to switch off the laser!**



Sidenote: Why not simply shift the original SL? We can of course move the IPF closer to TL by simply shifting forward the SL. However, this will <u>increase</u> overfilling O_{BA} (we are hoping to decrease this) which in turn will decrease effective excitation power on the sample plane (we are hoping to increase this).

Fig 6. Optical design of DL(a) and nTC_1 configuration (b-d). Left part shows positions of the lens, right objective back aperture filling factor. (b and c) shows different positions on

L2 and in result different FOV and psf shape. (d) shows configuration (b) without thin lens approximation.

5. Insert new lenses in place of the SL. In our tested nTC_1 solution for the Sutter MOM, we use 2 lenses in place of the SL: L1 and L2, with focal lengths 190 and 175 mm, respectively.

In our implementation, L1 is a modified Sutter MOM SL - VISIR 1534SPR136, Leica (Fig. 7b), and L2 is LA1229 from Thorlabs (Fig. 7c). This specific choice for L1 is one of convenience, but it can be equally be replaced by a new 190 mm lens.

nTC_1 can also be achieved using a single lens. However, in this case, the O_{BA} filling factor will be much smaller and difficult to control, and the psf will disproportionately inflate already for a small FOV increase.

For mounting L1, we use the original SL mount (as our L1 is a modified SL). For assembling and shifting L2, we use a 3D printed lens holder that fits into the existing tube that also holds the TL at its end (Fig. 6a). The 3D printable file Lens_holder.stl is found on our GitHub.

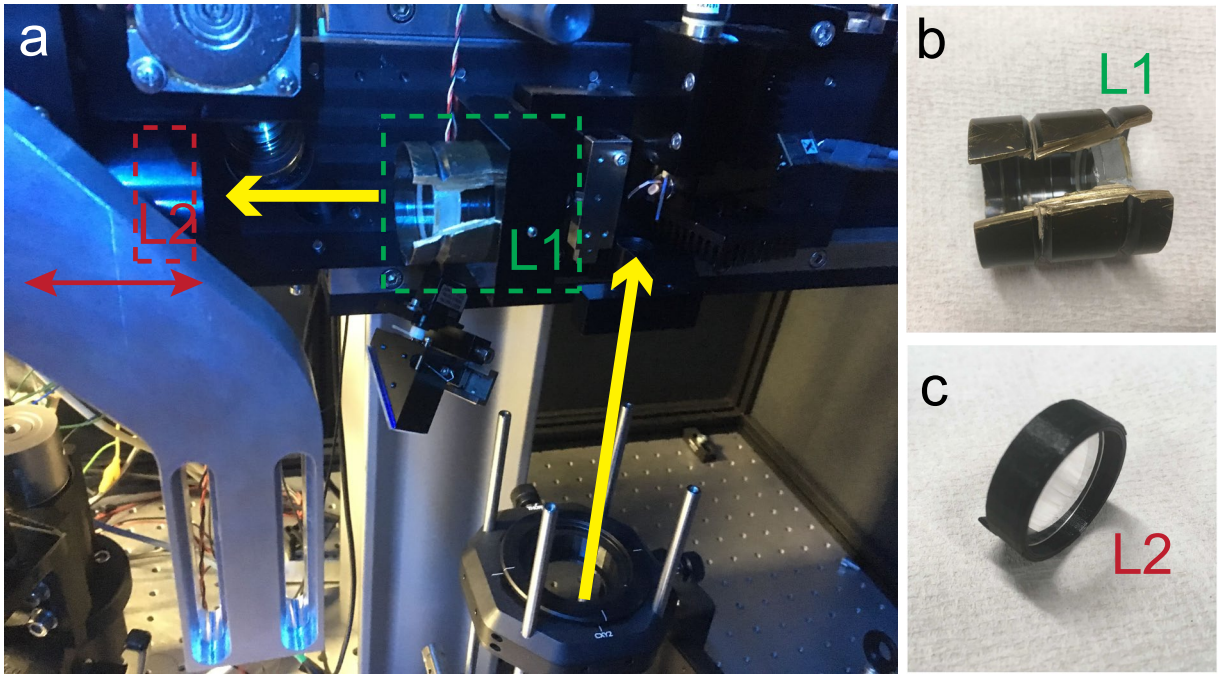


Fig 7. Sutter MOM setup with nTC_1 configuration. Location of L1 and L2 (a), L1 modified from Leica VISIR 1534SPR136 *(b), L2 with custom design 3D printed holder (c)*

6. **Select the "right" L1 and L2 lenses for your setup.**

Sketch out how you want the optical configuration of your setup to look like. Note original SL, TL and their apertures. Note existing holders for lenses and the physical constraints imposed by your setup's mechanics to position and shift L2.

For L1, use a lens with a focal distance $F_{L1} = D_{L1}$, such that by itself, L1 sets up an IFP exactly at the centre of the TL. We use a 190 mm lens, positioned 190 mm in front of the TL.

If desired, you can also use an L1 with shorter or longer focal distance than D_{L1}. However, if the focal distance is shorter, the range of FOV changes generated by shifting of L2 position decreases. In contrast, of L1 focal distance is longer than D_{L1}, it will bring the IFP behind the TL, effectively resulting in nTC_2 (discussed below). In this case we can shift between modification 1 and 2 mode by change a position of L2.

Make sure that the aperture (diameter) of L1 is suitable. It should be in similar range as the original SL ($Ø_{L1} \approx Ø_{SL}$) to make sure the incoming laser beam does not get clipped on the edges during scanning.

For L2, the focal length (F_{L2}) should be in the range between $F_{L1\ and}\ F_{L1}/2$. F_{L2} shortening introduce an overfilling of O_{BA}, or even TL aperture, resulting in power lost and fluctuation of excitation intensity on FOV. F_{L2} extending decrease range of IFP and (Øexc/Øobj) changes.

On our setup, we use a 175 mm F_{L2}, which is shifted along the laser path between 100 and 5 mm from TL.

Ultimately, the exact position vale of F_{L2} as well as the position of L2 relative to L1 and TL will dictate the position of the IFP, and thereby set the (Øexc/Øobj) factor, FOV and the size of the excitation psf.

If possible, choose an aperture of L2 that is similar to the one for L1 ($Ø_{L1} \approx Ø_{L2}$). *In our case it was not possible to use the exact same size, so we used the largest possible L2 diameter (25,5 mm) which just about fit into the 3D printed holder that was slotted into the tube that also held the TL.*

7. Test the result of your changes and establish the range of FOV shifts achievable by shifting L2 up and down the laser path. Start by setting L2 as close as it is possible to L1 and take a scan of your fixed sample. Likely, working distance will be longer than in the original DL configuration so you will need to refocus. Note the difference of a working distance and check the maximal FOV size. Also, zoom in to assess spatial resolution. Now, shift L2 closer to TL and again check FOV and working distance. Both should increase. Note the range of changes that can be achieved with this configuration.

Remember to shift L2 only when you are not scanning and your laser is disabled!

If you find that working distance and FOV starts to decrease again at some point when shifting L2 closer to TL, it means that L2 is further away from L1 compared to L1's focal distance. In this case, put L1 closer to TL or exchange it for the one with longer working distance.

8. As for 7, also check the objective filling factor (Øexc/Øobj) for a different positions of L2 (see initial steps paragraph). Shifting of L2 towards TL will increase underfilling (i.e. will make the laser spot smaller at the level of the objective's back aperture), and should result in an expansion of the *psf*.

If desired, you can try to keep a (Øexc/Øobj) constant even with increasing FOV by exchanging of L2 for one with shorter working distance as you shift it closer to TL. This effectively allows setting FOV somewhat independently of psf expansion (up to a point).

Sidenote: Remember that overfilling of O_{BA} will reduce effective excitation power, and can also lead to uneven excitation power across the FOV.

9. As desired, you can also directly measure the excitation *psf*, for example by imaging fluorescent beads embedded in agarose. When doing so, remember to keep laser wavelength and excitation power at the sample plane approximately constant to ensure fair comparison (measured *psf* dimensions strongly depend on these factors). Note that excitation power at the sample plane should also increase as you decrease Øexc/Øobj, so it is best to directly measure this on the sample plane using a power meter.

nTC_2 configuration

1. In nTC_2, goal is to move the IFP beyond the TL (but still in front of O_{BA}), again while ensuring underfilling of O_{BA} (Fig. 7). Most steps to set-up nTC_2 are very similar to those required to set up nTC_1. Therefore, first consult the description in nTC_1. From here, implementation nTC_2 will be straightforward:

2. Starting from a nTC_1 configuration, remove L1 from the setup and replace L2 with L3. (If you start from a DL setup, simply remove the SL entirely, and find a way to mount L2/L3 in front of the TL).

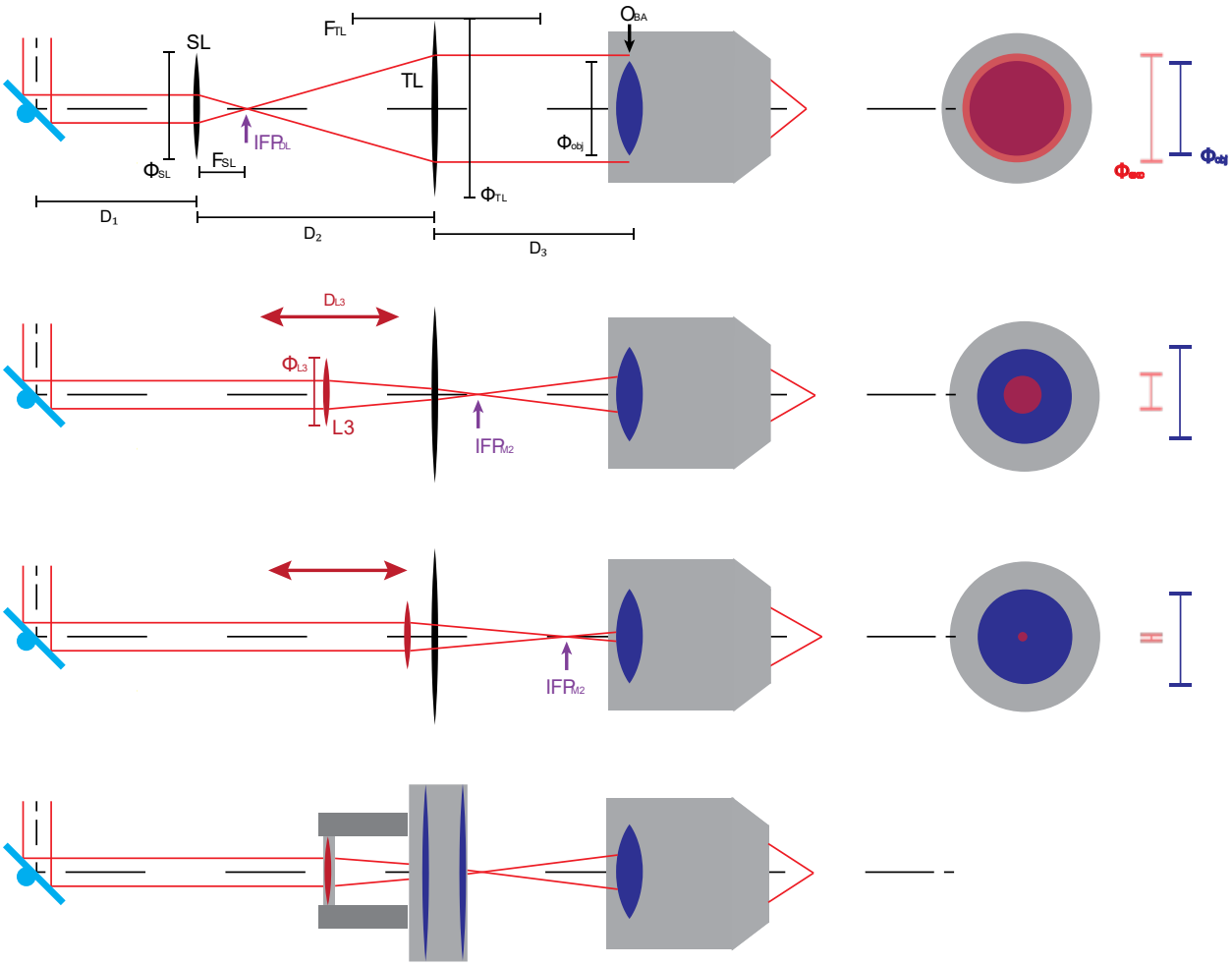


Fig 8. Optical design of DL(top) and versions of nTC_2 configurations (below). Left: positions of the lens(es), right: objective back aperture filling factor. Different positions of L3 and in result different FOV and psf shape. Bottom row shows configuration without thin lens approximation (see introductory definitions). To compare with nTC_1 check Fig 6.

3. **How to select L3**? The goal of L3, depending on its position in front of TL, is to shift the IFP anywhere between the TL and O_{BA}. *In our setup we use a 200 mm focal length Ø$_1$ plano convex lens (LA1708, Thorlabs) which is probably a good starting point for most optical configurations. L3 is assembled exactly like L2 in nTC$_1$*, using a *3D-printed holder. In our configuration we can slide it anywhere between just next to the TL and up to 10 cm in front of it.*

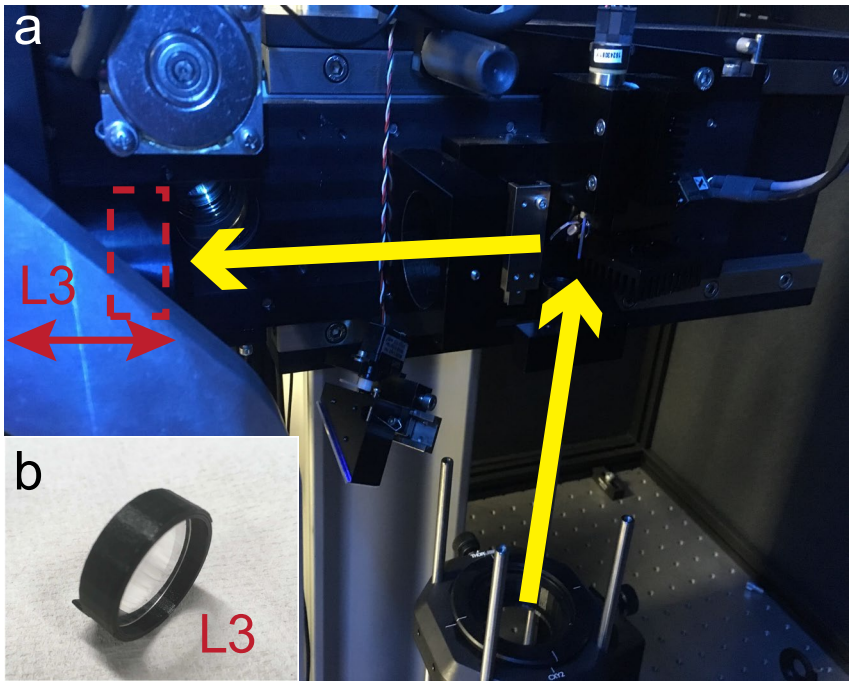


Fig. 9. Sutter MOM setup with nTC$_2$ configuration. Location of L3 (a) and L3 with custom design 3D printed holder (c)

Side note: If you remove all lenses except the TL and Objective (i.e. remove SL/L1/L2$^{(*)}$), your parallel beam coming from the scan mirrors should come into focus beyond the objective back aperture (F_{TL} is always bigger than D_3). Accordingly, O_{BA} will be illuminated by convergent beam. However, we want the beam to diverge, so you will need a L3 lens that is powerful enough to generate an IFP (just) before the laser beam reaches the objective.

Check always that the beam reaching the O_{BA} is divergent, not convergent (for example with a laser viewing card).

Throughout, when choosing new lenses, make sure they have high transmission efficiency in the infrared range such that the laser beam does not get attenuated too much.

Remember to use the highest possible aperture of the lens. Of course, this factor will be limited by space in Your scope (in our cases size of the tube where L3 is moved. Selection of a too small aperture will likely result in inefficient collection of the excitation beam after the mirrors (as now there is no longer a SL to direct the beam onto L3).

Electrically Tunable Lens

In our setup we implemented an Electricaly Tunable Lens (ETL) for rapid axial focussing. Our ETL is an EL-16-40-TC20D, Optotune, controlled by a custom driver and custom written software (see GitHub).

ETL Optics

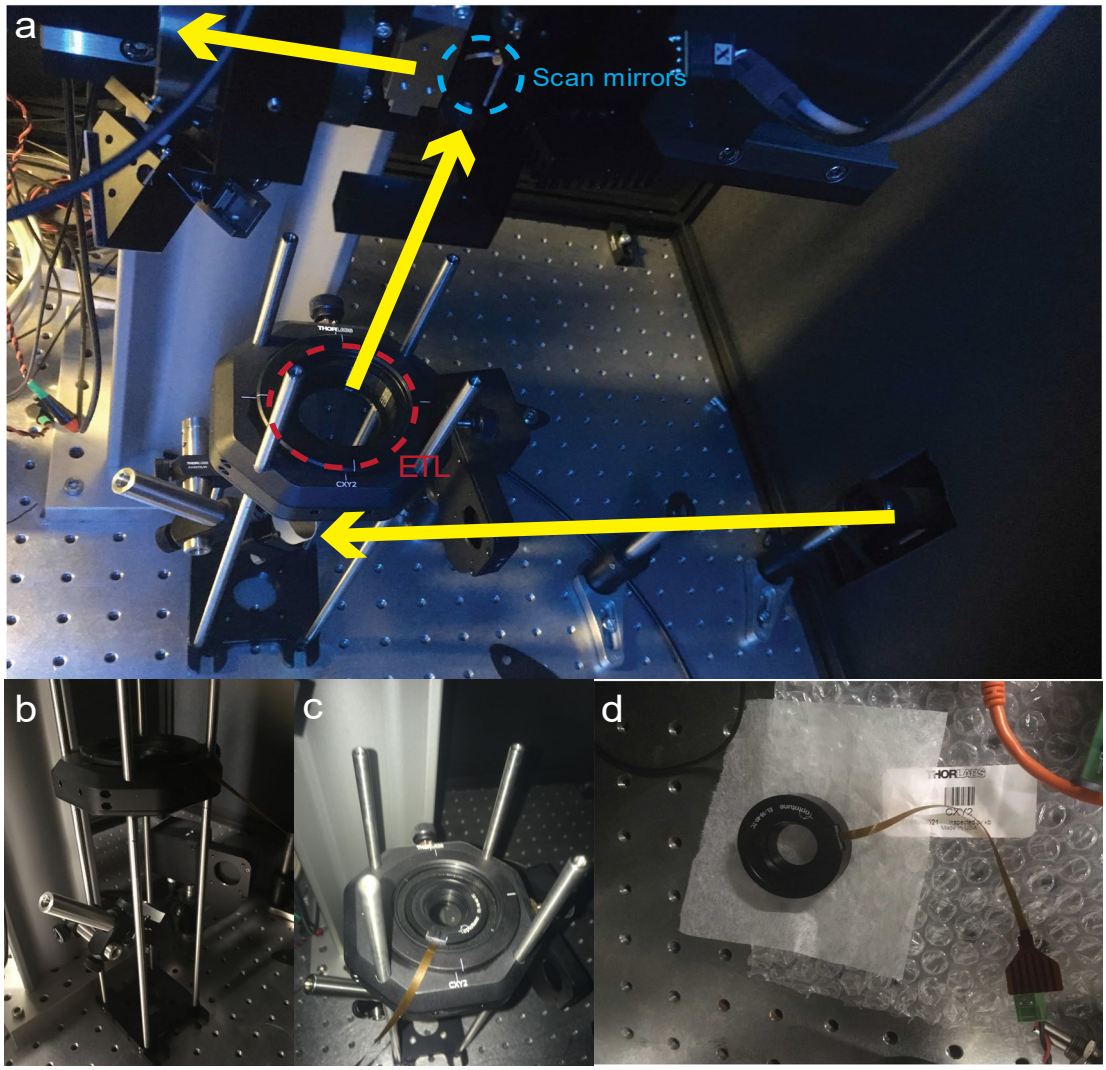


Fig 10. Sutter MOM setup with cage for ETL (a). View of the cage with ETL holder, and mirror reflecting excitation beam to scan mirrors (b), ETL on the holder (c) and ETL EL-16-40-TC20D by itself (d).

In our approach, we decided to use an ETL which operate in a range of -10 to +10 dpt. This means that it focuses from infinity (∞, no focus, parallel beam) to -10 cm for negative current and from ∞ to +10 cm for positive current. If the ETL is not supplied by any current, it does not strongly interfere light which is going through (according to the product manual, it introduces a negligible, long range focus).

The main idea is to locate the ETL far away from the objective (compare to original idea presented by Grewe (Grewe, Voigt, van 't Hoff, & Helmchen, 2011)). By changing the ETL's focus from infinity to a positive value (supply positive current) we can then introduce a small convergence into the excitation beam before it hits the scan mirrors. This effectively elevates the imaging plane under the objective. In our solution, we are operating around 25% of a maximum ETL power, which give us an effective focus shift of up to 600 µm.

On our setup, the ETL in mounted horizontally on custom cage build from Thorlabs elements (more details in bill of materials, BOM). Fig. 10a presents the layout of the ETL holder on our Sutter MOM setup. We replaced the Sutter MOM-s original tube with mirror (which bounces the excitation beam from below onto the scan mirrors) by our cage including a new mirror. This new mirror can be adjusted in xyz directions, and in reflection angles, to help setup a perfectly vertical beam. The ETL holder is adjustable in x and y, and is located 20 cm below the first scan mirror.

ETL driver hardware

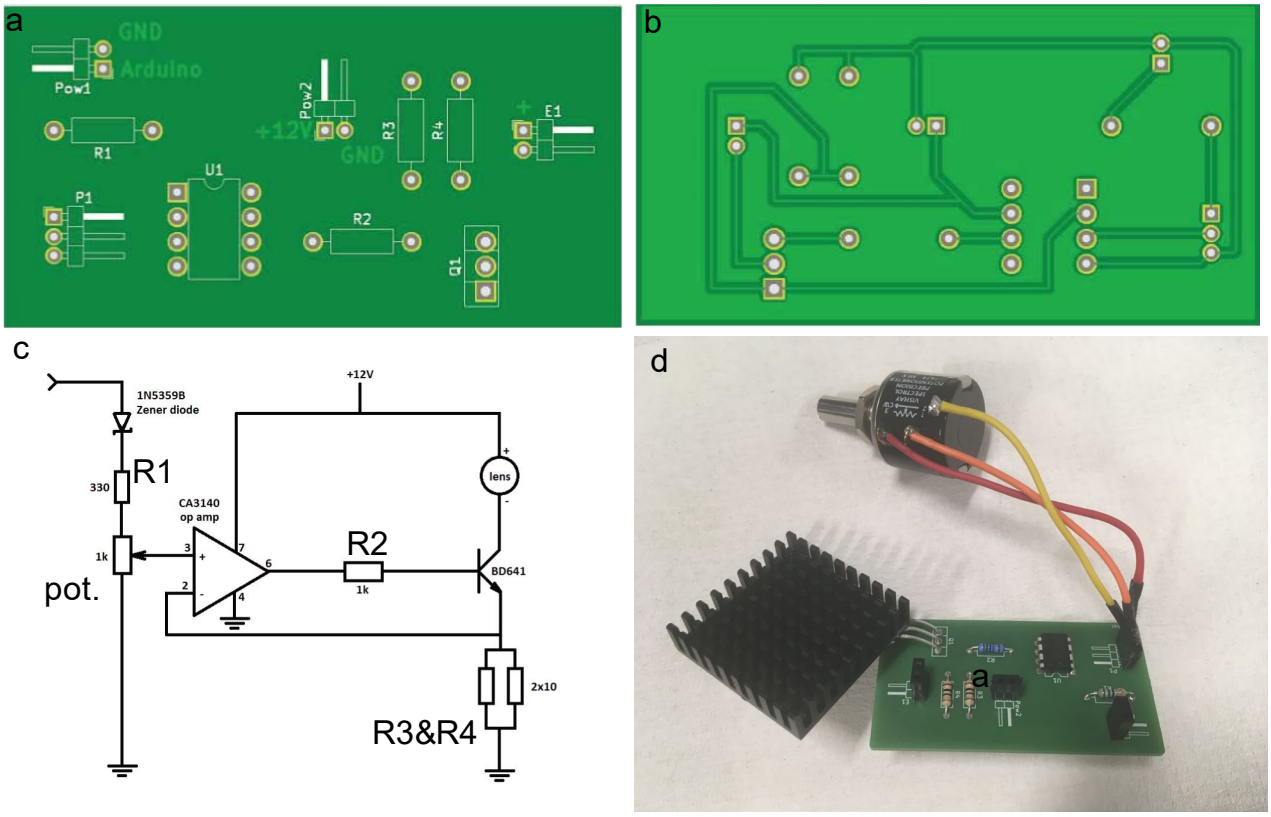


Fig 11. Top (a) and bottom (b) view of ETL driver printed circuit board (PCB). Schematic of electrical circuit (b) and photograph of the assembled driver (d). The Gerber files for producing the PCB are available on the project GitHub.

To control our ETL, we are using a custom designed current driver combined with an Arduino Due microcontroller. The driver's overall goal is to generate ETL command signals that are line-synchronised with the 2-photon scan engine. For this, the Arduino Due reads a line-signal from the scan engine (e.g. a copy of the fast-mirror command, or a line-synchronised TTL pulse) to count lines. For this, the Arduino is reading such as signal on Analog pin A1. In our case, this signal is simply a copy of the "banking signal" which we send to the Pockels Cell during the scan retrace (the latter is an optional "trick" to reduce bleaching during retrace times, this has been discussed previously e.g. in Euler et al. 2009 Pflueger's Archive).

Upon reading this signal, the Arduino "counts scan lines", and executes driver commands accordingly, based on a custom script controlled from a Matlab GUI (below). For this, the Arduino sends a driving signal through its Digital-Analog-Converter (DAC) output port into the custom driver board. On the Arduino side, this DAC signal is ranges from 0 to 4,095 (12 bit), which corresponds to a voltage range of 0.55 - 2.75 V, which serves as the sole scan input to the custom driver board. First, the 0.55 V offset (corresponding to 0 on the Arduino) is removed using an operational amplifier (op-amp). Next, the remaining signal is scaled using a potentiometer which allows defining the maximum current sent to the ET (which also indirectly defines its resolution). Finally, this signal is used to open a negative-positive-negative (NPN)-transistor, which works as a switch for a current from a separate +12V 0.42 A power supply.

Note that our current driver works only for a positive current, even if our ETL works also for the negative.

Note that we are using an additional Zener diode as a connector between Arduino output and our current driver input. Reason of using it was additional protection of current driver and ETL because of problems with series of unstable Arduino boards which we have been using (some of them temporary didn't work properly). This Zener diode introduce additional voltage drop to the system, so it doesn't need to be done on op-amp.

Figure 11a and b shows the PCB of our driver. PCB file *ETL_driver_board.kicad_pcb* is available for download on our GitHub. In addition, you will need the following electronic components (full details on BOM):

- Zener diode (1N5359B) (additional used as connector between Arduino and driver board – see note above),
- R1 330 Ω,
- 1kΩ potentiometer,
- Op-amp CA3140EZ,
- R2 1 kΩ,
- R3&R4 10 Ω.
- N6039G transistor
- 12 V 420 mA power supply.

All connection pins are labelled on the board (Arduino, +12, + (ETL), and GND for Arduino and power supply). The Pin next to + is a negative connection for ETL.

Note that in addition we attached a heat sink to the N6039G transistor which helps to dissipate heat if the driver is operated for prolonged periods of time (see photo in Fig. 11d).

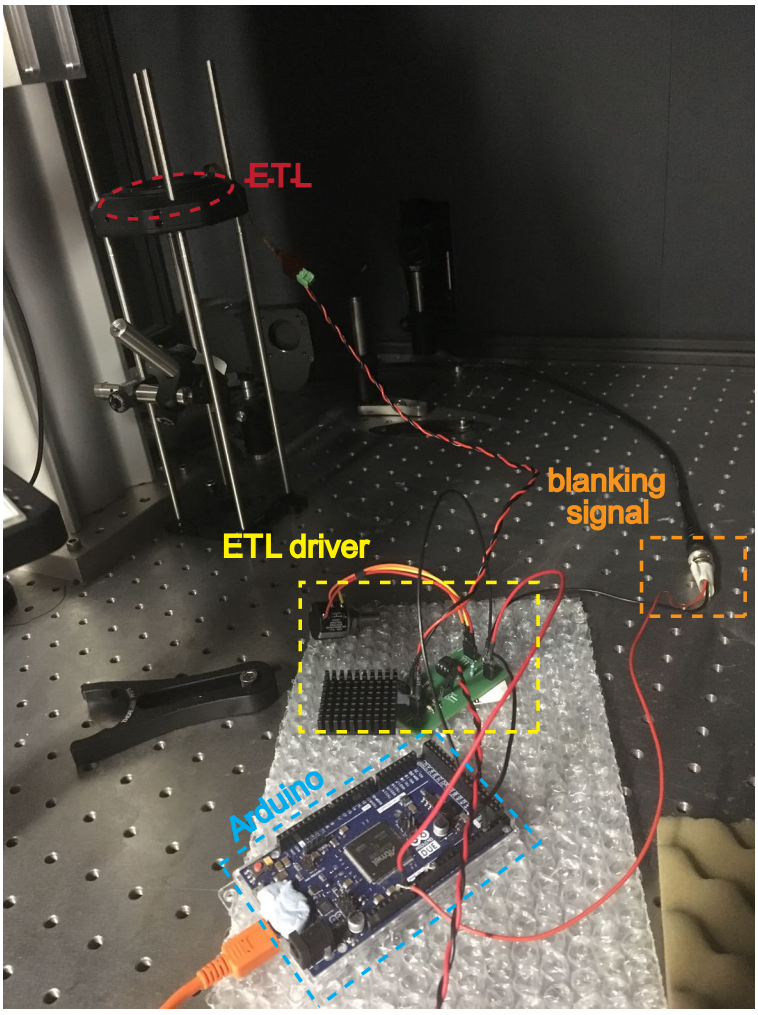


Fig. 12. Our ETL driving circuit with labelled ETL, ETL driver, Arduino Due board and blanking signal.

ETL software control

Two pieces of custom code are required: An Arduino Script, and the Matlab Graphical User Interface (GUI). Both are deposited on the project GitHub.

There are two versions of each, “6planes” and “HP”. The former scripts are for plane tilting and multiplane imaging, while the latter is for half-pipe scans.

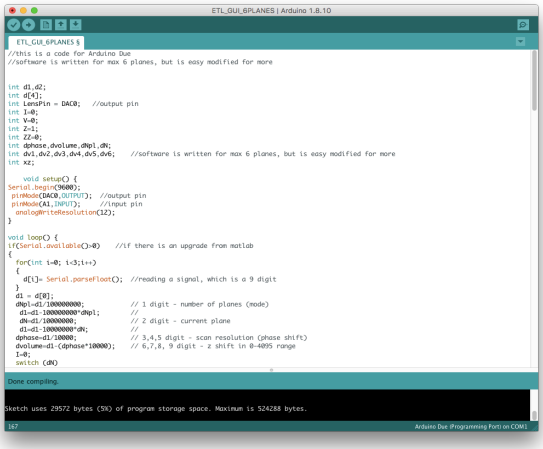
Arduino: ETL_HP.ino or ETL_GUI_6Planes.ino
Matlab: Gui_HP.m or Gui_6planes.m (and *.fig” files)

Our software control is fully independent from the software used to control the 2P microscope (e.g. ScanImage, or in our case ScanM), meaning that it can in principle be seamlessly integrated with any working 2p system so long as a line-synch signal is available to be read by the Arduino.

To communicate with the Arduino and upload the script, download the Arduino integrated development environment (IDE). The Arduino IDE is freely available at https://www.arduino.cc/en/main/software. Tutorials on how to connect the Arduino via serial link and upload scripts are widely available online, so we will not cover this step

here. You only have to upload the script once (the Arduino stores it locally), but you will need to keep the Arduino connected to your computer prior to changing any scan configuration so as to allow the Matlab script to communicate with it (see below).

Our GUI is written in MATLAB (Mathworks), which is widely using in academia. If you do not have it on your computer, check if your institution has a licence for it. There is also a 30-day free trial version https://uk.mathworks.com/products/matlab.html.




```
//this is a code for Arduino Due
//software is written for max 6 planes, but is easy modified for more


int d1,d2;
int d[4];
int LensPin = DAC0; //output pin
int I=0;
int V=0;
int Z=1;
int ZZ=0;
int dphase,dvolume,dNpl,dN;
int dv1,dv2,dv3,dv4,dv5,dv6; //software is written for max 6 planes, but is easy modified for more
int xz;

void setup() {
 Serial.begin(9600);
 pinMode(DAC0,OUTPUT); //output pin
 pinMode(A1,INPUT); //input pin
 analogWriteResolution(12);
}

void loop() {
if(Serial.available()>0) //if there is an upgrade from matlab
{
 for(int i=0; i<3;i++)
 {
 d[i]= Serial.parseFloat(); //reading a signal, which is a 9 digit
 }
 d1 = d[0];
 dNpl=d1/100000000; // 1 digit - number of planes (mode)
 d1=d1-100000000*dNpl; //
dN=d1/10000000; // 2 digit - current plane
 d1=d1-10000000*dN; //
dphase=d1/10000; // 3,4,5 digit - scan resolution (phase shift)
 dvolume=d1-(dphase*10000); // 6,7,8, 9 digit - z shift in 0-4095 range
 I=0;
 switch (dN)
 {case 1:
 dv1=dvolume;
 case 2:
 dv2=dvolume;
 case 3:
 dv3=dvolume;
 case 4:
 dv4=dvolume;
 case 5:
 dv5=dvolume;
 case 6:
 dv6=dvolume;}
}
int sensVal=analogRead(A1); //reading blanking signal
switch (dNpl)
{
 case 1: //for xyz scans

 if (sensVal+Z>1200){ // 1200 is a average noise value read from A1. PLEASE CHECK IF IT IS                    //CORRECT FOR YOUR SETUP
int xz=round(3*4095/dphase);
Z=1;
 I=I+1;
 }
V=I*dvolume;// z shift for xyz

if (I==dphase){
I=0;
 }

case 2: // 2 planes
if (sensVal<10){
Z=200; //2Z=200 is a treshold value betwen HI and LOW of blanking signal for our setup.
//PLEASE CHECK IF IT IS CORRECT FOR YOUR SETUP
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 }
if (I==dphase){          //change z volume no 1
 V=dv1;
 }
if (I==dphase*2){        //change z volume no 2
 I=0;
 V=dv2;
}
 break;
case 3:                  // 3 planes
if (sensVal<10){
Z=200;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 }
if (I==dphase){
V=dv1;
 }
if (I==dphase*2){

V=dv2;
}
if (I==dphase*3){
I=0;
 V=dv3;
}
break;
case 4: // 4 planes
if (sensVal<10){
Z=200;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 }
if (I==dphase){
V=dv1;
 }
if (I==dphase*2){

V=dv2;
}
if (I==dphase*3){

V=dv3;
}
 if (I==dphase*4) {

V=dv4;
I=0;
 }
break;

case 5: // 5 planes
if (sensVal<10){
Z=200;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 }
if (I==dphase){
V=dv1;
 }
if (I==dphase*2){

V=dv2;
}
if (I==dphase*3){

V=dv3;
}
 if (I==dphase*4) {

V=dv4;
}
 if (I==dphase*5){
I=0;
 V=dv5;
}
break;

case 6: // 6 planes
if (sensVal<10){
Z=200;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
```

```
case 6: // 6 planes
if (sensVal<10){
Z=200;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 }
if (I==dphase){
V=dv1;
 }
if (I==dphase*2){

V=dv2;
}
if (I==dphase*3){

V=dv3;
}
 if (I==dphase*4) {

V=dv4;
}
 if (I==dphase*5){

V=dv5;
}
if (I==dphase*6) {
I=0;
 V=dv6;
}
 break;
case 7: // single plane shift for z seachhing
V=dvolume;
break;
}
analogWrite(LensPin,V); //write value to current driver
}
```

Fig 13. Arduino code with comments for multiplane and xyz imaging.

ETL software control: Multiplane and XYZ imaging

First, upload the Arduino script ETL_GUI_6Planes.ino (called a “Sketch” in Arduino) to the Arduino board. Once this is done, you can close the Arduino IDE.

Next, open MATLAB and launch Gui_6PLANES.fig and Gui_6PLANES.m (If you are unsure how to do this, look for an online tutorial how to launch code in MATLAB.). In short, put both files into a common folder, change the MATLAB path to this folder, and execute Gui_6PLANES.m. This should bring up the GUI (Fig. 14).

From within the MATLAB GUI, connect the Arduino. For this, select the serial (COM)-port to which the Arduino is connected (Fig. 14, list 1) (It will be the same that you previously used to upload the Arduino script - if you are unsure how to identify the Arduino’s COM-port, search for one of the many tutorials online). In our cases it happens to be COM 13. *Note: ensure that the COM port is not busy – this can easily happen if the Arduino IDE’s serial monitor happens to the open, or if it is still uploading the script.*

Next tick the “ETL ON” checkbox to connect to the ETL. This sets up an open serial communication between Matlab and the Arduino (*this also means that you can no longer communicate with the Arduino from elsewhere, e.g. the Arduino IDE. So, if you want to upload e.g. a different Arduino script, you need to untick the ETL ON box again*).

From here, you are ready to start using the ETL in synch with the scan commands. *Remember, in our implementation, the ETL can only lift the imaging plane, so the lowest position of your desired scan should be defined by the actual objective position (ETL at position 0)*

Search-z function. To statically change the z-plane of your scan, simply enter a value between 0-4,095 into window (3) and press button (4) to confirm. Each time you do this the GUI sends a command to the Arduino, which in turn will send a static driver signal to the ETL and thus shift the imaging plane. This function is useful to explore by hand which values entered into the GUI correspond to what kind of actual z-shift. Confirm that this works by running any xy-scan on your scan software and changing this number (you don’t need to stop the scan for this). It may take a few seconds for any change to take effect.

Fig 14. MATLAB GUI for the multiplane and xyz imaging.

Z-shift calibration (optional). Use a structured fluorescence sample (e.g. a test-slide) that allows you to easily identify specific structures at different imaging depths. Prepare your setup for standard 2P imaging, find a suitable image plane without engaging the ETL (ETL disconnected or set to “0”). Now, put the maximum allowed value of 4,095 into window (3), and push button (4). This should jump the imaging plane upwards, and potentially quite substantially so (in our case some 600 microns). Now go down in your sample using the

objective and find the same imaging plane again. Take a note of the effective note the z-shift that you needed to use to compensate for the ETL. This is the maximum z-jump range possible with your configured optics and ETL-driver settings as defined by the potentionmeter (see driver). If this range is too small or too large for your purposes, rotate the potentiometer by a few turns and repeat the above. Once you are happy with the effective z-jump range, ensure that the dependency between the number entered into the GUI and the effective z-jump is approximately linear (it is very linear in our case, where we use ~25 max driving current on the ETL).

***Scan-synchronised ETL commands*.** For any of the above procedures, the line-counting feature of our custom driver is not really used as the driver is just outputting static commands. However, for multiplane imaging, tilting the scan plane (xyz scans) or for halfpipe scans (see next chapter), the ETL must be synchronised to the scan mirrors with the line-synch connection (see driver chapter above).

***Multiplane and staircase scans*.** Here, the control logic of our GUI is simple – for multiplane imaging it simply jumps the ETL to a new current (and hence effective focus position) once it counts a certain number of scan lines based on the values you enter under “select y resolution” and “Number of planes”. For example, if you choose a scan with 32 scan lines in your 2p control software (y = 32), you can then select 32 under “select y resolution” and the ETL will jump to a new plane every 32 lines, in perfect synch with the scan. Then you can define the number of planes. For example, if you choose 2 planes, it will execute 32 lines and depth 1, and then 32 lines at depth 2, and thereafter jump back to depth 1 etc.
Here, the specific depths it will jump to each time is defined in the window titled “planes”. Values entered in planes that are not used (e.g. plane 3 in a 2 plane configurations) will simply be ignored. For now, the GUI is configured for up to 6 planes to not clutter the GUI. However, it would be relatively straightforward for any experienced programmer to change this in the MATLAB code provided.

While this setup lets you set up straightforward multiplane scans, it can also execute some more “interesting” scan paths simply by playing with the numbers. For example, you can run a “step-scan” (i.e. half a scan at plane 1, and the other half in plane 2 – as for example shown for simultaneous eye-brain imaging in zebrafish in the accompanying MS). For this, taking our previous example, simply change the value of 32 to 16 under select y resolution. This will jump the ETL twice per frame and generate the intended effect).
Note also that there is no requirement for the scan planes to be evenly spaced, or event to be executed in any particular order. For example you could sample plane 1 at twice the rate as plane 2 and 3 by defining a 4-plane scan and setting z-values for the four planes accordingly.
Note: If you abort a scan at any time other than exactly at the end of a full frame, you may find that next time you run a multiplane or similar scan that the ETL and the scan mirrors are out of synch. This is because the GUI/Arduino only counts scan lines, and may thus have some remaining lines stored. To ensure this doesn’t happen, don’t update the Matlab GUI during scans. If the problem persists, simply stop the scan, and reload whatever ETL command configuration you want to execute. This will reset the counter.

XYZ scans To operate i.e XYZ mode (i.e. plane tilting), select your scan resolution (9), in the same manner for multiplane imaging (5). Select z-multiplication (10), the value of the z-shift of the last line. Z-multiplication works as follows: Say you use a 100x32 xy scan configuration in your scan mode and select a multiplication of 1. This means that for each new scan line, the ETL will be shifted by an ETL value of 1 to thus reach 32 by the last line (which corresponds to a shift of 32 in window 3). If for the same scan you will select a z-multiplication of 2, each line will be shifted by 2 to finally reach 64 by line 32 (and so on). *Note that the product of the number of the number of scan lines and the multiplication factor cannot exceed 4,095.*
In this way, you can also set-up a perfectly vertical scan. The simplest way might be to select a line-scan configuration in your scan software (i.e. y mirror silent). Alternatively, select an image scanning configuration and disconnect the y-mirror (just unplug it at the DAQ board - while it is not scanning!). The latter has the benefit that your scan software will display the XZ scan (as opposed to XY) as an image with Y corresponding to Z (some scan software solutions also do this for line scans so this may not be always necessary).

Note that in a "real" XZ mode (i.e. perfectly vertical), it is likely that the excitation volumes (point spread function) of subsequent z-lines will overlap because the psf is usually vertically elongated. This will give a blurred image. One way to ameliorate this effect is to slightly tilt the scan by engaging the Y mirror such that that subsequent psfs overlap less (i.e. a XYZ scan, where Y and Z are linked with the z-multiplication factor).

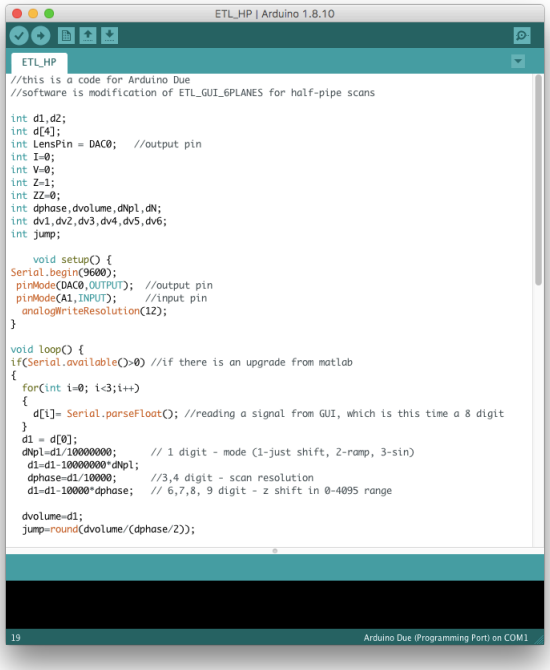
ETL software control: Half pipe scans
Upload ETL_HP.ino (Fig. 15) to the Arduino (instead of the "6planes" one used above). Similarly, launch the corresponding Matlab scripts GUI_HP.fig and GUI_HP.m (as above) and connect the Arduino to the GUI as before (window 1 and 2, Fig. 16).

The half-pipe GUI works in a very similar manner to the multiplane/xyz GUI.

As before, select your scan y-resolution (3). This GUI is set-up for 3 different modes (4). Standard z-shift of the entire plane ("just move"). Second, a ramp where the z-position updates linearly for each scan line up to half of the value defined under (3), and after that lineally decrease again (i.e. like a roof on a house). Third, a axial sine-wave scan. This operates like the ramp scan, except that now the z-value is updated in a sinusoidal fashion rather than linearly, which gives rise to a half-pipe-like z-scan pattern. The peak z-amplitude of these scan options is selected using slider (5) and confirmed using button (6). By definition, half-pipe and ramp scans always start from an ETL-value of zero.

ETL software control: Half pipe scans with 3 sinusoidal planes.
This mode operate on the same GUI as a standard half pike scan. Only one difference is on Arduino code. Upload ETL_HP_sin3planes.ino (Fig. 17) to the Arduino. This file is just a small modification of ETL_HP.ino described above. The difference is that third, sinusoidal scan mode operated on 3 planes imaging frame by frame. As using our ETL driver we are not able to supply negative current, now the value from slider 5 (fig. 16)

correspond to the flat, first plane, and two other planes are a sinusoidal positive half pipe with an amplitude at 2 values of slider 5, and negative sinusoidal half pipe with negative amplitude at 0.




```
//this is a code for Arduino Due
//software is modification of ETL_GUI_6PLANES for half-pipe scans

int d1,d2;
int d[4];
int LensPin = DAC0;                 //output pin
int I=0;
int V=0;
int Z=1;
int ZZ=0;
int dphase,dvolume,dNpl,dN;
int dv1,dv2,dv3,dv4,dv5,dv6;
int jump;

void setup() {
Serial.begin(9600);
 pinMode(DAC0,OUTPUT);          //output pin
 pinMode(A1,INPUT);             //input pin
 analogWriteResolution(12);
}

void loop() {
if(Serial.available()>0)            //if there is an upgrade from matlab
{
 for(int i=0; i<3;i++)
 {
 d[i]= Serial.parseFloat();         //reading a signal from GUI, which is this time a 8 dig
 }
 d1 = d[0];
 dNpl=d1/10000000;                  // 1 digit - mode (1-just shift, 2-ramp, 3-sin)
d1=d1-10000000*dNpl;
 dphase=d1/10000;                   //3,4 digit - scan resolution
 d1=d1-10000*dphase;                // 6,7,8, 9 digit - z shift in 0-4095 range

dvolume=d1;
 jump=round(dvolume/(dphase/2));

I=0;
}

int sensVal=analogRead(A1);
switch (dNpl)
{
case 1:                             // just shift
 V=dvolume;
 break;

case 2:{                            // ramp
if (sensVal<10){
Z=400;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 V=I*jump;
 }
if (I>=dphase/2){
V=(dphase/2)*jump-(I-(dphase/2))*jump;
 }
if (I==dphase){
I=0;

}
 break;
}
case 3:                             // sin
if (sensVal<10){
Z=400;
}
 if (sensVal+Z>1200){
 Z=1;
 I=I+Z;
 V=sin(radians(180*I/dphase))*dvolume;
 }

 if (I==dphase){
 I=0;
}
 break;
}
analogWrite(LensPin,V);
}
```

Fig. 15. Arduino code for Half-Pipe scans

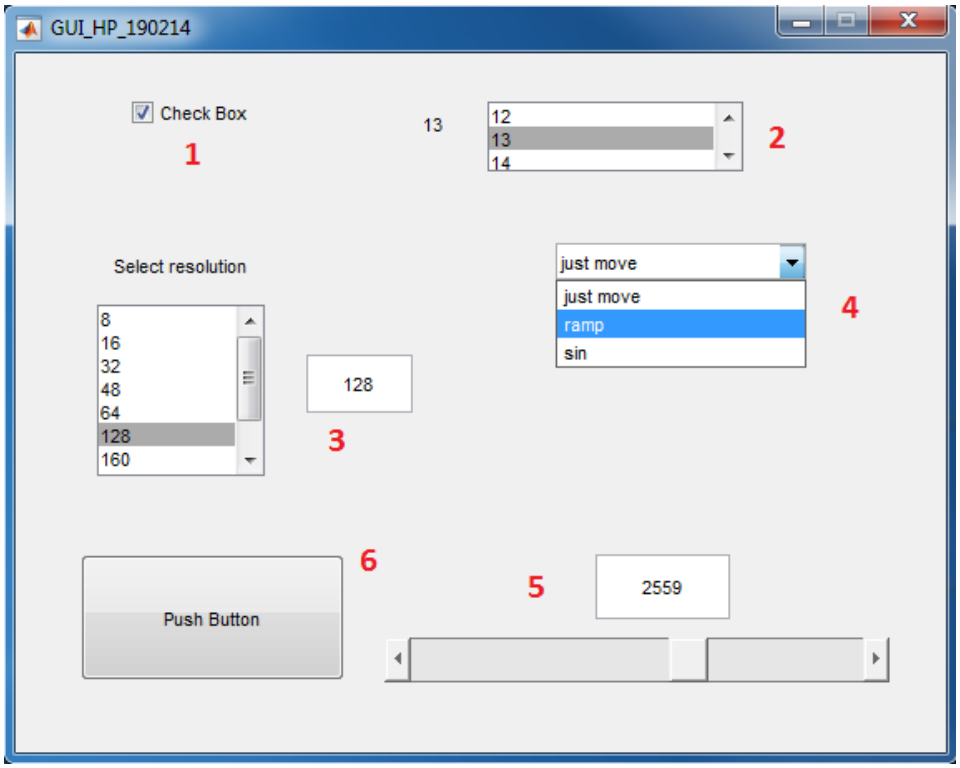


Fig. 16. *MATLAB GUI for Half-Pipe Scans.*


```
d[i]= Serial.parseFloat();          //reading a signal from GUI, which is this time a 8 dig
}
d1 = d[0];
dNpl=d1/10000000;                   // 1 digit - mode (1-just shift, 2-ramp, 3-sin)
d1=d1-10000000*dNpl;
dphase=d1/10000;                    //3,4 digit - scan resolution
d1=d1-10000*dphase;                 // 6,7,8, 9 digit - z shift in 0-4095 range

dvolume=d1;
jump=round(dvolume/(dphase/2));

I=0;
}

int sensVal=analogRead(A1);
switch (dNpl)
{
case 1:                             // just shift
V=dvolume;
break;

case 2:{                            // ramp
if (sensVal<10){
Z=400;
}
if (sensVal+Z>1200){
Z=1;
I=I+Z;
V=I*jump;
}
if (I>=dphase/2){
V=(dphase/2)*jump-(I-(dphase/2))*jump;
}
if (I==dphase){
I=0;

}
break;
}
```

```
//this is a code for Arduino Due
//software is modification of ETL_GUI_6PLANES for half-pipe scans

int d1,d2;
int d[4];
int LensPin = DAC0;                 //output pin
int I=0;
int V=0;
int Z=1;
int ZZ=0;
int dphase,dvolume,dNpl,dN;
int dv1,dv2,dv3,dv4,dv5,dv6;
int jump;

void setup() {
Serial.begin(9600);
pinMode(DAC0,OUTPUT);               //output pin
pinMode(A1,INPUT);                  //input pin
analogWriteResolution(12);
}

void loop() {
if(Serial.available()>0)            //if there is an upgrade from matlab
{
for(int i=0; i<3;i++)
{
```

```
case 3: // sin with up, down and flat (3planes)
if (sensVal<10){
Z=400;
}
if (sensVal+Z>1200){
Z=1;
I=I+Z;
V=dvolume;
}
if (I>=dphase){
V=dvolume+sin(radians(180*I/dphase))*dvolume;
}
if (I==dphase*3){
I=0;
}

break;
}
analogWrite(LensPin,V);
}
```

Fig. 17. Arduino code for Half-Pipe scans with sinusoidal 3planes imaging.