Tel-Aviv University

Faculty of Engineering

School of Electrical Engineering

אוניברסיטת תל-אביב

הפקולטה להנדסה

בי"ס להנדסת חשמל

“Three-dimensional positioning of ultrafast beam steering elements for volumetric multiphoton imaging reconstruction in mammalian brain”

Project Number: 15-1-1-1027

Final Report

מבצעים:

חגי רוסמן 026553123

ערן גרבינר 301348934

מנחים:

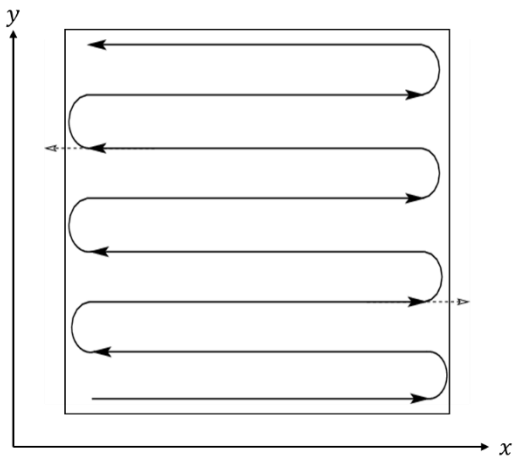
ד"ר פאבלו בלינדר אוניברסיטת ת"א

מקום ביצוע הפרויקט: Department of Neurobiology, TAU

The System:

The system is divided into 3 major subsystems: the laser, optical path and the sensing part. Our project is within the sensing part of the system. The laser is a "Coherent Chameleon Ultra ii". A Titanium-Sapphire laser with 140 [fs] width, frequency of 800 [MHz] and wavelength of 680-1050 [nm]. The uniqueness of the laser is in its ability to create a 2-photon fluorescence from a fluorescent material that was injected to the examined subject. The spread of the fluorescent material in the tissue can reveal details about the functionality of the neuro and cardiovascular systems of the subject.

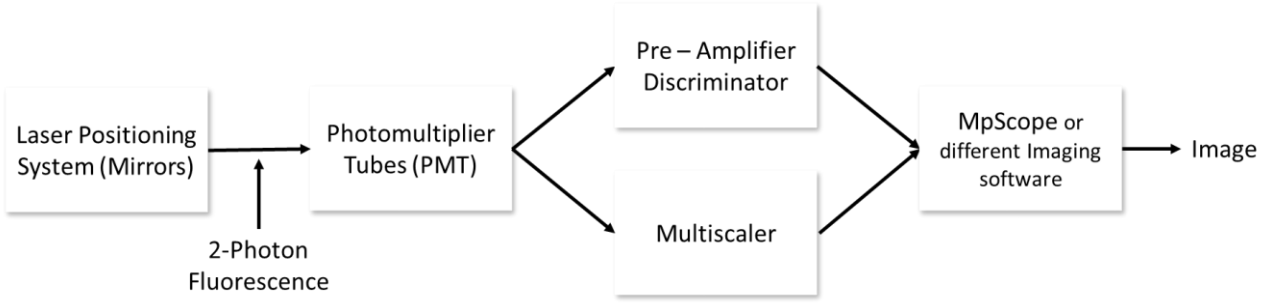
The laser beam performs an automatic scan of a desired frame using two Galvanometric mirrors. Each controls a different axis. If the frame is a square length a, the complete motion cycle of the mirror allows the beam to complete one full travel length a. The mirrors change their angles with different speed: while the slow mirror completes one cycle the fast one can finish few dozens of them creating a raster scanning pattern as shown in drawing 1.



Drawing 1: *Raster scanning. In this example, the x axis is 8 times faster than y.*

While scanning, a 2-photon emission may occur and the photons might hit the PMT sensor. The PMT transform the photons into a massive amount of electrons creating an electric spike. The PMT has a Poisson behavior i.e. the amount of electrons may change from one event to another even if the intensity or amount of photons is the same.

From the PMT there are two optional paths depending on the equipment the user decided to use. The purpose of this stage is to transforms the PMT's signal into a valid measurement of a photon packet. It can be done by a pre-amp discriminator which preforming an average measurement over time. Or a Multiscaler, which is much more accurate and can actually perform 'photon count'. It can identify the signal from the PMT without averaging in time. To conclude, using the pre-amp we get an analog signal representing the signal from the PMT, and using the Multiscaler we can perform a distinct photon count. Either way, the last stage is an imaging software which is currently MpScope (the full path is shown in drawing 2).



Drawing 2: *The path from emission to image in the 'sensing' stage of the system.*

The challenge of the imaging software is to determine the location of the photons based only on the time stamp of their measurement.

Different Problems of Constructing an Image:

The basic challenge in the construction of an image is that our measurements contains only data regarding the time of arrival of photons. There is no data about the origin of those photons within the tissue. The only data that has connection to location is the voltage of the Galvo mirrors which determines the angle and therefore the location of the beam. Association of even one measurement to an exact location can help anchoring the rest.

As said, when examining a frame, the Galvo mirrors are traveling back and forwards over the frame using fast and slow axis. In order to create an image based on the measurements, one should associate each photon packet to the exact location of the Galvo mirrors during this specific measurement. Naturally we can divide the frame into pixels, creating an $n \times n$ matrix. Every travel from one side to another in the fast axis will create a row. One cycle in the slow axis creates an image. Each pixel is a time gap. During it, the mirrors diverted the beam to scan above the pixel's 'area'. The problem is how to determine these time gaps, or "Delta-Time Pixel" (DTP), for each pixel.

The travel from one side to another isn't linear: the mirror's speed is changing and experiencing sharp accelerations, especially near the raster's edges, where the mirror is turning. Hence, each pixel has a different DTP. More, due to the snake pattern, the odd rows will be constructed differently than even ones - from left to right or the opposite. For one direction (odd rows for example), the acceleration part would be on the right side of the image and the deceleration on the left. The opposite will occur for the opposite direction (even rows for example). This occurs since the voltage being inputted to the mirrors has different characteristics during rise and fall. As closer we are to the raster's center, where the mirrors has zero acceleration (the acceleration is changing direction), the mirror's speed would appear constant. If the speed is constant, we can assume a linear behavior – same DTP for each pixel. To conclude, for the determination of the DTP for each pixel, in order to know which measurements to associate to it, we need to determine how far we are from linear behavior and correct it.

Another problem that might occur is "Pixel Shifting". If the pixels were positioned with a shift, hence did not start in the right place of the frame the image will have a duplication phenomenon as seen in image 1. The shift effects differently on odd and even lines creating de facto 2 images, one for odd and one for even which don't anchored right to one another.

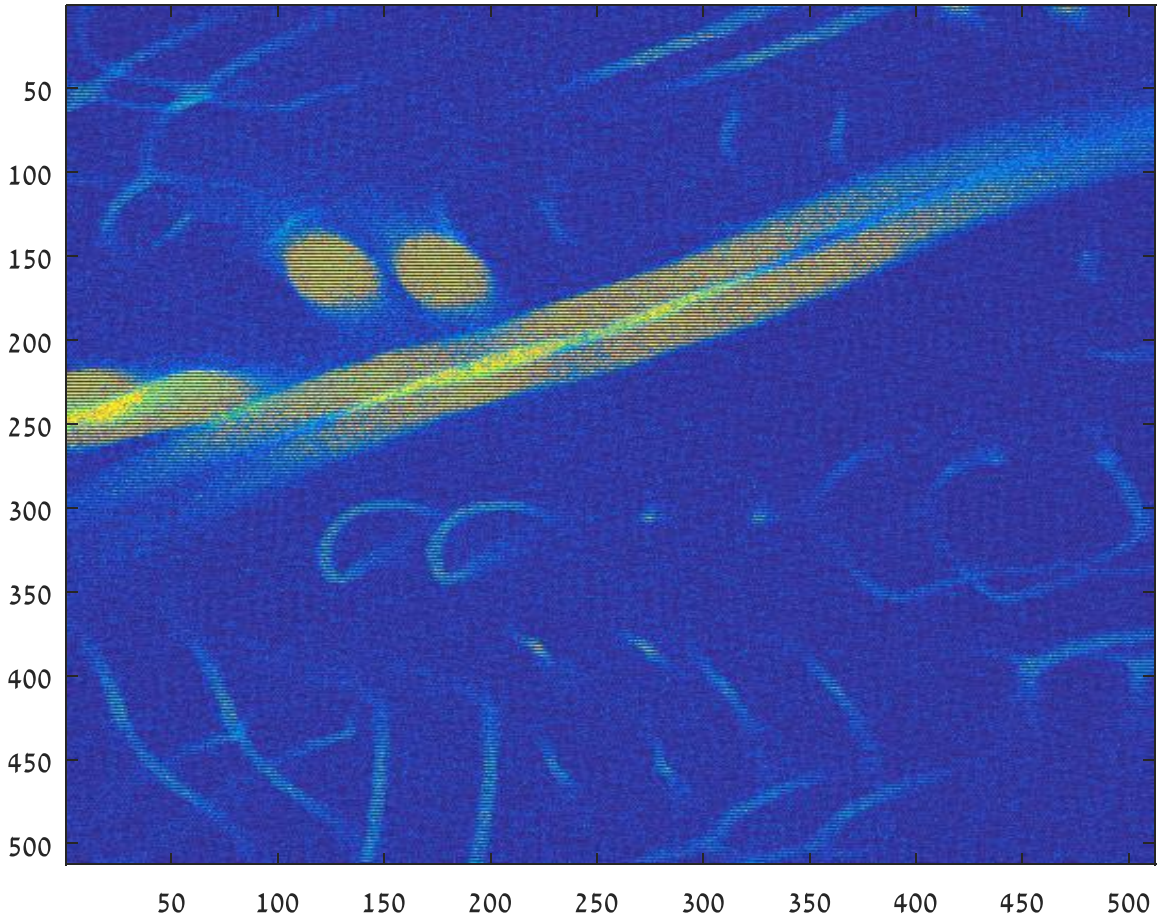


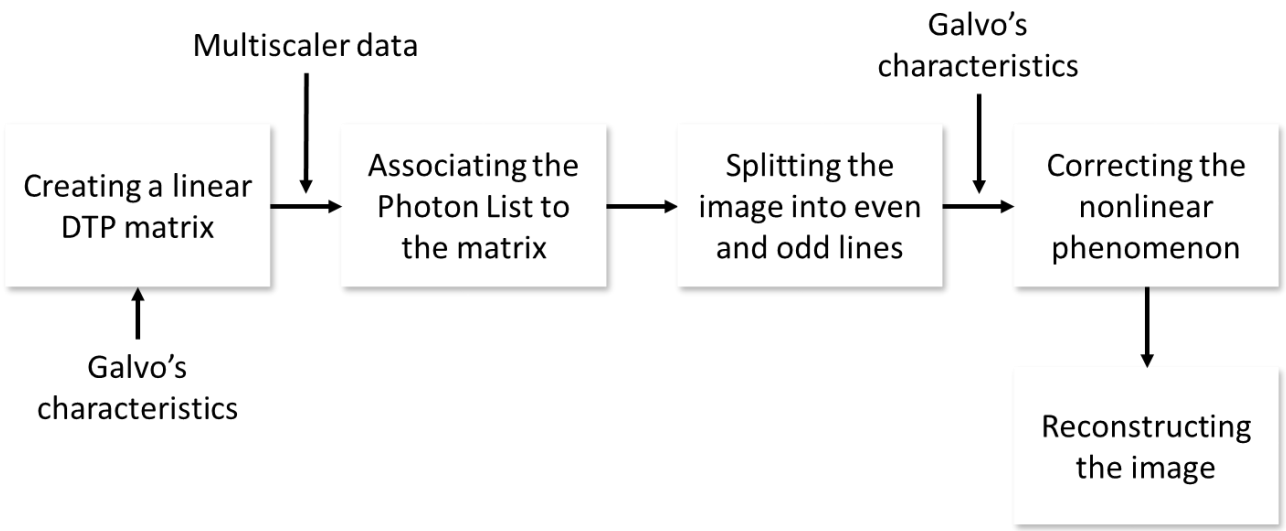
Image 1: *The duplication phenomenon caused by "Pixel Shifting".*

Suggested Solutions and Tools:

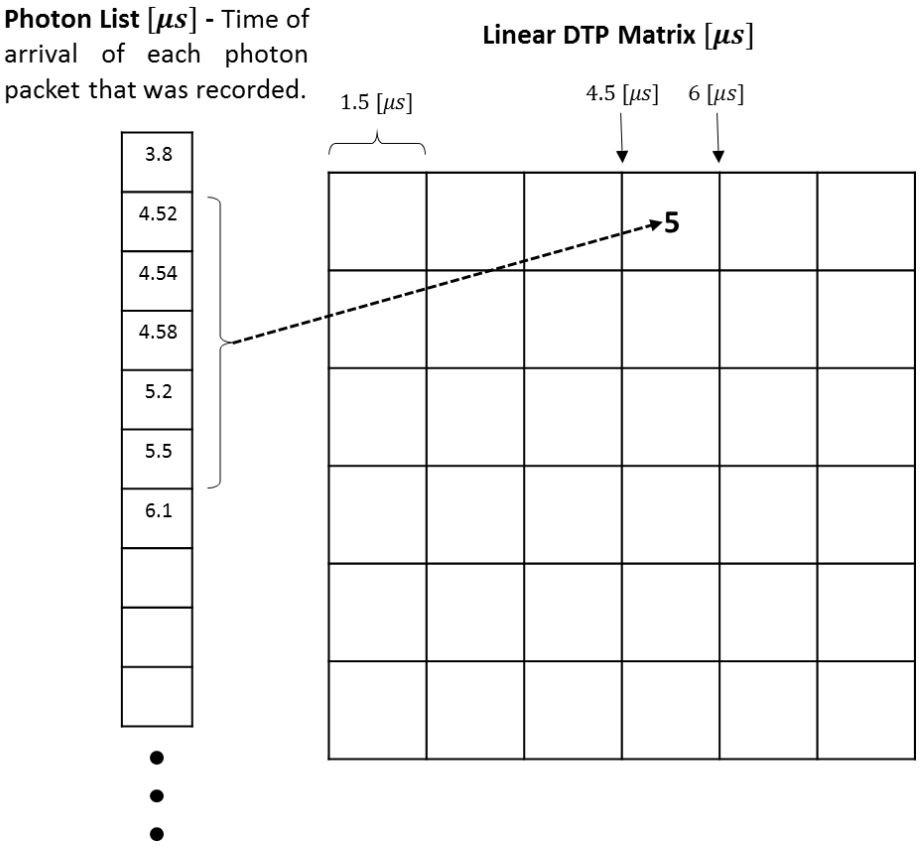
We suggest different tools which are helpful in different stages of image constructing. We will explain the setup which is relevant for every tool. Eventually, all tools are accessible together in one graphical user interface.

Image Correction Algorithm for Image Construction out of Multiscalar Data:

The algorithm assumes we know where the frame started.

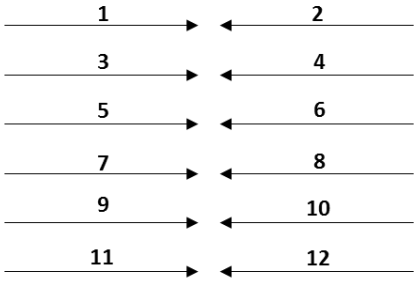


1. **Creating a linear DTP matrix:** The first stage is to build a $n \times n$ linear DTP matrix. Using the mirror's frequency data it is easy to calculate the time the fast mirror needs in order to scan a row. We divide this time frame linearly to each pixel in the row. The user can choose the discretization level by choosing n.
2. **Associating the photon list to the matrix:** From the Multiscalar data we can create a list of time of arrivals for the different photon packets. Using this "Photon List" we can associate each photon packet to a pixel providing us a "Linear Image" built under full linear approximation.



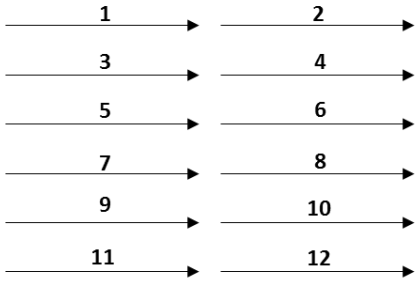
Drawing 3: *The DTP linear matrix being constructed according to the Photon List that contains time for arrival of each photon packet that has been recorded. If for example 5 photon packet were recorded during the time gap of the 4th pixel, it will now contain the data "5".*

3. **Splitting the linear image:**
    a. **Option 1:** Splitting the linear image into two different images – one for odd and one for even lines in order to correct each image separately.



Drawing 4: *Splitting the image into two different images, one for odd and one for even lines. The direction of the rows is opposite to one another.*

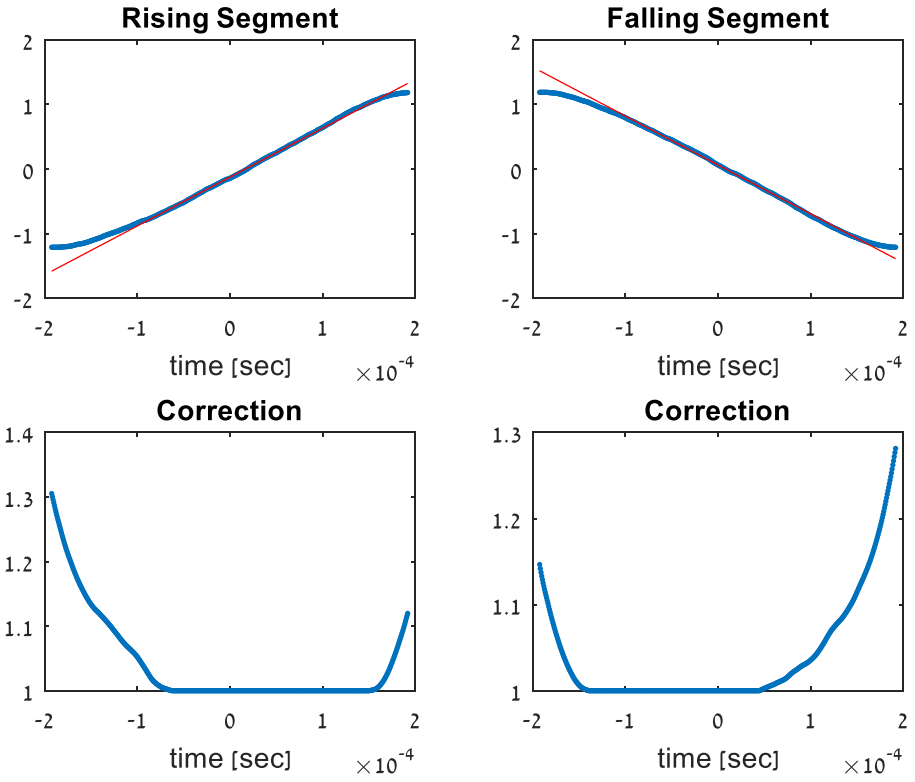
b. **Option 2:** If the same raster was being imaged few times back and forward we can take odd and even lines from different frames to create "one direction" image – only rise or fall voltage signal. Using this option allows us to correct both even and odd lines the same way.



Drawing 5: *Splitting the image into two different images, one for odd and one for even lines. The direction of the rows is now the same since we used different frame's data for each group.*

4. **Correcting the nonlinear phenomenon:** The correction stage is based over the assumption that for part of the travel, the mirror does behave linearly:
   a. **Creating a "True Location Vector" (TLV):** The mirrors can provide a voltage measurement during its operation. The voltage represents the mirror's angle which determines the location of the scanning. Using Oscilloscope we can measure the voltage and extrapolate a 512 measured points of voltage. Each point of each pixel. If the mirrors where traveling linearly we would expect these measurements to fit a straight line. As seen in image____ the measurements do not fit a straight line and as we approach the edges of the row they are less linear.
   b. **Creating a "Straight Line Vector" (SLV):** In order to create the straight line we choose the middle point of the measurements. As said, in this point the mirror had a constant speed. Out of this point, and other points in its surroundings, we can extrapolate a straight line (see drawing___).
   c. **Creating a "Correction Vector" (CV):** Division between the TLV and the SLV provide a "Correction Vector". Since around the middle point the values approaching zero, the division might diverge. As a result we determine the values to be 1 after a certain point (as seen in drawing ___).
   d. **Changing the Pixels Location:** Each pixel has a correction factor in the CV. We divide the location value of each pixel by the correction factor in order to determine the new location of the data. In order to prevent high intensity in the borders of the frame, if multipole pixels are shifted to the same pixel, only the average value would be added to the new pixel location. During this process the image is getting smaller with less pixels but we don't lose data since any photon packet that been recorded has an influence on the final image.

Beside for the actual shifting process, whole stages can be done offline. Using the voltage characteristics of different frequencies of the mirrors, we built a library of vectors so users should only input the rate they were using.

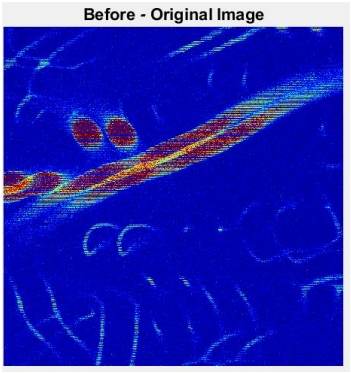


Drawing 6: *Upper part: in blue, the extrapolate measurements of the mirror's voltage. In red: the closest straight line. Lower part: The "Correction Vector", a division between the actual extrapolate measurements to the straight line approximation points.*

5. **Reconstructing the Image:** After both even and odd lines had been corrected we combined the two images together in order to create the new corrected image.

Pixel Shift Image Processing Algorithm:

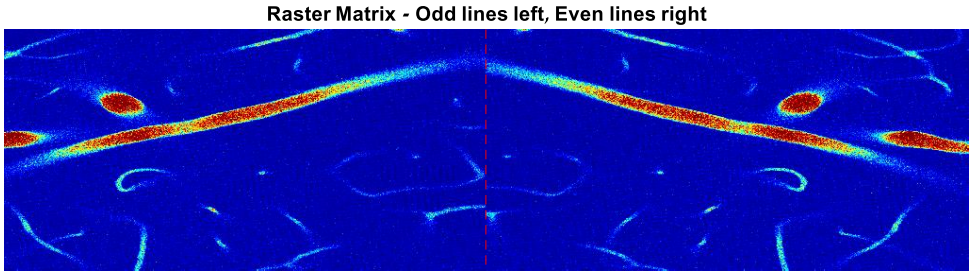
As explained before, one of the main issues in reconstructing an image is a pixel delay or shift caused by the response time of the Galvo X-Y system and the non-uniformity of responses in the forward\backward scan scanning directions. This issue is even more severe when the non-linearity is taken on top of it and results in a "doubling" effect in the produced image as explained before. To deal with this, a Semi Automatic Pixel-shift detector was implemented.



Let's take a step-by-step look at this:

We start with a sample image who exhibits the problematic issues, shown on the right.

We show the Raster scanned matrix which decouples the odd and even lines. The variance between the forward and backward scan can already be noticed



So next thing, we decided to find a way to automatically detect the shift differences between the odd and even lines and then fix by shifting them separately. The outline of the algorithm is:

Semi-Auto-Fix-Shift Algorithm

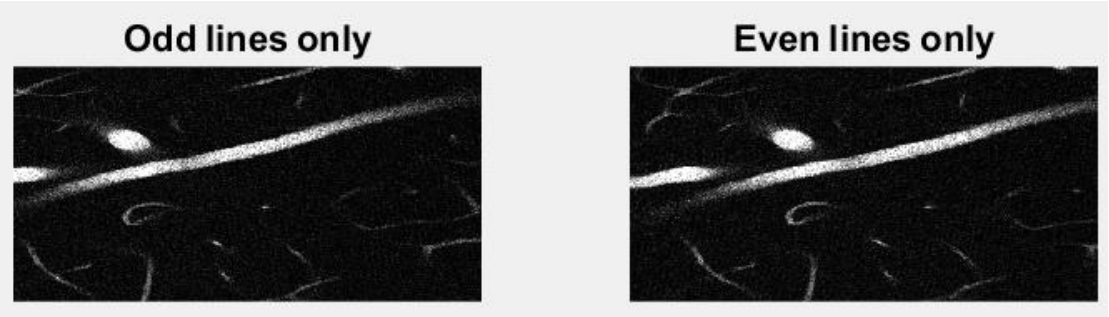
Input: Original distorted Image, window parameters

Output: Fixed Image and shift parameters found

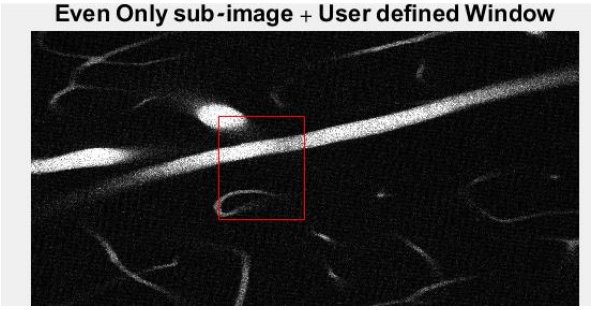
1. **Split** - Take Original Image and split to 2 Odd lines and Even lines only sub-images
2. **Window -** Open a smaller window (red rectangle demonstrated below) defined by the user window parameters in the "Even Only" sub-image. By default, we open a window at the center of the image with size of about an eighth of the image.
3. **Matching** - Use image processing methods - Cross-Correlation or SSD (sum of square differences) to find the best match for the smaller window (from the "Even only" sub-image) in the "Odd-only" sub-image. Find the shift between the user window coordinates and the found match coordinates – this is the effective so-called "pixel shift".

4. **Shift & Fix** – perform a raster shift (like a snake) with the amount of pixel shift found to obtain a fixed image. Demonstration:

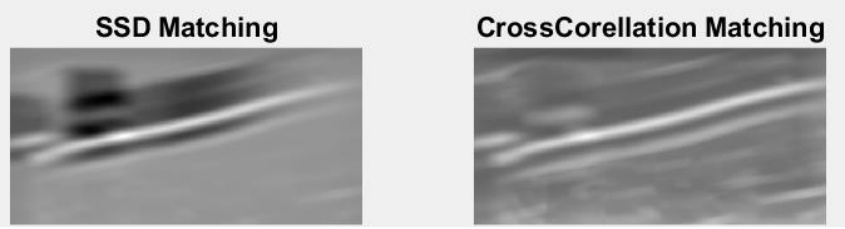
1. **Split**



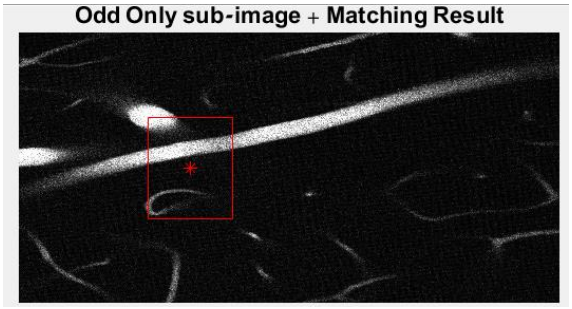
2. **Window**

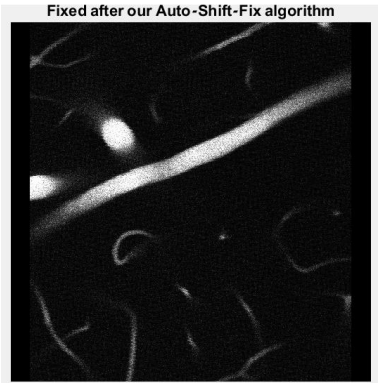


3. **Matching**



4. **Shift & Fix**

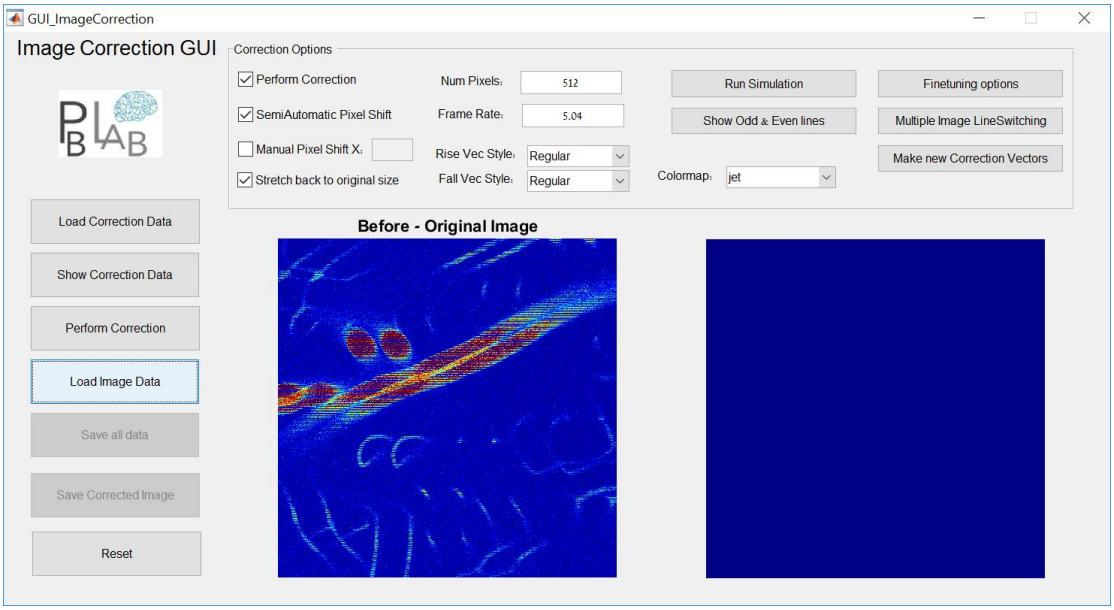




Shifts found were: xshift = 53, yshift = -1

GUI guides

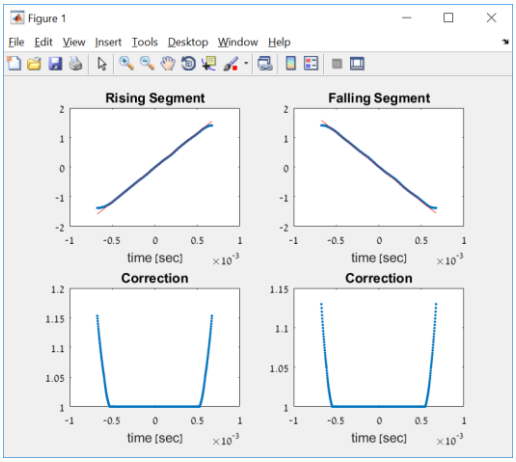
Image Correction GUI



Left Button Panel:

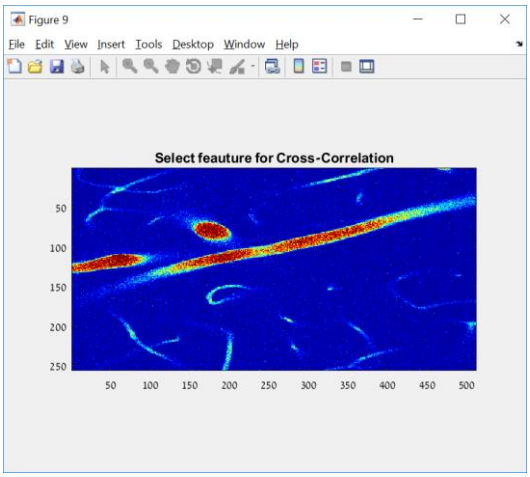
Load Correction – Open a pre-saved *.mat file containing the data needed for the correction steps. These matrices are created and saved in the Scanning Mirror Function Creator GUI.

Show Correction Data – Opens a new figure with the scanning functions, linear functions and corresponding correction vectors both for rise (forward scan) and fall (backward scan):

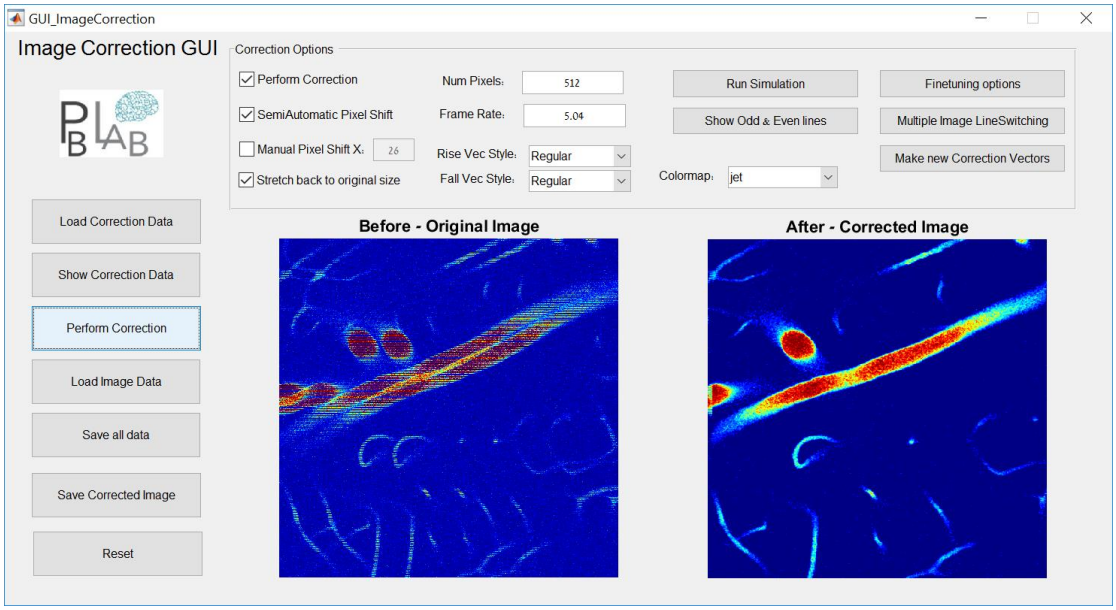


Load Image Data – allows the user to select a single image file

Perform Correction – Initiates a correction procedure. First a Pixel Shift correction is performed either manually (with the user input in the correction options tab) or semi-automatically using cross-correlation techniques. If the semi-auto option is invoked, then a window of only even lines will open and the user is asked to select a feature for cross-correlation:



After the feature is selected, the Algorithm will find the best fit, deduce the pixel-shift needed for correction and performs the shift on the image. Then the pixel-shifted image undergoes correction through the correction vectors selected by the user and an output corrected image is displayed on the right:



Save all date – allows the user to save all images, parameters and correction vectors to a Matlab matrix for futher use.

Save corrected image – Saves the corrected image in an image format to a file selected by the user.

Reset – Restarts the GUI and cleans all data.

Correction Options

Perform Correction - Check if Correction using the input correction vectors is wanted

SemiAutomatic Pixel Shift – Check if the user wants to invoke the process which performs cross-corrleation between odd and even lines and finds the best pixel shift match needed. This shift will be displayed in the textbox next to the manual shift option.

Manual Pixel Shift X – lets the user input the pixel shift manually.

Stretch back to original size – After the correction is done, the image is narrower in size. This option stretches back the image to the original size.

NumPixels – number of pixels in each dimension (input optional)

Frame Rate – Scanner frame rate (input optional)

Rise\Fall Vec Style – allows the user to choose what correction vectors will be used in the correction process, the options are:

Regular – perform threshold & smooth fixes to the calculated correction

Symmetric – Take Rise vector and make a mirror symmetric Fall vector out of it for both rise and fall

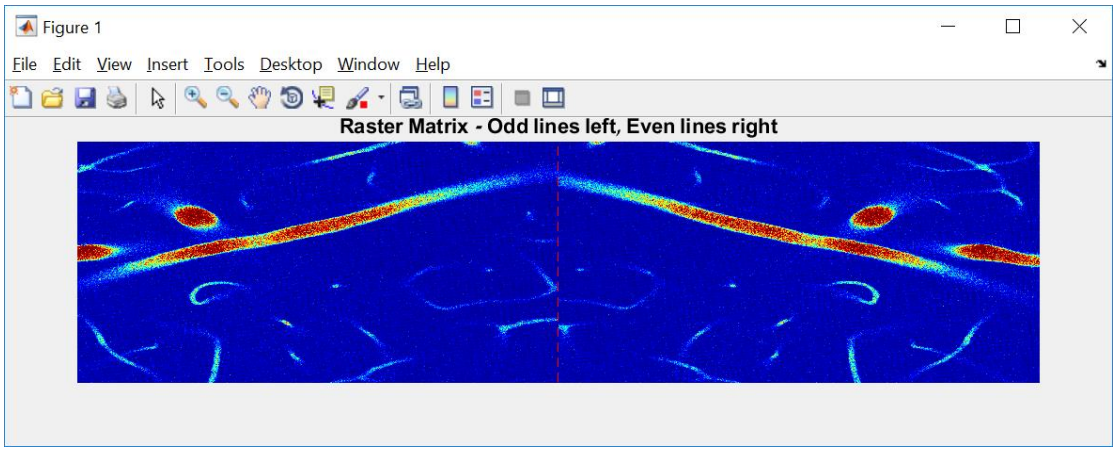
Rise+Fall – Take right side from rise and left side from fall for both rise & fall vectors

Raw – Raw correction calculations with no fixes

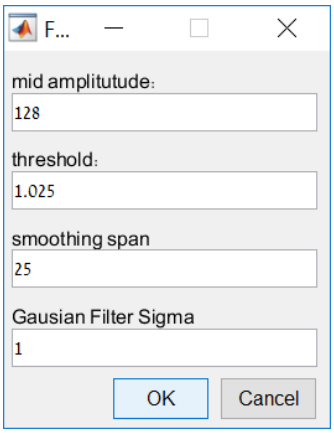
Colormap – select colormap for images shown. Default is jet. Parula and gray are allowed.

Run Simulation button – inputs a simulation image for testing

Show Odd & Even button – shows the true raster scan image, broken into odd lines (left side) and even lines (right side:



Fine-tuning options – opens a window with several options:



Mid amplitude – parameter that affects the fitting and creation of the linear vectors. The parameter decides how many pixels from each side from the middle are taken into account for the liear fit. Default is NumPixles/4.

Threshold – Decides from what correction vector value the output will be 1.

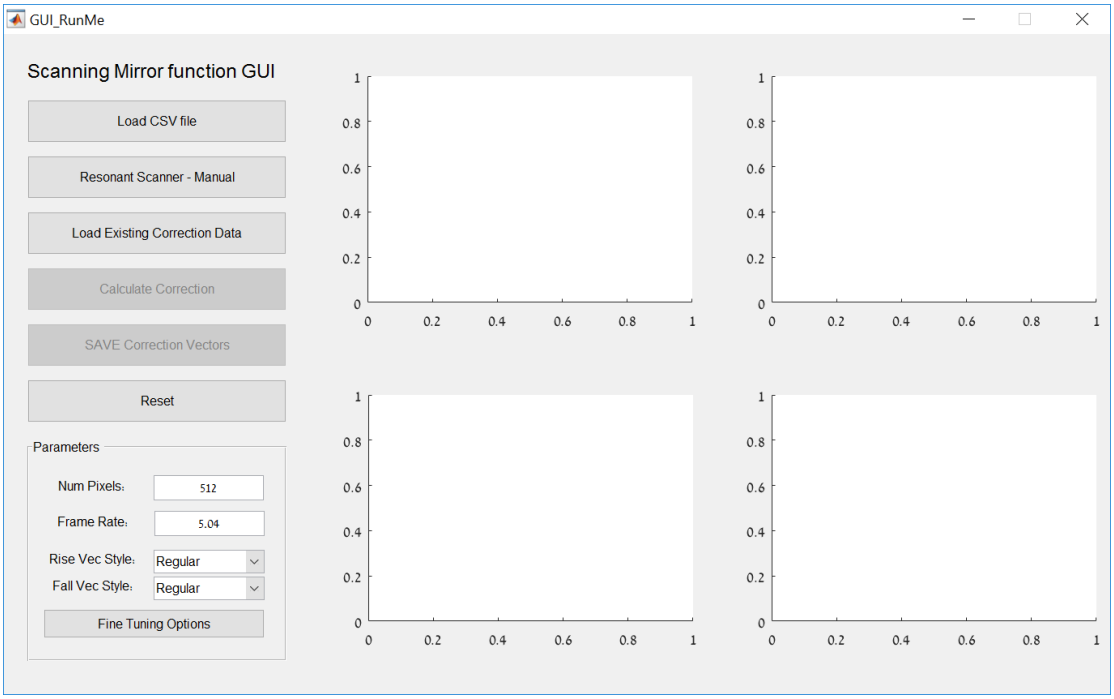
Smoothing Span – span of smoothening filter on the correction vectors.

Gaussian Filter Sigma – option for filtering the output image after all corrections are done.

Multiple Image Line Switching – For stacked images, another option of LineSwitching is available through this button

Make new correction vetors – Opens the Scanning function Creator GUI

Scanning Mirror function GUI

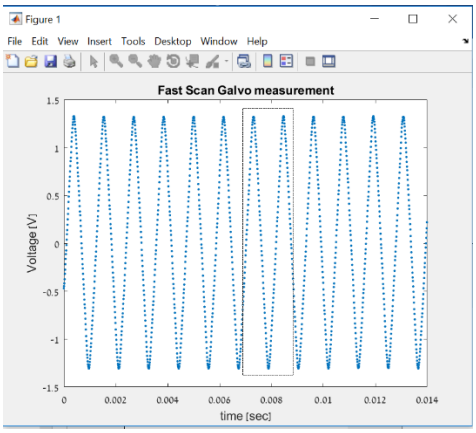


Load CSV file – user inputs a CSV file of a scanner. The waveform is

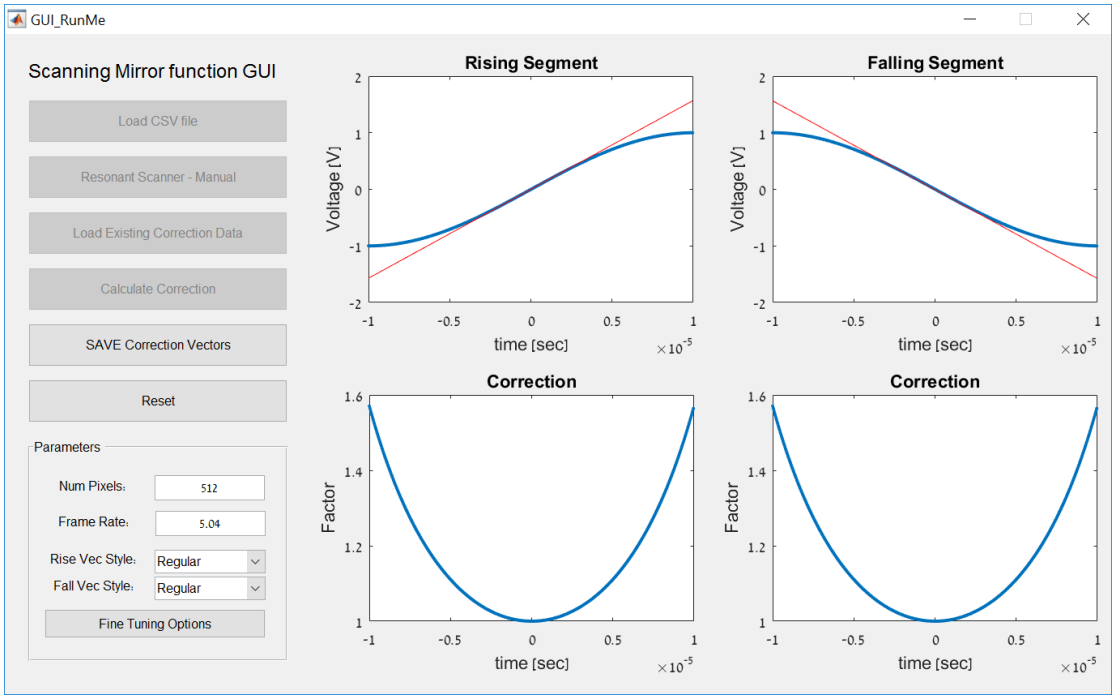
Resonant Scanner – Manual – If a fast resonant scanner is used, the there is no need for a CSV file and an option to simulate a sine wave of a given frequency is given.

Load Existing Correction Data – allows loading existing data for review

Calculate Correction – first, this opens a figure where the user is asked to select a rectangle containing two maximas and one minima



Then, the data for the Rise and Fall vectors is extracted, and a linear fit is performed. These are shown in the 2 upper main figures. Then the correction vector is calculated for vothe rise and fall vectors and shown in the 2 lower main figures.



Save Correction Vectors - saves the correction vectors to a *.mat file

Reset - clears all and resets the GUI

For Parameters inputs, see explanations in Image Correction GUI

Conclusion and Future Work

The original purpose of the project was to create full three-dimensional image. Since few crucial components of the system weren't available to us we finally focused our work in creating only two-dimensional image. The lab personal where able to create that kind of image but with various problems which affected their research. Consequently we dedicated our work to explore those problems and to try and find practical solutions to it.

During the project we found out few times that our understanding of a certain problem was wrong and we pursuit a false solution. This ongoing process thought us a lot about the nature of academic research in general and experimental research in particular - The experimental system is so complicated and vast that often understanding the effect of each part over the other which eventually causing a problem, is a tricky mission.

Although the different solutions we suggested are tailored to tackle a specific problem for the specific setup of the system, we tried to build it as robust as we can so key features would be useable in the future as well. The main trend in the lab is to replace the current imaging software with new system which is being developed by the lab personal. This system would use transformation of location data of each photon to digital data and attaching it to the measurements. Using this data, it would be easier to build the image photon by photon without the commercial imaging software. Our tools would help in creating an image in the final path that is now being developed. More, our work helped the lab personal understanding the different problems so they could develop the new system better.

References

[1] Michael J. Sanderson, "Acquisition of Multiple Real-Time Images for Laser Scanning Microscopy", Microscopy and Analysis, July 2004

[2] Peng Xi, Yujia Liu and Qiushi Ren Peng Xi, " Scanning and Image Reconstruction Techniques in Confocal Laser Scanning Microscopy", "Laser Scanning, Theory and Applications", Chapter 27, DOI: 10.5772/14545, April 2011

[3] Quoc -Thang Nguyen, Philbert S. Tsai, David Kleinfeld, "MPScope: A versatile software suite for multiphoton microscopy", Journal of Neuroscience Methods, December 2005

[4] Pologruto TA1, Sabatini BL, Svoboda K, "ScanImage: flexible software for operating laser scanning microscopes", Biomed Eng Online, May 2003

[5] MicroscopyU, Nikon – "Resonant Scanning in Laser Confocal Microscopy", http://www.microscopyu.com/tutorials/resonant-scanning-in-laser-confocal-microscopy