UNIVERSIDADE D
COIMBRA

Hugo Miguel Virtuoso Simões Sebastião

Development of device control and real-time data analysis methods for a nonlinear multiphoton fluorescence lifetime imaging microscopy (MP-FLIM) system

Thesis submitted to the University of Coimbra in fulfilment of the requirements of the master's degree in Biomedical Engineering under the scientific supervision of Dr. rer. Jana B. Nieder (INL), and António Miguel Lino Santos Morgado, Ph.D. (U Coimbra).

February, 2022

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

University of Coimbra

Faculty of Sciences and technology

Department of Physics

Hugo Miguel Virtuoso Simões Sebastião

Development of Device Control and Real-Time Data Analysis Methods for a Nonlinear Multiphoton Fluorescence Lifetime Imaging Microscopy (MP-FLIM) System

Supervisors:

Dr. rer. Jana B. Nieder, Research Group Leader of the Ultrafast Bio- and Nanophotonics group of the INL - International Iberian Nanotechnology Laboratory, Braga, Portugal

António Miguel Lino Santos Morgado, Ph.D., Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC)

Coimbra, February 2022

This work was developed at

INL - International Iberian Nanotechnology Laboratory

in the frame of the project: ExtreMED – Extreme Ultrashort Pulses for Advanced Medical Applications and Diagnostics with grant number: NORTE-01-0247-FEDER-04593, funded by the PO Norte; FCT. via the PROJETOS DE I&DT COPROMOÇÃO – INTERNATIONAL PARTNERSHIPS program.

A collaboration between:

- SPHERE ULTRAFAST PHOTONICS, S.A.
- International Iberian Nanotechnology Laboratory (INL)
- University of Porto
- University of Texas at Austin (UTA)

Acknowledgments

Firstly, I would like to thank my supervisors, Dr Jana Nieder and Professor Miguel Morgado, for all the teaching, support, and advising throughout my endeavours to accomplish this thesis. I would like to thank the ExtreMED team for all the teamwork and support. An honourable mention is due to Dr Christian Maibohm, for all the hours spent teaching and guiding me.

I would like to express my admiration for all the work that International Iberian Nanotechnology Laboratory has been developing.

This thesis would not be possible without the help and support of many people. To João Martins, Filipe Camarneiro, Leonor Ribeiro, along with all the UBNP team, thank you. Thanks are due to all my professors from the University of Coimbra, for providing me with the knowledge and background needed to elaborate the present thesis. Moreover, I would like to thank the Sphere Ultrafast Photonics for developing and providing crucial materials for the development of the microscope, on which I have worked during my thesis studies.

I would like to mention my degree fellows, with whom I shared my academic journey. I would like to thank all my family and friends for always being at my side during this journey. I thank my mother Cristina for the support and preoccupation during my life which allowed me to reach this point. I also want to mention my father Helder by providing support and help all these years. I thank my brother and sister, Afonso, and Catarina, for all the fun and laughs which allowed me to overcome some difficult times. At last, I would like to express my great gratitude to Gabriela Lapa for all the support, unconditional love, and never-ending encouragement.

Resumo

Nos últimos anos a microscopia multifotónica emergiu como uma promissora área de investigação, servindo como uma ferramenta pré-clínica para observação detalhada do metabolismo e comportamento celular. A microscopia multifotónica permite criar imagens com profundidade no objeto, devido à sua capacidade de penetrar nos tecidos, reduzindo a fototoxicidade. Isto torna a microscopia multifotónica adequada para observações a longo prazo em tecidos vivos.

A microscopia multifotónica pode ser aplicada em células com componentes fluorescentes desconhecidas ou com fluoróforos aplicados. Cada fluoróforo é excitado pela luz com um comprimento de onda específico.

Recentemente, as vantagens dos lasers de ultra-banda promoveram o desenvolvimento da tecnologia SyncRGB-FLIM. Esta tecnologia incorpora um procedimento de imagem integrado, sendo muito eficiente na distinção de diferentes áreas pela distribuição dos fluoróforos.

Esta tese descreve o desenvolvimento de um software para o controlo de dispositivos e análise de dados, que permite o uso do microscópio SyncRGB-FLIM. O controlo de dispositivos e a análise de dados permite a criação de imagens em tempo real de células, sendo capaz de distinguir um máximo de três fluoróforos presentes.

O software foi testado em laboratório, onde os primeiros dados de imagem foram recolhidos e analisados em amostras celulares em tempo real. Demonstrou-se que dois ou mais fluoróforos podem ser distinguidos apenas pela sua fluorescência, utilizando apenas um único detetor.

Palavras-chave: Microscopia Multifotónica Fluorescente, Análise de Dados, Controlo de Dispositivos, SyncRGB-FLIM.

Abstract

Over the past years, multiphoton microscopy has emerged as a powerful research stream and pre-clinical tool for observing detailed cellular and metabolic behaviour. Multiphoton microscopy allows deep imaging, due to its ability to penetrate tissues while inducing reduced phototoxicity. This makes multiphoton microscopy suitable for long-term observations in living tissues.

Multiphoton microscopy can be applied to label-free cells with natural fluorescent components or applied in cells with administrated fluorophores. Each fluorophore is excited by a specific wavelength range.

Recently, advantages in ultrabroadband few-cycle lasers have promoted the development of SyncRGB-FLIM technology. This technology has an integrated imaging procedure and takes advantage on distinguishing different areas by their fluorescence lifetime distribution. This is possible due to the broad spectral profile of the few-cycle lasers, exciting fluorophores with different wavelength excitation ranges.

This thesis describes the development of dedicated device control and data analysis software that enables the use of SyncRGB-FLIM microscopy. The device control and data analysis allow the creation of real-time imaging of cells labelled with up to three fluorophores.

The software was tested in the laboratory, where first image data was collected and analyzed on multi colour-stained cellular samples in real-time. It is shown that two or more chromophores can be distinguished merely by their fluorescence using just a single detector.

Keywords: Fluorescence Multiphoton Microscopy, Data Analysis, Device Control, SyncRGB-FLIM.

Acronyms and Abbreviations

- APD – Avalanche Photo Detectors
- a.u. – Arbitrary Units
- d-scan – Dispersion Scan
- DNA – Deoxyribonucleic Acid
- ExtreMED – Extreme Ultrashort Pulses for Advanced Medical Applications and Diagnostics
- FCT – Fundação para a Ciência e a Tecnologia
- FLIM – Fluorescence Lifetime Imaging Microscopy
- FP – Fluorescent Proteins
- fs – Femtosecond
- GmbH – Gesellschaft mit beschränkter Haftung
- INL – International Iberian Nanotechnology Laboratory
- IRF – Instrument Response Function
- mm – Millimeters
- MRS – Magnetic Resonance Spectroscopy
- ms - Millisecond
- mw – Milliwatts
- NADH – Nicotinamide Adenine Dinucleotide + Hydrogen
- NADPH – Nicotinamide Adenine Dinucleotide Phosphate
- ND – Neutral Density filter
- NI – National Instruments
- NIR – Near-InfraRed
- nm – Nanometer
- PET – Positron Emission Tomography
- pH – potential of Hydrogen
- PMT – Photomultiplier Tubes
- PNG – Portable Network Graphic format
- ps – Picoseconds
- ROI – Regions of interest
- ROS – Reactive Oxygen Species
- SEM – Scanning Electron Microscopy
- SHG – Second Harmonic Generation
- S.I. - International System of Units
- SP – Short Pass filter
- SyncRGB-FLIM – Synchronous fluorescence imaging of red, green, and blue dyes enabled by ultra-broadband few-cycle laser excitation and fluorescence lifetime detection

- TCSPC – Time-Correlated Single Photon Counting
- TEM – Transmission Electron Microscopy
- Ti:Sa – Titanium-Sapphire laser
- TTL - Transistor-Transistor Logic
- TTTR – Time-Tagged Time-Resolved
- USB – Universal Serial Bus
- V - Volts
- µm – Micrometer
- 3D – 3 Dimensions

List of Figures

List of Tables

Contents

1 Introduction

This initial chapter presents the motivation behind the study (Section 1.1), followed by a basic description of the area of study (Section 1.2), presents the ExtreMed team (Section 1.3), highlights the main goals of the project (Section 1.4), and shows the outline of the remainder (Section 1.5).

1.1 Motivation

Nonlinear microscopy has been widely used in medical and biological fields, mainly due to its ability to provide images of cells without the use of invasive techniques. Nonlinear microscopy allows the observation of the interactions between proteins, which are the essential components that govern cellular life. Notably, it allows the observation and deeper understanding of cellular processes such as DNA replication, translation, secretion, transcription, and metabolism. (Phizicky and Fields, 1995).

In these applications, it is of extreme importance not only to extend the life of the samples but also to improve image resolution and depth, which ultimately results in more reliable information.

The ExtreMed team aims to develop, use, and evaluate the potential benefits of ultrafast laser sources for nonlinear microscopy.

The software for the nonlinear microscope acquired by the International Iberian Nanotechnology Laboratory (INL) was completely developed by me, following the instructions and requirements of the ExtreMed team. The development of a nonlinear microscope using ultrafast laser sources unlocks the possibility of real-time protein-protein interaction studies, where its single-scan procedures translate into reduced laser exposure of the sample, ensuing more photoprotective conditions.

1.2 Area of study

Optical microscopy is spread into multiple branches, being the cornerstone of the first step of the scientific method in many areas of knowledge. Therefore, microscopy improvements necessarily translate into relevant scientific advances (Dawe et al., 2006).

In medicine, some types of microscopies have an important role in the identification of abnormal cells collected through biopsies, while in biology these microscopies are useful in magnifying small samples (Mullen et al., 2016; Yoshida et al., 2007).

Optical imaging is currently the most used observation technique in life sciences. Around 80% of modern biomedical microscopy relies on light microscopy, despite the recent and vast improvements in Scanning Electron Microscopy (SEM) and Transmission Electron Microscopy (TEM) (Hell, 2007). Light microscopy is still more popular than SEM and TEM because these techniques cannot match the ability of light microscopy to deal with different focuses within three-dimension objects, while light microscopy is easier to use, requires less sample preparation, and has lower costs (Hell, 2007).

Although very old, light microscopy has not yet achieved its full potential in biomedical imaging and new types of this technique are still being developed with biomedical applications (Dawe et al., 2006). Generally, light microscopy magnifies a sample by illuminating it or causing it to emit light in visible, near-ultraviolet or near infra-red ranges, with the magnification obtained by passing the light through a group of lenses (Dawe et al., 2006).

The interest of the scientific community in optical microscopy has led to constant research and development of new techniques, amongst which stands out fluorescence microscopy. This powerful technique combines the advantages of being a type of light microscopy with the ability to detect cellular constituents, such as proteins (Hell, 2007).

Luminescence is used to characterize any emission of light from a substance. Luminescence takes place when a material has at least one electronic excited state, from which an excited electron can relax back to its ground state while delivering a photon with the released energy (Murthy and Virk, 2014). There are two notable types of luminescence processes: phosphorescence, which occurs in a time scale of milliseconds, and fluorescence, which occurs in the nanosecond range (Görlitz, 2018).

In phosphorescence processes, the excited electron in the triplet state has the same spin orientation as the ground state. Transitions between electronic levels with the same spin are not allowed, which can be overcome by the perturbation of spin-allowed electronic transitions using intensity borrowing (Baryshnikov et al., 2017). Fluorescence processes happen due to the proprieties of fluorophores, where the excited electron has the opposite spin

to the electron in the ground state, allowing the transitions to happen, according to Pauli's Principle (Jarvis and Sweetman, 2015).

The scope of the present thesis is on fluorescence processes.

There are several fluorescence-based microscopy techniques, such as confocal microscopy, wide-field microscopy, multiphoton microscopy, total internal reflection microscopy (Görlitz, 2018). Broadly speaking, these techniques enable molecule-specific imaging by using antibodies, genetically induced fluorescence, and other types of staining (Görlitz, 2018). Despite the recent technical advances, most of these techniques still present some limitations in terms of spatial resolution due to the diffraction of light in materials (Huang et al., 2009). Usually, these limitations are overcome by super-resolution techniques. Multiphoton fluorescence microscopy is one of the most used techniques in practice (Huang et al., 2009).

The multiphoton fluorescence microscopy imaging is limited to the focal plane of the objective lens. Therefore, it has inherent optical sectioning. Changing the focus position in the object allows imaging at different depths, making possible three-dimensional imaging by combining the images obtained at different depths (Miller et al., 2017).

The fluorophores that can be used in multiphoton microscopy depend on the spectrum of the laser since only fluorophores with excitation wavelengths that are inside the spectrum of that laser can be excited. Hence, when a laser has a broader spectrum, the list of fluorophores that can be excited increases.

Maibohm et al. (2019) compared the standard light source for multiphoton, which is the 70 fs Titanium - Sapphire laser (Ti:Sa), and an ultra-broadband 7 fs Ti:Sa laser. They observed that even although the 70 fs Ti:Sa laser allows 3D imaging, this laser is not optimized due to scattering and absorption events by the surrounding substances, which are wavelength-dependent. Also, they highlighted that, the ultra-broadband 7 fs Ti:Sa laser has an optimized spectral profile to match the absorption of various fluorophores, reaching 50% deeper inside the tissue, providing more reliable 3D imaging, while exciting fluorophores along with all visible spectra.

1.3 ExtreMed Team

The ExtreMed team is supervised by Jana Nieder and is also constituted by Christian Maibohm and Leonor Ribeiro. Christian Maibohm is an expert in microscopy, and his main task is to ensemble the microscope. Leonor Ribeiro has a deep working experience with different

microscopy techniques and is the main user of the microscope. I was integrated into the ExtreMed team, with the main task of the microscope software development.

1.4 Goals and Objectives

The ExtreMed project aims to develop a novel nonlinear multiphoton microscope setup, using an ultra-broadband femtosecond laser, optimized for deep tissue analysis and high throughput imaging of organoids and other 3D *in vitro* models. My main task in ExtreMed consisted of software development focused on device control and output data analysis. The software needed to be user-friendly, able to store all relevant information, and should allow direct visualization of the fluorescence lifetime results, with the option of direct feedback on the results when interacting with the software.

1.5 Thesis Outline

Besides this introduction, the present thesis is structured into 5 chapters.

Chapter 2 presents the state of the art on fluorescence-based microscopes and imaging using Fluorescence Lifetime Imaging Microscopy (FLIM). Femtosecond laser sources for multiphoton microscopy and ultrashort laser pulse characterization techniques are also discussed. This chapter concludes with the explanation of fluorescence microscopy in metabolic imaging. Chapter 3 discusses the devices of the microscope that are controllable and non-controllable by the software. Chapter 4 describes the control procedures of the controllable devices and the types of available analyses on the output data, with the discussion of the results, and underlining the relevance of the software in relation to other existing software. Chapter 5 highlights the achieved goals and the challenges overcome during the project development.

2 Fluorescence Microscopy

This chapter presents an overview of fluorescence microscopy (Section 2.1), namely its physical process (Subsection 2.1.1), fluorescence lifetime (Subsection 2.1.2), one-photon and multiphoton excitation processes (Subsection 2.1.3), fluorescence microscopy setups for one- and two-photon excitations (Subsection 2.1.4), and issues related with contrast (Subsection 2.1.5) and labelling and probes (Subsection 2.1.6). Section 2.2 presents Fluorescence Lifetime Imaging Microscopy (FLIM), Section 2.3 explains the femtosecond lasers for two-photon microscopy, Section 2.4 introduces secondary harmonic generation, Section 2.5 characterizes the broadband few-cycle laser pulses, and, finally, Section 2.6 highlights the importance of fluorescence multiphoton microscopy in metabolic imaging.

2.1 Overview

2.1.1 Physical process of fluorescence

Fluorophores are molecules that absorb photons with specific energy and emit photons of lower energy, which are slightly red-shifted, considering the electromagnetic spectrum (Murthy and Virk, 2014). This property makes fluorescence an effective technique, by allowing to observe the fluorescent objects by filtering out the excitation light with a dichroic mirror, or a Neutral Density (ND) filter, while not blocking the emitted fluorescence. The difference in energy, known as Stokes shift, is represented in Figure 2.1.

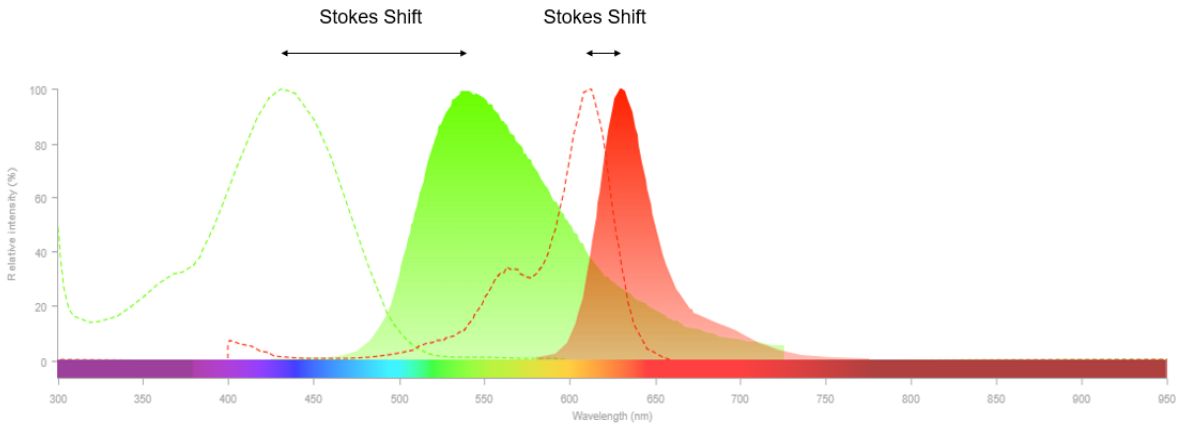

Figure 2.1: Fluorescence excitation (dashed lines) and emission (areas) spectra of different fluorescent compounds. The colour green corresponds to Alexa Fluor 430, and the colour red corresponds to Alexa Fluor 610. This image was created by the author using the spectraviewer software from the ThermoFisher website (https://www.thermofisher.com/order/fluorescence-spectraviewer#!/).

Fluorophores contain atoms that share several detached electrons in molecular orbitals. These detached electrons determine the efficiency of the fluorescent compound and the wavelengths of absorption and emission. The electronic ground state is typically at a singlet state, nominated as S_0.

The Jablonski diagram, exhibited in Figure 2.2, shows that, after a photon is absorbed, one of the electrons of the fluorophore is excited to another orbital, in the first excited state, S_1. In this state, the electron distribution around the atomic nuclei is altered compared to its distribution in the ground state. The electrons change their spatial distribution almost instantaneously (in femtoseconds), whilst the much heavier atomic nuclei move more slowly and take significantly more emission time. The excited electron is then relocated in a higher orbital, at a greater distance from the nuclei than before, resulting in a relaxation of the molecular structure, and setting in motion a vibration of the molecule to a new equilibrium position of atoms. The molecule vibrational energy rapidly dissipates in form of heat, from the collisions with the surrounding solvent, decaying into the vibrational ground state of the electronical excited state. Since the molecule energy is still higher than the energy in the ground state, the excited state is not stable for a long time. The molecule then returns to the ground state by emitting the excess energy as light, i.e., fluorescence. The decay to the ground state changes the electronic configuration around the nuclei, ending in a vibrationally excited state that then returns to the vibrational ground state. Non-radioactive decay may also occur when collisions with the solvents cause the decay of the excited state to the ground state, resulting in the absence of emission of fluorescence photons. For a more detailed description of the physical process of fluorescence see Dobrucki and Kubitscheck (2017).

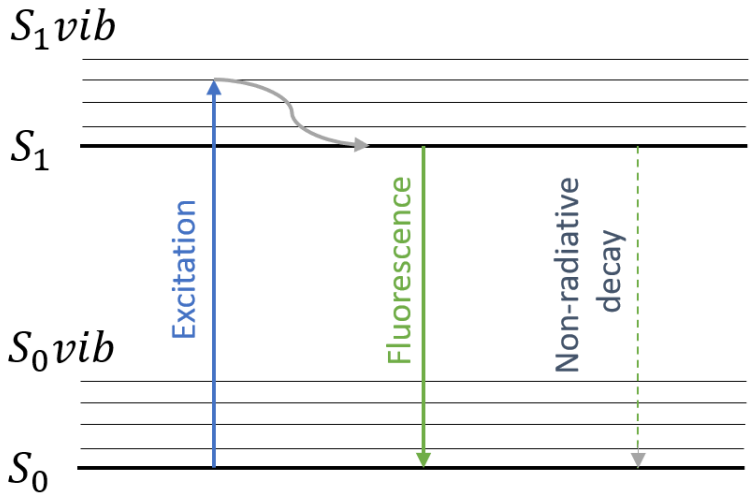

Figure 2.2: Jablonski diagram illustrating the transitions between ground and excited states for fluorescence and non-radioactive decay, with the vibration energy loss to the solvent represented by the zig-zag grey arrow. This figure was adapted by the author from Scientific Volume Imaging, (2021).

2.1.2 Fluorescence lifetime

The time spent in the excited state is designated as fluorescence lifetime, τ, which is characteristic of each specific fluorophore (Dobrucki and Kubitscheck, 2017).

Since fluorescence is an incoherent process, the release rate of photons is stochastically controlled by the population dynamics, which results in an exponential intensity decay, represented by Equation (2.1), where A_0 is the number of photons when $t = 0$, A_1 represents the number of photons that are still excited at a given time, t, and Γ is the decay rate (Görlitz, 2018).

$$A_1(t) = A_0 e^{-t \times \Gamma} \tag{2.1}$$

The characteristic fluorescence lifetime is determined by Equation (2.2) (Dobrucki and Kubitscheck, 2017).

$$\tau = \frac{1}{\Gamma} \tag{2.2}$$

Several surrounding factors affect the fluorescence lifetime, namely molecular binding, pH, and ion concentration (Dobrucki and Kubitscheck, 2017).

2.1.3 One-photon and multiphoton excitation processes

To excite a fluorophore molecule, it is necessary a certain amount of energy, which can come from one-photon or from multiphoton. In one-photon absorption, the excitation is usually done by ultraviolet or blue light, where one-photon has enough energy to excite the molecule. In the case of multiphoton, the excitation can be by two or three photons, usually by infrared light. This thesis focus on the two-photon case. Figure 2.3 illustrates the excitation energy from one-photon and infrared two-photon with the respective energy of emission.

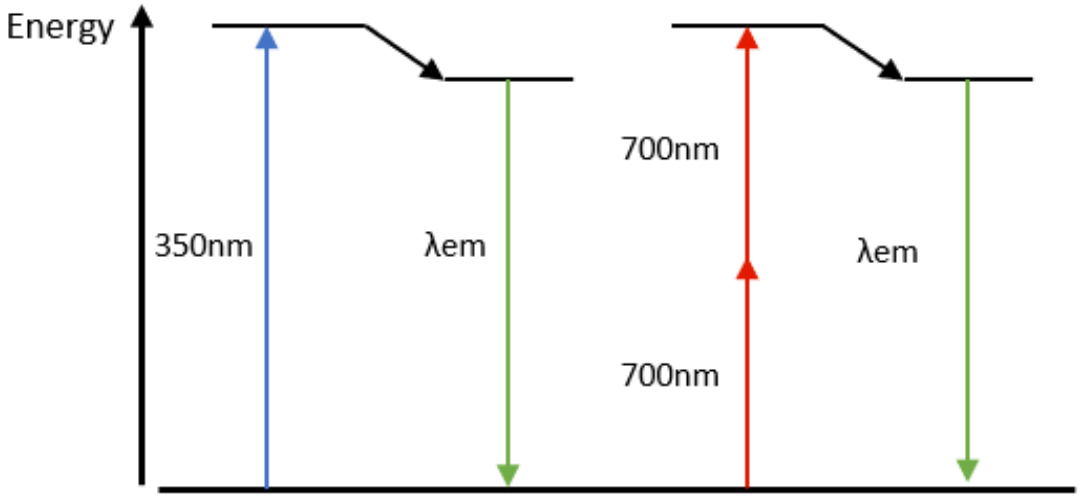

Figure 2.3: Excitation energy for 350 nm single-photon (single blue arrow) and two 700 nm infrared photons (two red arrows) with the respective energy of emission (green arrows). This figure was adapted by the author from Worbs (2006).

Two-photon absorption is typically achieved by a near-infrared (NIR) laser in which both photons are virtually absorbed at the same time. The molecule absorbs the first photon, stays in an intermediate state, and in a time frame shorter than 10^{-15} seconds, absorbs the second photon (Microscopy U, 2021). The second photon must be absorbed before the intermediate state falls back to the ground state. This requires a large density of photons to secure that the second photon interacts with the molecule while it is in the intermediate state. In these events, fluorophores are excited by two photons with lower energy, while having, typically, the same emission of one-photon absorption. If the absorption process is allowed, fluorophores that are excited by ultraviolet light (350 nm wavelength) can also be excited by two photons in the near-infrared (700nm wavelength), which can only occur if the photon density in the sample is sufficiently high.

The required photon density increases directly with the number of photons that are necessary to create the absorption event (single, double, or triple absorption). To create two-photon events, the photon density needs to be one million times more than in the case of one-photon absorption. Therefore, a high density of photons is necessary, which is only achieved with an extremely high-power laser (Fujimoto and Farkas, 2009).

The high-power laser can be accomplished by having the laser in continuous wave mode, which delivers the intensity required to increase the density of photons. However, continuous lasers require an enormous amount of power and tend to photobleach the sample (Becker & Hickl, 2021). Pulsed lasers avoid the use of such an amount of energy, by having high peak power while keeping the average power low, achieving the same results (Becker & Hickl, 2021).

In theory, the emission spectrum of one-photon and two-photon absorptions of a dye should be the same. However, in practice, it is observed a blue shift of the spectrum in the two-photon comparatively to one-photon absorption, which is explained by the two-photon absorption promoting higher energy photons (Bestvater et al., 2002).

The difference in generated fluorescence created in one-photon and multiphoton absorption events is represented in Figure 2.4. In the one-photon absorption, the photon has enough energy to excite the molecules, exciting the sample along the path of the laser. However, the focused and defocused sides of the cone create out-of-focus fluorescence. In the two-photons case, absorption occurs only in the focal plane of the laser, which is the only location in the laser path where exists enough photon density to create the events. In this case, the z-plane is already defined by the focal point of the laser (Becker & Hickl, 2021). Moreover, in the one-photon process, fluorophores are excited at nearly all depths, which leads to increased photobleaching relatively to multiphoton excitation. (Becker & Hickl, 2021). Photobleaching is the photochemical change of the fluorophore molecules in such a way that they come permanently unable to fluoresce, increasing each time the fluorophore is excited (Becker & Hickl, 2021).

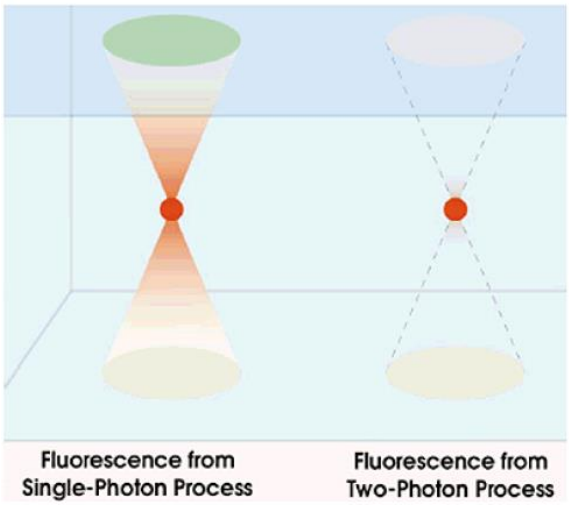

Figure 2.4: Fluorescence below and above the focal plane for the single-photon process (left), resulting in considerable out-of-focus light, and fluorescence for the two-photon process (right) that originates only from the focal plane. Source: Worbs (2006).

Although three-photon absorption has been less used in practice than two-photon, it has special uses for some types of microscopes, providing better resolution in specific situations (Hell et al., 1996).

2.1.4 Fluorescence microscopy setups for one- and two-photon excitations

Until now it has been discussed what happens in a single absorption point. To characterize a three-dimensional object, the absorption point needs to move throughout that object. When scanning the object with a laser, there is the option of beam scanning or sample scanning. When exerting both options there is always out-of-focus fluorescence in one-photon excitation, which results from the excitation of the object along the z-axis (see Figure 2.5). Confocal microscopy aims at avoiding one-photon out-of-focus fluorescence, i.e., one of the basic properties of this technique is the ability to use a confocal aperture, thereby enhancing the signal-to-noise by avoiding the detection of out-of-focus fluorescence (Cheng, 2006).

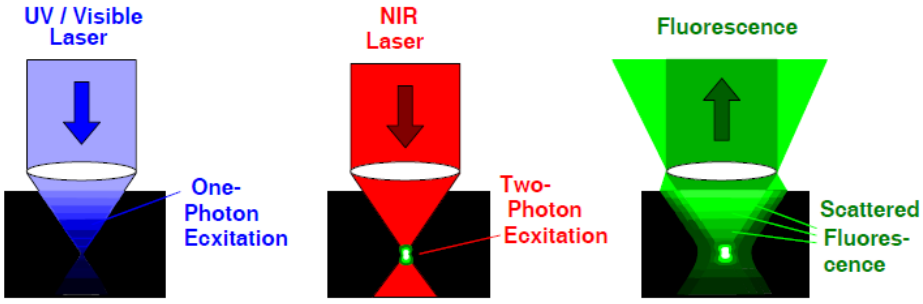

Figure 2.5: One-photon excitation - Excitation in the laser path, decreasing with increasing depth (left). Two-photon excitation - the NIR laser penetrates deeply into the sample, exciting fluorophores in the focal plane (middle). The fluorescence from a deep focus is scattered on the way out (right). Source: Becker & Hickl (2021).

Confocal descanned and non-descanned setup

In the late 1950s, Marvin Minsky developed confocal microscopy, aiming at imaging dense tissues by using two pinholes to restrict the excitation and detection beams and hence suppressing the detection of out-of-focus light (Naredi-Rainer, et al. 2017). When Minsky constructed the first confocal microscope, divergent UV lamps were used as the excitation source (Naredi-Rainer, et al. 2017). So, to create a point source in the focal plane from where the fluorescence signal would originate, a pinhole was inserted in the excitation path to select a collimated part of the light.

When using a laser instead to perform the excitation, the pinhole between the light source and the object is not needed, because the excitation beam is already collimated and able to create a point source in the focal plane, when overfilling the microscope objective back aperture. Many times, the laser beam diameter is smaller than the microscope objective back aperture size, and must therefore be expanded, to create a diffraction-limited spot in the focus called, i.e., a point source. (Naredi-Rainer, et al. 2017).

The confocal pinhole used in the detection path, between the object and the detector, defines or filters the light that reaches the detector (Naredi-Rainer, et al. 2017). The process of the fluorescence signal on the pinhole, and then collimating the signal again with a lens, is called spatial filtering (Kozma and Kelly, 1965). The pinhole is typically chosen to be equal in size to the first minimum of the Airy disk, which is the location with higher intensity, as illustrated by Figure 2.6.

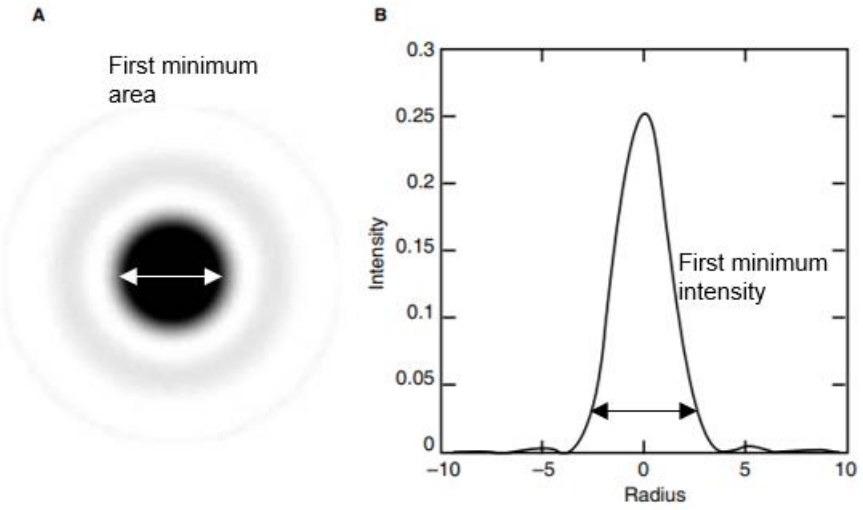

Figure 2.6: A – Airy disk. White arrows represent the first minimum of the Airy disk and B – Intensity of the Airy disk relative to the radius. The black arrows represent the intensity of the first minimum of the Airy disk. Source: Semwogerere and Weeks (2005).

Figure 2.7 shows the schematics of confocal microscopy, illustrating the function of the pinhole in confocal detection. Fluorescence photons produced in the selected z-plane can pass through the pinhole, and be detected, while photons from other z-planes are blocked by the pinhole. Widening the area of the pinhole increases the range of z-planes where photons can be detected. This process is called sample sectioning. Since only photons emitted from the focal plane are detected, this allows the observation of an optical slice of the object.

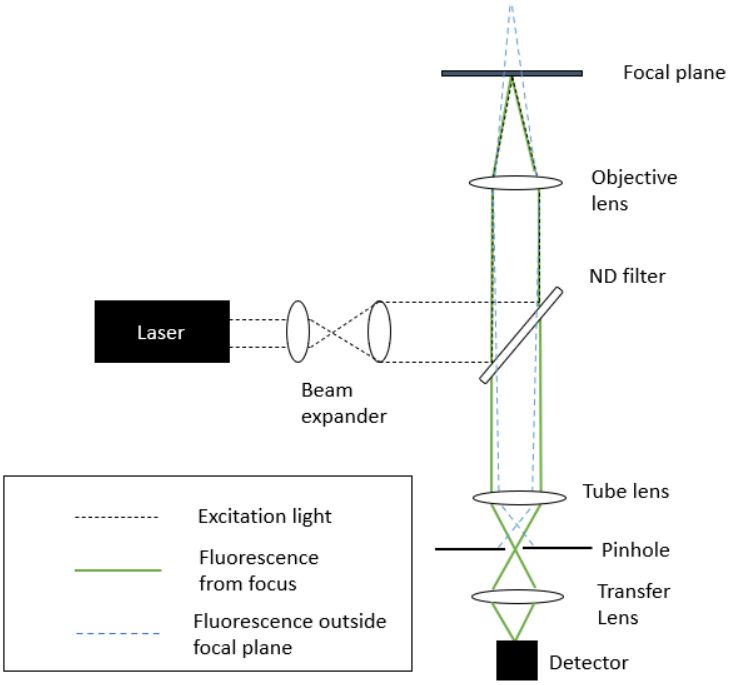

Figure 2.7: Schematic layout of confocal microscopy for one-photon excitation, illustrating the function of the pinhole in confocal detection. The excitation light is focused on an object. Fluorescence light from the focal plane passes through the pinhole and reaches the detector. Adapted by the author, based on Naredi-Rainer, et al (2017).

The presence of a pinhole in the detection path leads to an enhancement in image contrast, increasing spatial resolution. However, the signal throughput decreases due to the spatial filtering mechanism. To overcome the low signal intensity, light must be detected by a sensitive detector, with an internal gain mechanism by charge multiplication, such as Photomultiplier Tubes (PMT), Avalanche Photo Detectors (APD), or other detector types capable of detecting the arrival of single photons, converting the light signal into an electric one (Becker, 2021).

As described earlier there are two ways of moving the point source across the sample for characterization, either sample or beam scanning. Depending on the chosen scanning principle, the microscope layout will differ. The simplest way is sample scanning where the laser is stationary, and the collected fluorescence signal will therefore also not change the angle which it leaves the back aperture of the microscope objective. This means that the pinhole can be placed in a fixed position in what is called a non-descanned setup (Figure 2.7). This technique is a non-descanned one because it is not necessary to trace back the fluorescence light.

When the scan is done by beam scanning, i.e., by changing the laser light angle on the back aperture of the microscope objective, using for instance galvanometric mirrors, the non-descanned setup does not work. In this situation, the angle of the fluorescence light from the

focal plane changes continuously during the scan and is filtered out by the stationary pinhole, like out-of-focus fluorescence. Only sample positions very close to the centre position can pass through the pinhole and reach the detector, considerably limiting the scan area (Figure 2.8).

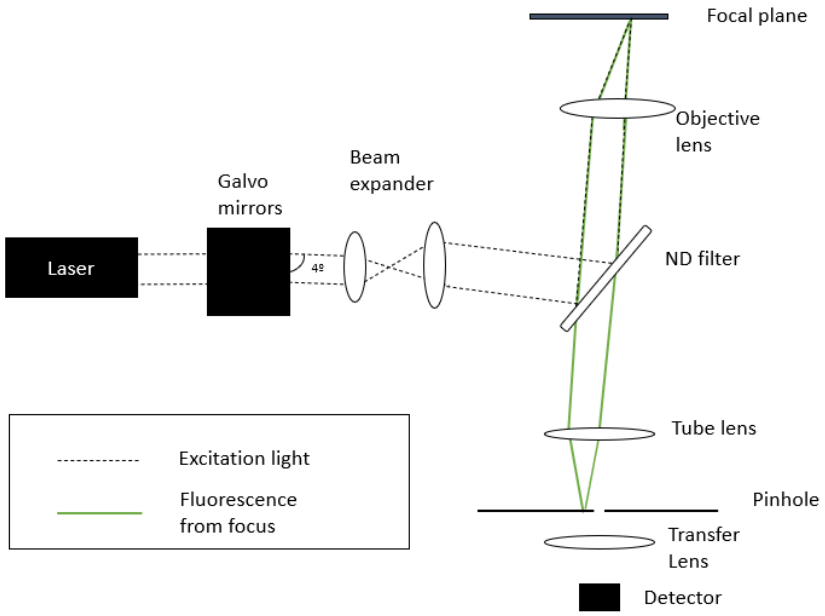

Figure 2.8: Non-descanned confocal setup for one-photon microscopy, by changing the laser angle to perform the imaging. Adapted by the author, based on Naredi-Rainer, et al (2017)

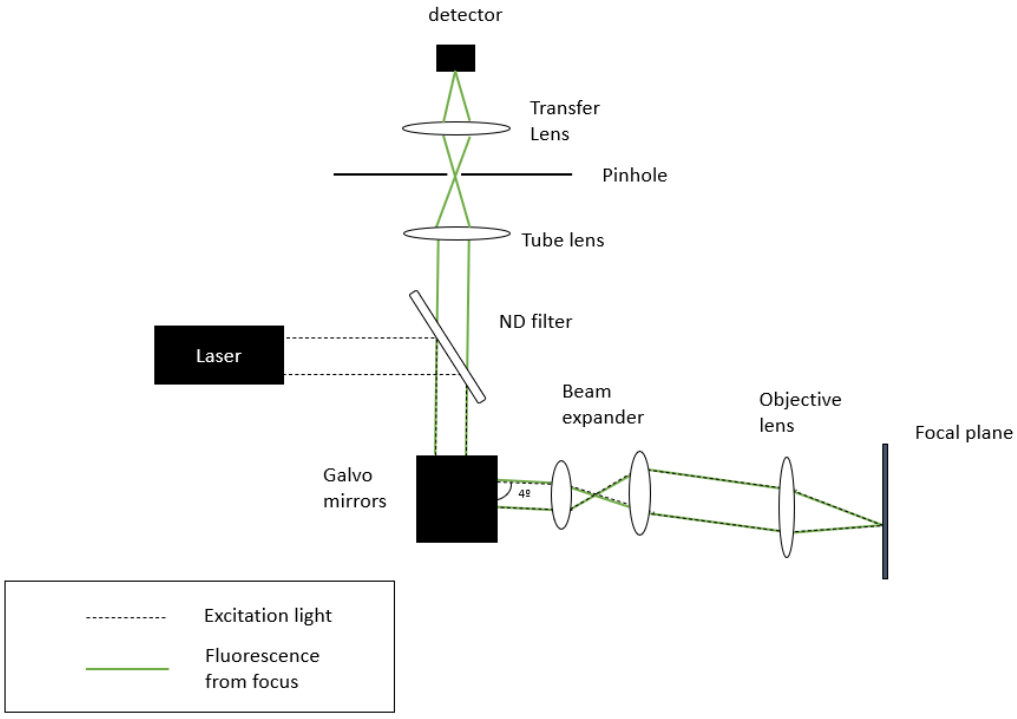

Figure 2.9: Descanned confocal setup for one-photon microscopy, changing the laser angle to perform the imaging. Adapted by the author, based on Naredi-Rainer, et al (2017).

A solution could be to move the pinhole in synchronization with the beam scanning, but this is unpractical. Another, and experimental simpler solution is passing the light back through the galvanometric mirrors, thereby always setting the fluorescence light angle to 90º, as seen in Figure 2.9. The in-focus fluorescence passes through a stationary pinhole and is detected. Since the fluorescence signal is passed through the galvanometric mirrors before detection, this method is called a descanned setup (Gibson et al., 2011).

Multiphoton detection setup

In a multiphoton microscope, the pinhole in front of the detector can be omitted since the fluorescence signal only stems from the focal volume, and no out-of-focus fluorescence needs to be filtered out. The omission of the pinhole allows for the detection of not only ballistic photons, i.e., photons that are on the optical axis, but also fluorescence photons that are scattered by the sample on their way back. Since the scattered photons enter the objective at a given angle, they also leave the back aperture of the objective at an angle and must be collected to reach the detector, thereby providing vital sample information. The angle from which scattered photons must be collected increases directly with the depth of the scan (see Figure 2.5).

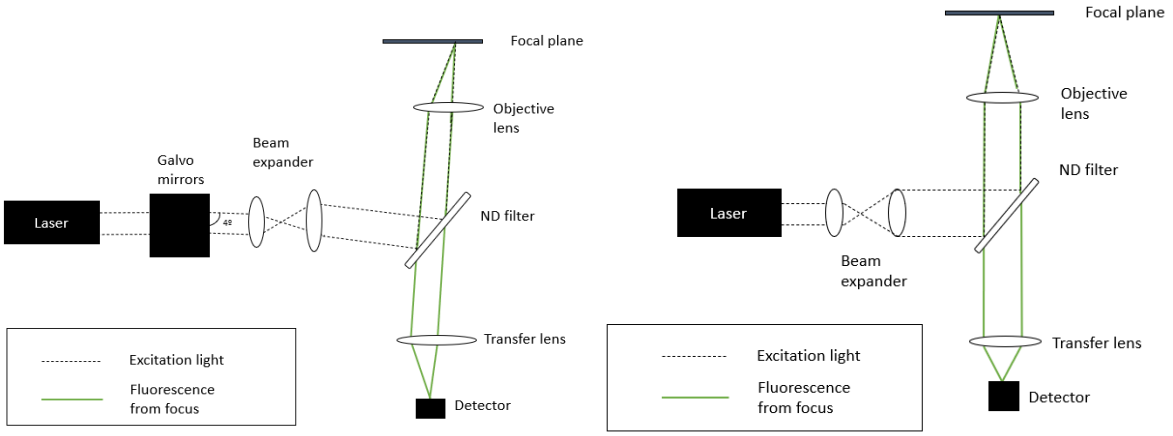

Figure 2.10: Non-confocal setup for multiphoton microscopy, for beam scanning (left) and sample scanning (right). Adapted by the author, based on Becker & Hickl (2021).

This problem is overcome by implementing a transfer lens, large enough to collect all angles of the fluorescent signal, projecting all photons to the detector, as seen in Figure 2.10. A transfer lens can be used for both beam and sample scanning. Since the angle of the fluorescence light does not impact the position of the focus on the detector, the detection efficiency for different fluorescence angles is the same. Because in multiphoton microscopy

there is no need to trace back the fluorescence light, it is always used on non-descanned detection setups (Figure 2.10).

2.1.5 Contrast

This subsection, heavily based on Cheng (2006) and Boyd (2003), gives a general idea of how contrast is developed in microscopy imaging.

All microscopy techniques require a contrast mechanism that allows the observer to see the shape of the structures of interest. In general, contrast is the difference of brightness and colour between the shapes of interest and the background. In fluorescence microscopy, changes of brightness, or colour might result from different fluorescence intensities, spectra, or fluorescence lifetimes.

In optical microscopy, the contrast provides information about the shape of the object and its parts and depends on physical, chemical, and biological phenomena. So, there must be an interaction between the light beam and the object to create image contrast. This interaction may include several mechanisms, such as linear and nonlinear absorption, single and multiphoton fluorescence, Raman emission, fluorescence spectral shift, fluorescence lifetime, refraction, reflection, phase shift, scattering, changes in polarization, and harmonic generation (Cheng, 2006).

When the beam of light has high intensity, nonlinear optical phenomena might occur, such as nonlinear absorption, multiphoton fluorescence, and harmonic generation (Boyd, 2003). These phenomena happen due to the modification of the optical properties of the object. Multiphoton fluorescence and harmonic generation are nonlinear phenomena since they occur when the response of the object depends in a nonlinear manner on the strength of the optical field (Boyd, 2003). For example, second-harmonic generation is originated from the atomic response that scales quadratically with the strength of the applied optical field (Boyd, 2003).

Instead of using the optical properties of the object to create contrast, fluorescent agents can be deployed in the object to create contrast by exploiting their properties when excited by a laser (Sevick-Muraca et al., 2002). Some of the fluorescence properties used to create contrast are intensity, fluorescence lifetime, fluorescence spectra, and relative concentrations of fluorophore fractions (Becker, 2021). The fluorescent agents recognize tissues or cellular structures through molecule-specific labelling. This results in very high contrast, sensitivity, specificity, and selectivity, allowing the observation in real-time of the structures inside a cell (Nienhaus et al., 2017).

2.1.6 Labelling/Probes

This subsection, heavily based on Nienhaus et al., (2017), deals with labelling/probing in fluorescence imaging.

Labelling in fluorescence microscopy can be understood through the railroad worm analogy. The railroad worm is an insect that emits two different light colours, yellow and red, and is the only insect that emits red light (Firefly Conservation & Research, 2022). Figure 2.11 exhibits the luminescence of the railroad worm.

Figure 2.11: Luminescence of the railroad worm. Source: Firefly Conservation & Research (2022).

Despite not observing some features of the railroad worm, such as shape, size, and colour, the observer knows it is a railroad worm. The observer reaches this conclusion because there is no known other insect that emits red light. Although presently there are almost one million known insects in the world, all of them are not visible given that they blend with the black background or have a different colour than red. This is the specificity of fluorescence labelling, which is crucial for fluorescence microscopy. A huge range of fluorescent molecules is available to be used as specific labels.

These fluorescent molecules are equipped with a functional group that recognizes tissues or cellular structures, allowing visualization with high contrast. Choosing the labelling might be difficult because fluorophores vary in physicochemical and optical properties. Table 2.1. lists some of these properties. Functional groups, such as organic dyes, fluorescent nanoparticles, and genetically encoded fluorescent proteins (FPs), are specific to a target,

having different performances according to the target properties, such as the target molecule, concentration, and location.

A suitable label must be photostable, excitable by the light source without the production of excessive fluorescence, and with high brightness to be easily detected by the instrumentation (Nienhaus et al., 2017).

Table 2.1: Key properties of fluorophores

Properties	**Definition/Impact**
Optical properties	
Excitation spectrum	Wavelength dependence of the ability of light to induce fluorescence emission of a chromophore
Extinction coefficient ε_λ $(M^{-1}\,cm^{-1})$	Quantity characterizing a fluorophore's ability to absorb photons at a particular wavelength λ
Emission spectrum	Wavelength dependence of the emitted fluorescence
Stokes shift	Wavelength shift between excitation and emission bands
Fluorescence quantum yield (QY)	Ratio between the number of photons emitted and the number of photons absorbed
Molecular brightness	Product $QY \times \varepsilon$ quantifies the relative rate of photon emission from different fluorophores under identical excitation conditions
Photostability	Resistance to photobleaching, quantified by the quantum yield of photobleaching, i.e., the ratio of the number of photobleached molecules to the total number of photons absorbed in a sample during the same time interval
Physicochemical properties	
Size	Most often, the smaller the better
Material	May affect chemical stability, toxicity, and options to functionalize the probe
Solubility	Most biological applications require aqueous solvents
Cytotoxicity	Effect on sample viability
Phototoxicity	Effect on sample viability
Conjugation chemistry	Determines labelling target
Localization of target	Affects the choice of conjunction
Metabolism of target	Determines temporal stability of labelled construct in live cells

Notes: Different properties of fluorophores must be considered when deciding which fluorophore should be used in labelling**.** Source: Nienhaus et al., (2017).

2.2 Fluorescence Lifetime Imaging Microscopy (FLIM)

Fluorescence Lifetime Imaging (FLIM) microscopy consists of measuring the fluorescence lifetime of the molecules found in every pixel of an image. The lifetime can be measured through frequency-domain metrics, based on phase shift and demodulation of the resulting fluorescence signal after the excitation pulse; or through time-domain measurements, obtained using the principle of Time-Correlated Single Photon Counting (TCSPC), which is

mainly applied in laser scanning microscopes (Becker, 2021). Figure 2.12 shows how TCSPC records the distribution of the photons through time after the excitation pulse.

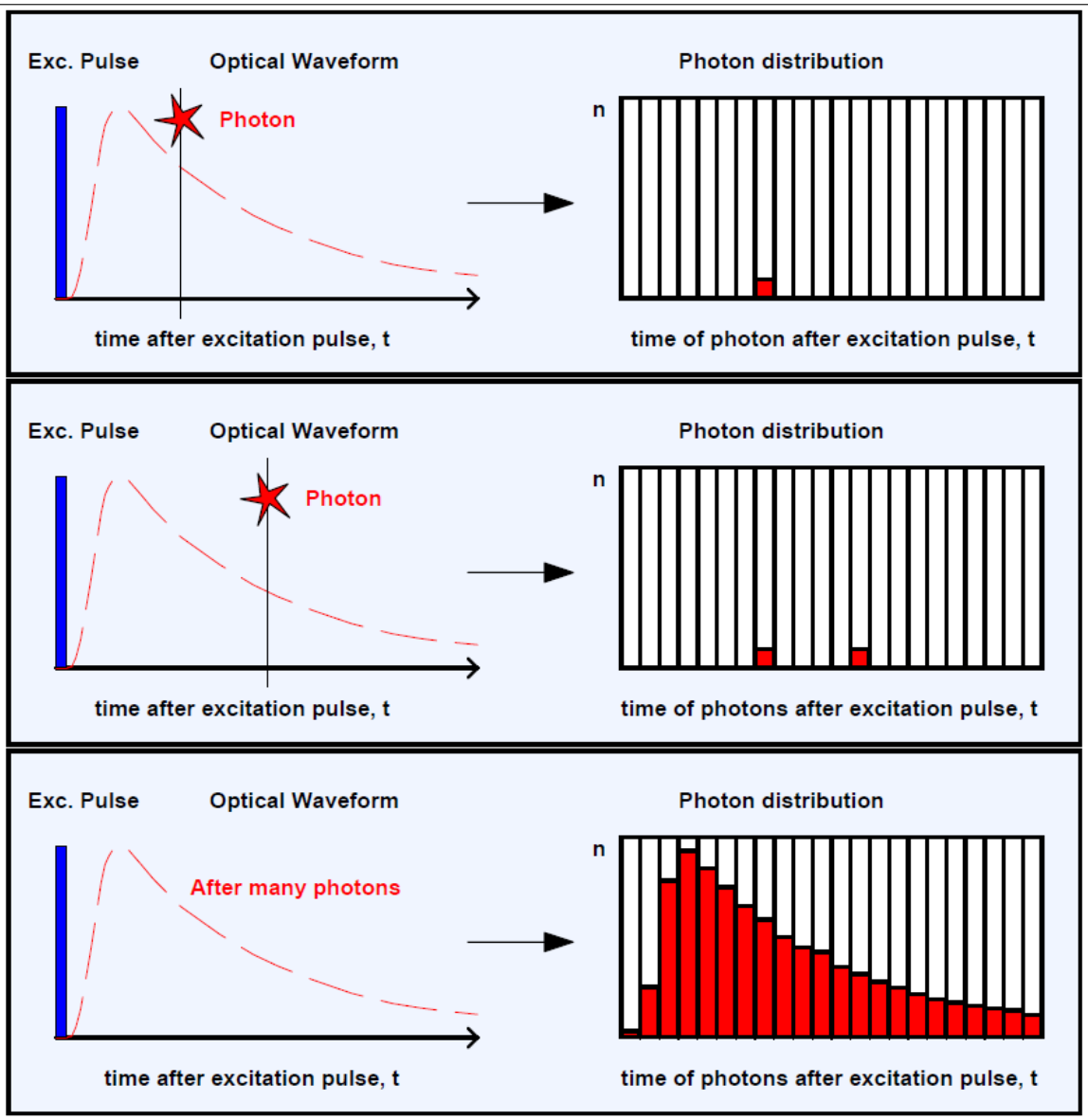

Figure 2.12: Principle of TCSPC. The recording process builds up the distribution of the photons over time after the excitation pulse. Source: Becker (2021).

TCSPC is the most accurate imaging method to determine lifetime information. It relies on a focused beam with a high repetition rate pulsed laser that scans the object (Becker, 2021). Depending on the type of laser, the sample fluorescence is excited by one-photon or multiphoton through a confocal setup, non-descanned detection, or a combination of both.

When an individual fluorescence photon is identified by the detector, the corresponding arrival time is measured using a fast electronic circuitry. This technique commonly uses detectors able to detect single photons, such as photomultiplier tubes, hybrid detectors, and

avalanche photodiodes (Becker, 2021). After the detection of many photons, the distribution of photons through time is built, resulting in a waveform that is related to the decay profile of the detected fluorescence. The fluorescence lifetimes are extracted from that waveform through a fitting algorithm. The fitting algorithm used in FLIM can process multiexponential decays, making it possible to distinguish different fluorophores by extracting all lifetime components with their respective weight.

In terms of hardware, the TCSPC instrument is synchronized with the laser pulse rate and the photon detector. This allows acquiring the photon arrival time associated with the pulse that excited it (see Figure 2.12).

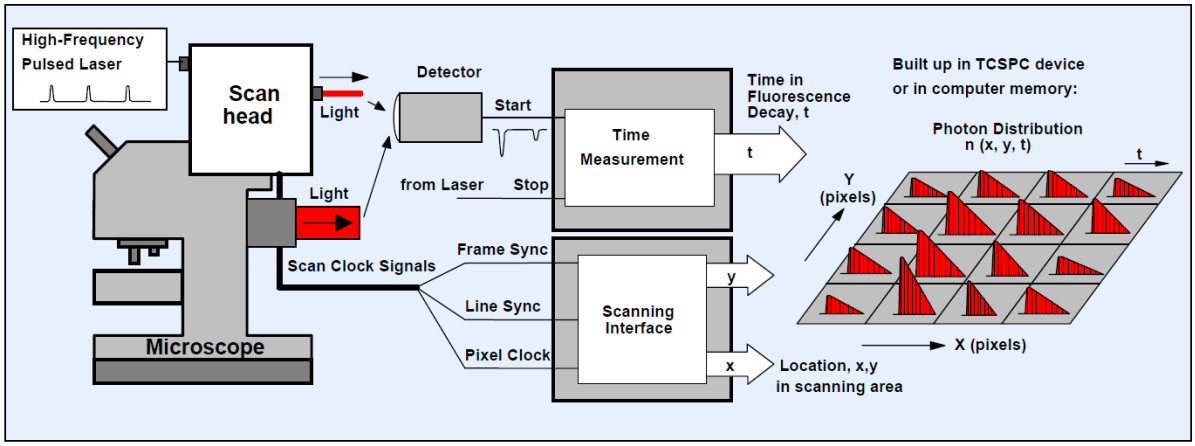

Figure 2.13: Multidimensional TCSPC architecture for FLIM. Source: Becker (2021).

After having the lifetime for each pixel, a colour can be associate with each lifetime range, resulting in a coloured image (Figure 2.13). This type of imaging has speed limitations since for each pixel it is necessary to obtain fluorescence information for a specific amount of time.

From the TCSPC information, it is possible to obtain the intensity for each pixel, which corresponds to the total photon counts present. The intensity can also be used to create contrast by associating each pixel to a colour range between the lowest and the highest intensity pixels.

The detectors have an instrument response function (IRF). When passing a temporal input through an electronic system, the IRF is the temporal response of the system to a Dirac delta function input signal (Görlitz, 2018). A detector can record the fluorescence decay by producing a time-dependent output signal resulting from the convolution between the IRF and the actual fluorescence signal.

2.3 Femtosecond Lasers for Two-Photon Microscopy

A so-called standard laser source for two-photon microscopy is a Ti:Sa laser, which in our case provides pulses of 70 fs when transformed limited, has a relatively narrowband spectral wavelength band and can be tuned between 730 and 1000 nm (Maibohm et al., 2019). In comparison, the ultra-broadband spectrum of the 7 fs Ti:Sa laser source used in this work spans the same spectral range without any spectral tuning needed. Furthermore, the wide spectrum of the 7 fs laser have enough intensity across the full spectrum for efficient excitation, and the spectrum profile can be optimized to match the absorption of various fluorophores(Maibohm et al., 2019).

Figure 2.14 shows the intensity as a function of the wavelength for different spectra profiles. The spectral profile of the 7 fs laser matches the absorption of various fluorophores in a multicolour labelled sample. Besides the possibility to excite fluorophores in the visible spectra range, the 7 fs has less scattering and absorption relative to the 70 fs laser, reaching 50% deeper inside the tissue (Maibohm et al., 2019).

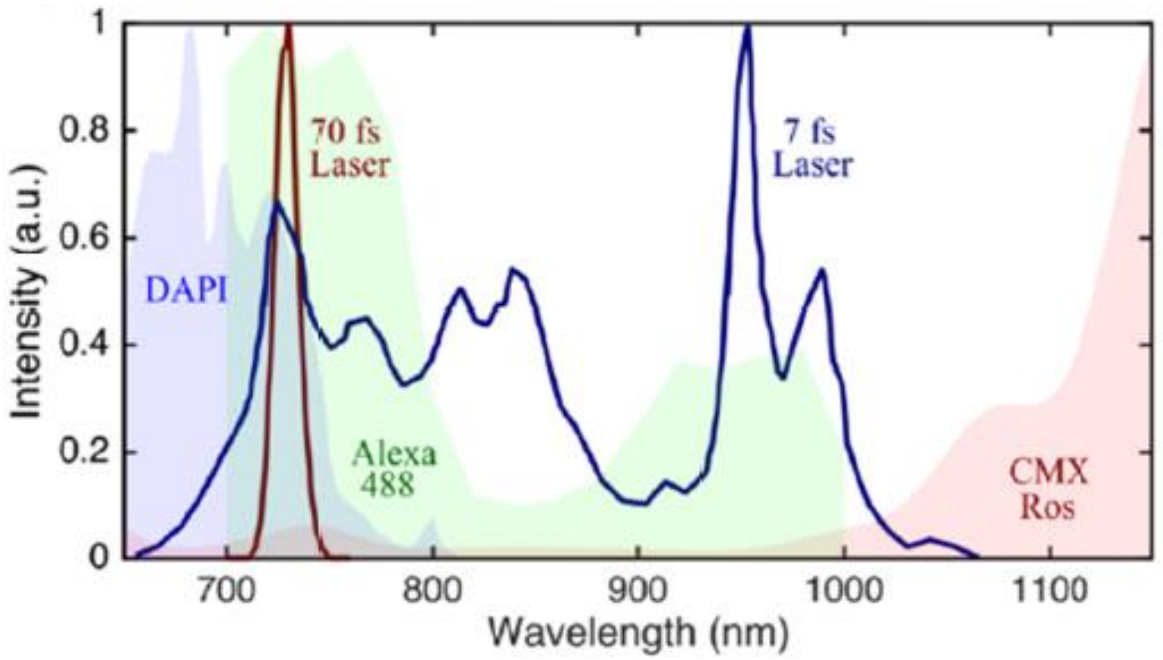

Figure 2.14: Intensity as a function of the wavelength. Laser sources of 7 fs and 70 fs, with the excitation range for different fluorophores (Maibohm et al., 2019).

2.4 Second harmonic generation

Besides two-photon fluorescence, other nonlinear signals may occur, known as secondary, or even tertiary, harmonic generation (Cheng, 2006). In a Second Harmonic Generation (SHG), there is the destruction of two photons of frequency ω, with the simultaneous creation of one-photon of frequency 2ω. Since the destruction of two photons needs to be almost simultaneous, this only occurs in the focal plane of the incident light, where there is very high intensity. SHG is sensitive to the orientation between the polarization of the incident light and the symmetry condition of the object, providing information about the crystal orientation, regularity, and molecular structure. In a third harmonic generation, three photons of frequency ω are destroyed to create one-photon of frequency 3ω, requiring more energy than the SHG. Secondary and tertiary harmonic generations are diagrammatically represented in Figure 2.15.

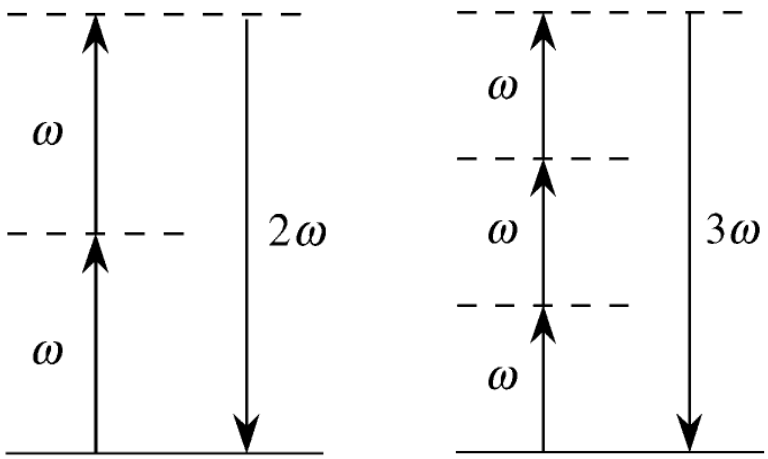

Figure 2.15: Energy diagram of secondary (left side) and tertiary (right side) harmonic generations. Source: Boyd (2003).

2.5 Characterization of Broadband Few-Cycle Laser Pulses

There is a strong demand for high signal gain in ultrashort pulses, which are achievable by performing pulse optimization. External pulse compression schemes can be employed to provide the highest signal possible in few-cycle regimes. To optimize the signal, it is crucial to characterise the pulse and verify if the laser keeps its properties after amplification. There are

several techniques for pulse characterization, based on phase reconstruction or differential chronocyclic tomography (Li et al., 2007; Walmsley and Dorrer, 2009).

Miranda et al. (2012) developed a technique, called dispersion scan (d-scan), which characterizes ultrashort pulses while compressing them. This technique uses chirped mirrors and glass wedges. The chirped mirrors introduce fixed quantities of negative dispersion and are used together with a pair of glass wedges, which adds positive dispersion, and the total amount of dispersion can be fine-tuned by moving one glass wedge on a translation stage. This technique is illustrated in Figure 2.16.

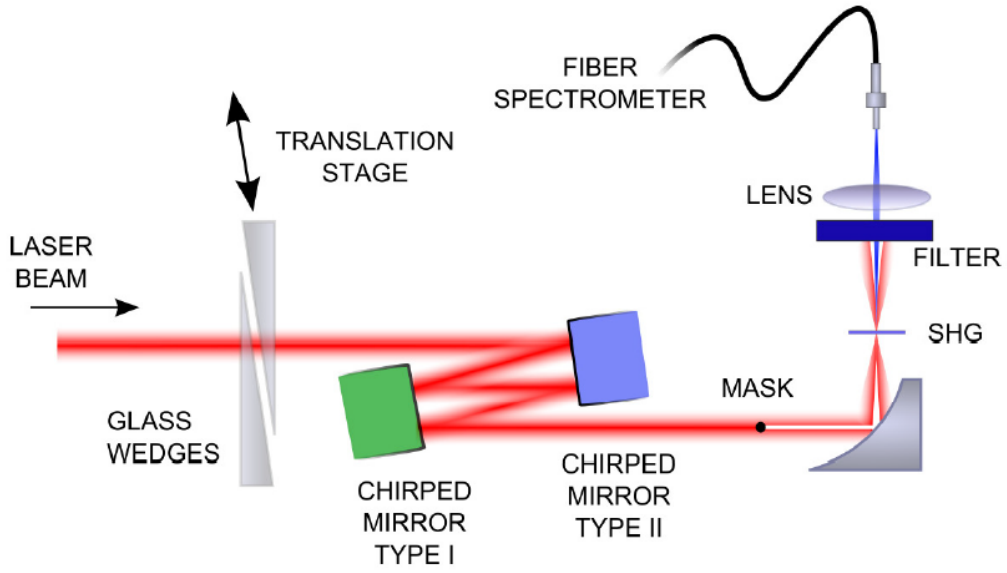

Figure 2.16: Setup required for a d-scan. Source: Miranda et al. (2012).

The initial pulse from a laser source is perfect and has zero dispersion. If an objective has a positive dispersion of 4000 fs^2, then there is need to add negative dispersion to reach the perfected pulse (no dispersion), which is achievable by adding the negatively chirped mirrors. Each pass or reflection on the chirped mirror adds minus 1300 fs^2 of dispersion, hence more chirped mirrors are needed for the total to be negative. By introducing 42 reflections on the chirped mirrors, a negative dispersion of -5200 fs^2 is reached, rendering a setup with a total negative dispersion -1200 fs^2. The glass wedges provide 1200 fs^2 of dispersion to reach the perfect pulse.

The optimal dispersion of the glass wedges is computed by measuring the fundamental spectrum and the SHG in a crystal around a large range of dispersions, which considers different amounts of insertion of the glass wedges, together with a numerical iterative algorithm (Miranda et al., 2012).

The d-scan technique has multiple advantages, such as its simplicity, sensitivity, relaxed bandwidth requirements, ability to measure the relative phase between well-separated frequency components, and ability to find the laser focus (Miranda et al., 2012).

Before doing the d-scan, the focus of the laser, which corresponds to the point where there are more SHG traces, needs to be found. The focus of the laser is measured by a spectrometer, and only then the software finds the best glass wedge insertion. Figure 2.17 exhibits the result of the d-scan from an experiment conducted in September 2021, showing that the optimal glass insertion is around 31 mm.

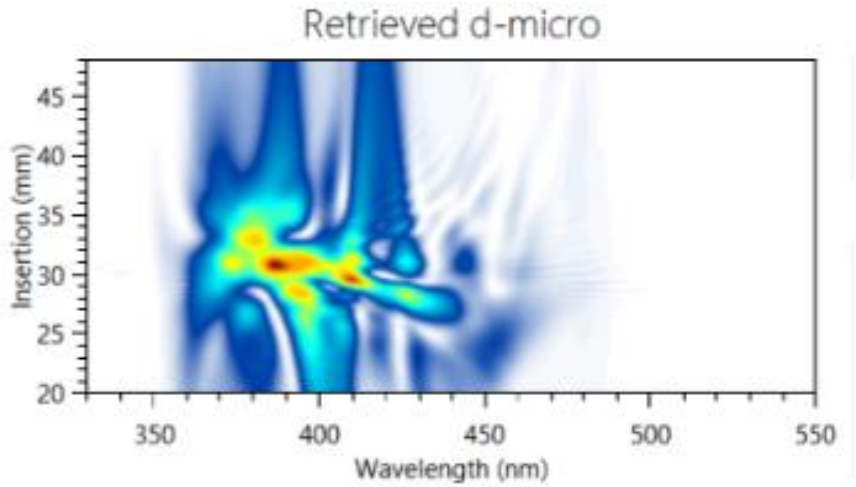

Figure 2.17: Result of d-scan from an experiment conducted in September 2021 at INL, using a 7 fs Ti: Sapphire laser. The d-scan shows that the optimal glass insertion is around 31 mm.

2.6 Importance of Fluorescence Multiphoton Microscopy in Metabolic Imaging

Many optical approaches have successfully monitored metabolic imaging, such as Positron Emission Tomography (PET) and Magnetic Resonance Spectroscopy (MRS) (Momcilovic & Shackelford, 2018). The main advantage of optical microscopy for monitoring metabolism is that it increases spatial resolution while enabling the visualization of metabolism in a single cell (Liu et al., 2018).

Mitochondria is the powerhouse of any cell, having a major role in cell energy metabolism and therefore influencing biological functions, such as fatty acid oxidation, Reactive Oxygen Species (ROS), and apoptosis. Therefore, mitochondria can be used as a prime target for cancer therapy (Faria et al., 2019). Fluorescence of NADH and NADPH has

been employed to monitor metabolism in cancer cells. NADH and NADPH have an emission spectrum indistinguishable in aerobic and anaerobic conditions of the metabolism but show different fluorescence lifetimes when free or attached to a protein (Faria et al., 2019). Since the NADH and NADPH have distinct lifetimes, it can be possible to monitor the cells metabolism label-free. This is possible by FLIM using two-photon microscopy, which is capable to do deep lifetime imaging with high resolution (Bower, 2019).

3 Methods and Hardware

This chapter contextualizes my main assignment during the thesis' work in terms of methods and hardware. The assignment consisted of software development capable of device control and data analysis, which was crucial for the proper working of the microscope. Section 3.1 presents the software requirements, and development approach and testing. Section 3.2 presents the microscope hardware setup, with a focus on its controllable components.

3.1 Software requirements, development approach and testing

Before initiating the software development, it was important to systematize the desirable requirements of the microscope system. These requirements were discussed in several meetings with the other elements of the ExtreMed team. Table 3.1 presents the software requirements.

Table 3.1: Software requirements

Qualitative attributes	System requirements
Performance	Perform without delays.
Usability	Usable through a friendly easy-to-use graphical interface.
Compatibility	Compatible with different operating systems
Flexibility	Easy implementation and change of features in an open-source coding language.
Multitask	Able to control different components (have a library for coding) and perform data analysis.
Data output	Returns comprehensive data, following the principles of accessibility, readability, and flexibility.
Data re-usability	Data should be recordable, uploadable and workable in the software or external ones.
Data analysis	Able to perform analyses in real-time.

The software was developed with Python 3.8.5. This is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is characterized by its accessibility (it is a free open-source platform), completeness (it has vast library sources), and portability (it is internally consistent with codes written in other programming languages while being able to be used in different operating systems). The user interface was developed in a Qt Framework port for Python, known as PyQt 5.

Before starting software development, all libraries of the controllable components were studied. These components are the shutter, galvanometric mirrors, x-y microscanner, z-piezo, and TCSPC electronics. The functions libraries of each device, written in C^{++}, were converted into Python through a Python wrapper code, which strongly relies on the python library *ctypes,* allowing to read C^{++} computing code. After the conversion, the Python functions of the components, based on the initial libraries, were developed for each device. Most notably, the software was developed in a twofold manner, aiming at the simplification of the control features, and allowing an easier implementation on the user interface.

After implementing the device control subsystem, scanning patterns using the galvanometric mirrors, x-y microscanner, and z-piezo were developed to obtain data using the TCSPC electronics. Additionally, mechanisms to save all relevant data in text files were created. The text files are automatically saved with a name based on key information to be easily findable.

After the creation of device control and data recording subsystems, the data analysis subsystem was implemented. Data analysis codes were directly connected to the recording subsystem, allowing real-time analysis. An option to load recorded data was also created, allowing the user to immediately upload a valid file on which he may perform other analyses.

Mechanisms for the FLIM, intensity imaging, and pixel histograms were implemented. These functions provide information from each pixel and are easily altered and upgraded. The imaging was done using the *matplotlib* library, while the fitting algorithm was done using the *SciPy.* The software benefits from the *NumPy* library, simplifying the software development.

The software development was simultaneously accompanied by testing procedures. Several samples with known structures were scanned, which allowed feedback on the device control, data recording, and data analysis functions. To aid the user when controlling the different devices, an auxiliary function was constructed to read the physical positioning of the z-piezo, galvanometric mirrors and x-y microscanner.

The hardware components that constitute the SyncRGB microscope prototype to be built in the context of the ExtreMed project and for which control software was developed in the framework of this thesis, are described below.

3.2 Microscope Hardware Setup

The ExtreMed demonstrator is a custom-built microscope platform, where each part has been selected and mounted together to achieve the goal of performing SyncRGB-FLIM measurements with both beam and stage scanning.

Figure 3.1 shows a schematic overview of the first version of the beam and stage scanning SyncRGB-FLIM microscopy, and how they are connected. The microscope layout is separated into numbered subsections to ease the description.

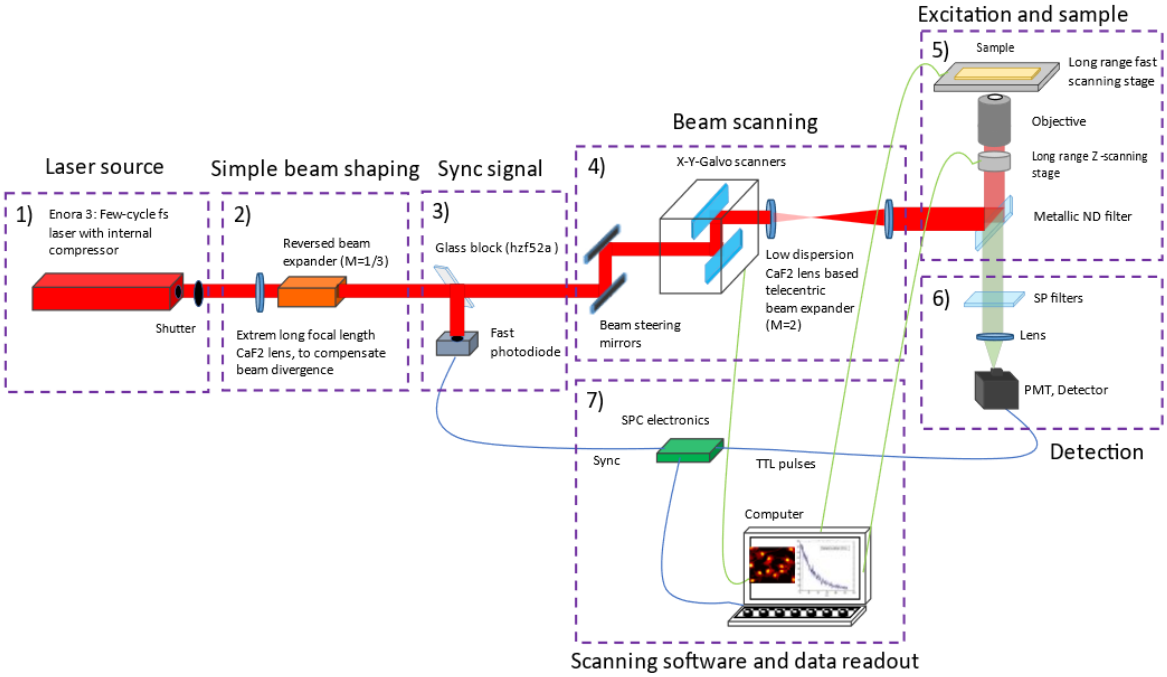

Figure 3.1: Schematic layout of SyncRGB-FLIM microscope.

1) The excitation source for the SyncRGB-FLIM microscope consists of a passive mode-locked Ti:Sa laser providing ultra-broadband few-cycle excitation (Enora 3) developed by Sphere Ultrafast Photonics. The laser can deliver sub 10 fs pulses with an output spectrum covering the wavelength range from 650 to 1050 nm (see Figure 3.2), and an average power between 50 and 90 mW at a repetition rate of 80 MHz. The laser oscillator output has a pre-compression step involving a transmission through a set of glass wedges, where one glass wedge is mounted on a linear translation stage so that the amount of glass inserted into the beam path, and thereby the amount of positive second-order dispersion, can be controlled. The number of reflections from negatively chirped mirrors introduces a fixed amount of negative dispersion to the laser pulse. In this case, 42 reflections introduce a negative second-order dispersion of 5200 fs^2.

Alongside the d-scan hardware, which comprises a specifically designed measurement head that fits the microscope stage, this configuration allows pulse characterization to be executed at the focus of the microscope objective. The software to analyse the d-scan traces was also provided by Sphere Ultrafast Photonics.

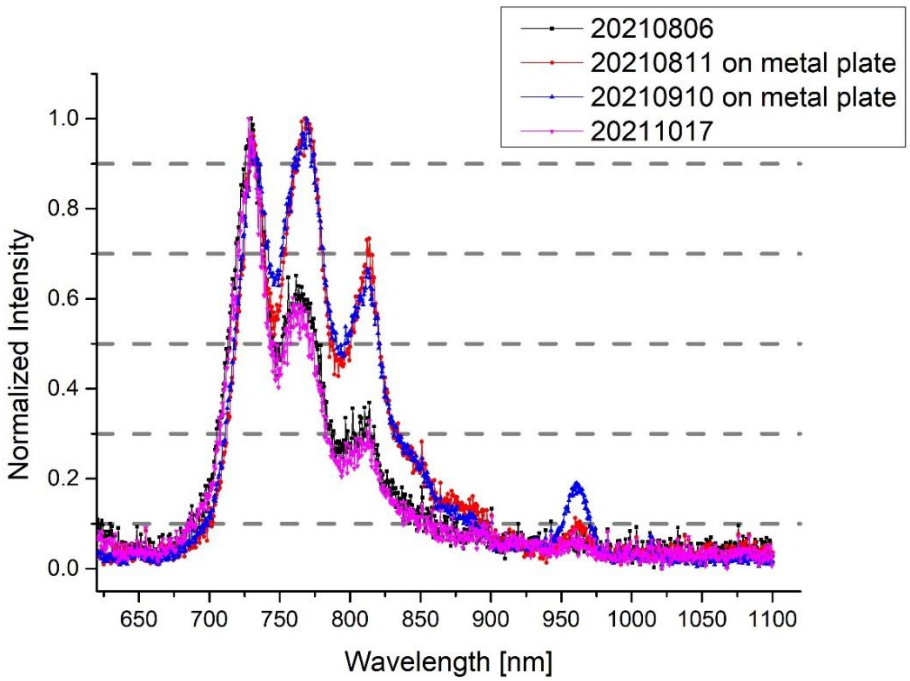

Figure 3.2: Enora 3 laser spectrum from different dates, with and without metal plates.

Firstly, the laser is tuned while observing the spectrum and the output power. After setting the laser properly, the d-scan is performed, for characterizing the pulse. A measurement head is placed on the focal plane of the microscope, aiming at measuring the spectrum and the SHG. Although the software provides diverse types of information, the focus should be on the pulse width, relative peak power (i.e., the amount of power from the laser that is used to perform the fluorescence excitation), and optimal insertion (i.e., the amount of glass that is used to provide dispersion). The d-scan needs to be performed in the focus of the laser, which corresponds to the spot where the measurement head provides more SHG signals.

In October 2021 a d-scan was performed, which produced the information presented in Figure 3.3 along with the following information:

- Pulse width: 9.44 ft
- Relative peak power: 69.7%
- Optimal insertion: 31.02 mm

The pulse width was close to the benchmark of 7 femtoseconds.

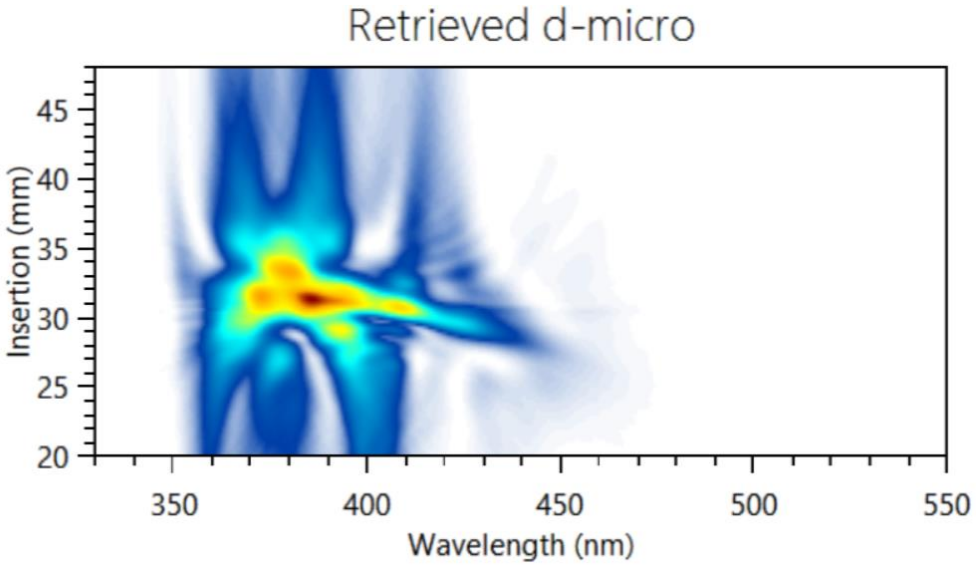

Figure 3.3: Example of d-scan of the laser in October 2021, showing that the optimal glass insertion is around 31 mm, obtained using the software provided by Sphere Ultrafast Photonics.

2) The reverse beam expander, together with an extremely long focal length lens, is used to reduce the diameter and divergence of the laser beam. In this case, the beam expander reduces de beam to 1/3 of the initial beam diameter. To limit the introduction of dispersion from the different elements the inverse beam expander is reflective using parabolic silver mirrors, which does not add any dispersion to the laser pulse. The extremely long focal length lens is made from CaF_2, which is a low dispersion type of glass.

3) The glass block of the high dispersion glass hzf52a splits the laser beam in two after the reverse beam expander, by reflecting a few % of the laser to the fast photodiode. The fast photodiode is connected to the TCSPC electronics, which is used to provide the time of when the laser pulse is sent.

4) Guiding mirrors are used to change the direction of the laser beam, directing the laser to the galvanometric mirrors. The galvanometric mirrors can either be parked in the zero or centre position when stage scanning is performed and will work as normal mirrors or move in beam scanning. After the galvanometric mirrors, there is a beam expander, consisting of two positive CaF_2 lenses that are used to expand the diameter of the laser beam 2.5 times. This allows the laser beam to overfill in the objective lens, which is needed to create a diffraction-limited spot at the focus. The second function of the beam expander, or telescope, is to work as a telecentric system when performing beam scanning. Here the deflection angle of the galvanometric mirror is translated into an input angle at the back aperture of the microscope objective, thereby moving the focus spot in the focal plane.

5) After the beam expander, the laser beam is partly reflected by a metallic ND or neutral density filter towards the microscope objective and the sample for fluorescence signal generation. The fluorescence signal is collected by the same objective and sent through the same ND filter to the detection path. The amount of light from the laser that is reflected in the sample and light from the fluorescence of the sample that reaches the detector is dependent on the optical density of the ND filter used. Highly effective dielectric dichroic mirrors can´t be used with ultra-short laser pulses since they introduce too much phase distortion to be able to maintain an ultra-short pulse.

6) After the ND filter in the detection path, a short pass (SP) filter is used to block any light above 680 nm that reach the detector. Since the ultra-broadband laser spectra have a tail below 680 nm, additional 633 nm short pass filters are used to remove the last laser light from the fluorescence signal. The light that goes through the SP filters is then focused on the detector by a lens. The detector used is a photomultiplier tube, connected to the SPC electronics, which provides a signal when it detects a photon.

7) The signal from the PMT detector is sent to the TCSPC electronic, together with the synchronization from the fast photodiode for recording TCSPC curves. A computer is used to control all the controllable parts, along with the obtained data from the TCSPC electronics, to perform data analysis.

Figure 3.4 and Figure 3.5 show the microscope components placed in the lab at INL, numbered to correspond to Figure 3.1.

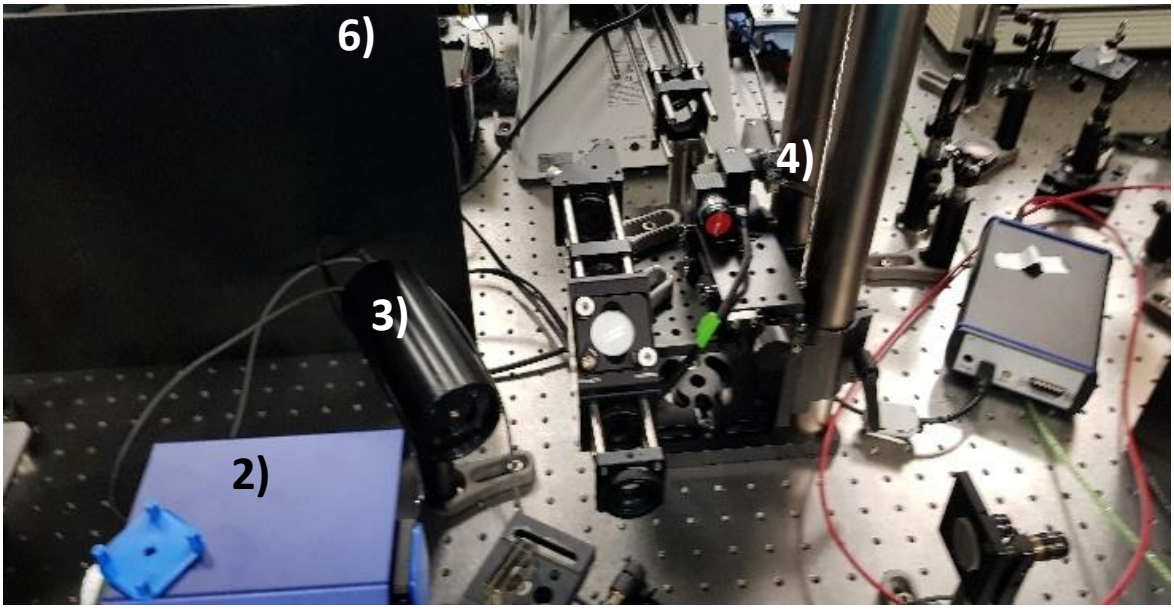

Figure 3.4: View of 2) beam expander, 3) photodiode, 4) beam scanning with the galvanometric mirrors, and 6) PMT detector.

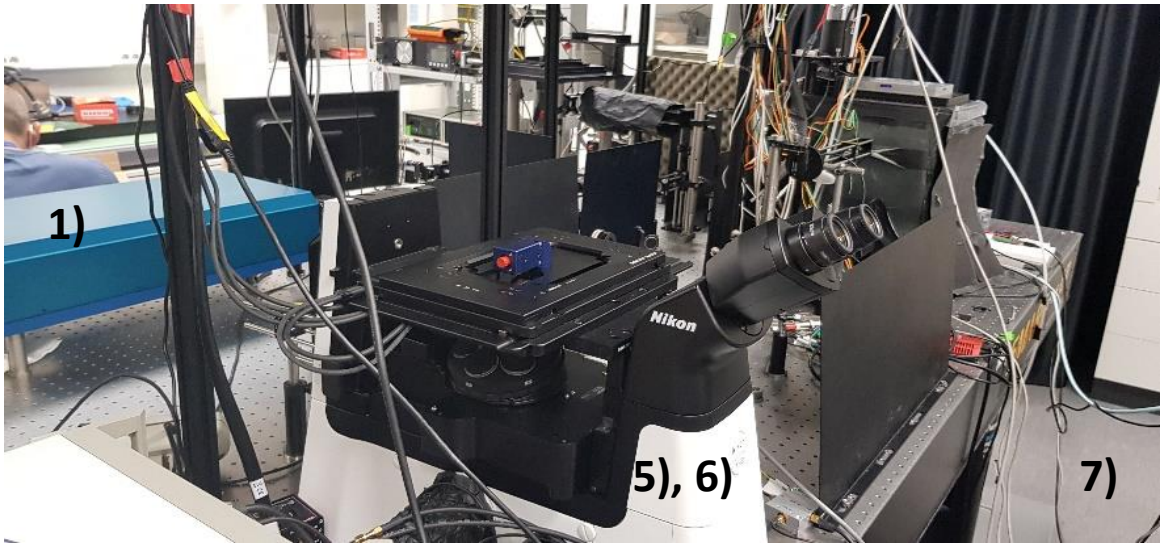

Figure 3.5: View of 1) laser source, 5), 6) excitation and sample, and 7) software controller.

3.2.1 Computer and software controllable components

Shutter

The shutter used is the SHB025T - Ø1/4'' Low-Reflectance Diaphragm Optical Beam Shutter, produced by Thorlabs, GmbH (Newton, New Jersey, USA). Figure 3.6 shows the shutter in the closed and open positions. The shutter is controlled by voltage. With 5V, the shutter is fully open and with no voltage the shutter is fully closed. The shutter takes 3ms from being in one stage to the other (Thorlabs, 2022a). The shutter has a jitter of 1.8ms, i.e., it takes 1.8ms to start rising or closing after the signal is provided. The shutter is connected to a National Instruments (Austin, Texas, USA) NI-DAQmx card to provide the control voltage. The shutter is used to block the laser beam from reaching the sample to limit photobleaching, when not scanning.

Figure 3.6: SHB025T - Ø1/4" Low-Reflectance Diaphragm Optical Beam Shutter. Source Thorlabs (2022a).

x-y galvanometric scanners

The galvanometric mirrors, displayed in Figure 3.7, are known as “GVSM002-EC/M” and were also obtained from Thorlabs, GmbH. These scanners are used for beam scanning where both accuracy and speed, controlled by voltage, are of utmost importance.

Figure 3.7: Galvanometric mirrors, model “GVSM002-EC/M”. Source: Thorlabs (2022b).

The galvanometric mirrors are also controlled by voltage, provided by a National Instruments' current-generation data acquisition driver card (NI-DAQmx). This galvanometric mirror model can use three different types of voltage/degree conversion, namely 1V/º, 0.5V/º, and 0.8V/º (Thorlabs, 2022b). In our setup, the standard configuration of 0.8V/˚ is used.

x-y microscanner

The x-y microscanner is a High-Speed Motorized XY Scanning Stages, model MLS203. Its picture is shown in Figure 3.8.

Figure 3.8: High-Speed Motorized XY Scanning Stage. Source: Thorlabs (2022c).

Table 3.2 lists the MLS203-1 stage specifications.

Table 3.2: Specifications of the MLS203-1 stage.

Travel Range	110 mm x 75 mm (4.3" x 2.95")
Velocity (Max)	250 mm/s
Acceleration (Max)	2000 mm/s2
Horizontal Load Capacity (Max)	1.0 kg (2.2 lb)
Min. Achievable Incremental Movement	0.1 µm
Absolute On-Axis Accuracy	<3 µm

The x-y microscanner is connected to a “BBD202” controller, illustrated in Figure 3.9. Both the stage and the controller were also manufactured by Thorlabs, GmbH. The fast-scanning stage is used for both sample positioning before scanning and stage scanning.

Figure 3.9: “BBD202” controller. Source: Thorlabs (2022d).

Z-scanning stage

The z-scanning stage model is ND72Z2LAQ PIFOC Objective Scanning System 2000 µm, manufactured by the Physik Instrumente Group, which is also considered a z-piezo. This z-scanning stage has a resolution of 5 nm, and a settling time around 20ms (Physik Instrumente Group, 2022). The device's picture is shown in Figure 3.10. The z-stage is needed for creating 3D images and this model was chosen since it has a very long traveling range of 2000 µm, needed for deep tissue imaging.

Figure 3.10: ND72Z2LAQ - z-scanning stage. Source: Physik Instrumente Group (2022)

TCSPC electronics

For the Time-Correlated Single Photon Counting (TCSPC), the device used was the "MultiHarp 150N" from PicoQuant (Berlin, Germany) (Figure 3.11). The MultiHarp is connected to a photomultiplier tube, that provides the signals on when a photon is detected, and to the fast photodiode for synchronization. The minimum time bin reachable for the MultiHarp is 80 ps which is the time bin size used in all scans presented in this thesis.

Figure 3.11: MultiHarp 150N - Time-Correlated Single Photon Counting (TCSPC) electronics. Source: PicoQuant (2021).

Table 3.3 lists the MultiHarp 150N specifications.

Table 3.3: Specifications of MultiHarp 150N.

Detection channels	4
Temporal resolution	80ps
Dead time	<0.65ns
Acquisition time	1 ms to 100 hours
Maximum number of time bins	65536
Interface	USB 3.0
Outstanding features	Put together high throughput with short dead time

4 Software for Device Control and Data Analysis

This chapter will descript in detail the individual functions of the custom developed software for the ExtreMed microscope demonstrator. Then will describe the interface and functionality of the software, as well as the general thoughts about the implementation of the scan functionalities.

The developed custom software allows to control a x-y microscanner or galvanometric scanning to obtain TCSPC data for each pixel. An automated protocol stores the relevant measurement settings in a text file, together with the recorded data. The device control software that interconnects the 4 hardware elements (shutter, x-y microscanner, galvanometric mirrors, and TCSPC) was fully developed during the thesis' work. Figure 4.1 shows the main interface window that connects the hardware elements and performs the data analysis.

The device control, data analysis, and graphical user interface were implemented in the same environment, allowing real-time interactions. To prevent the user interface to freeze during the scanning process, all functions have a subfunction "app.processEvents()", which executes pending actions, so that any real-time interaction is not queued in the software sequence, but is instantly executed instead.

The microscope is initiated, and all the controllable components are activated, when pushing the button "Connect microscope" (Figure 4.1a). In case of the user decides to activate the z-piezo, the button "Activate z-stage" (Figure 4.1b) must be checked before pressing the button that connects the devices. The z-piezo provides some features that are not crucial for running the basic XY-scanning program, being the reason why there is this extra optional to connect in the software.

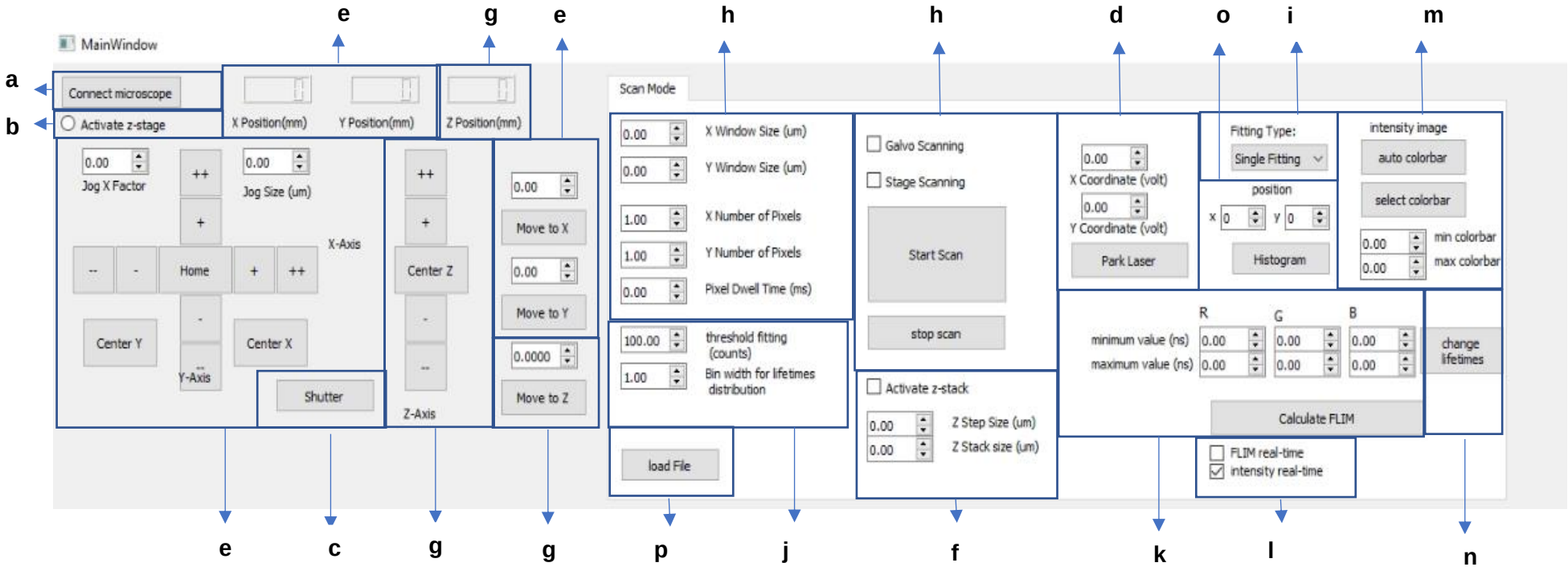

Figure 4.1: Interface main window of the ExtreMed microscope software.

a Button to initialize the microscope devices.

b Check box to activate the z-piezo.

c Button to turn on and turn off the shutter.

d Input fields for the X and Y coordinates in volts, which set the galvanometric mirrors into the desired angle, after clicking the "Park Laser" button.

e x-y microscanner accessible functionalities with position reader.

f Check box that activates the z-stack to acquire 3D images, and fields to input size and step of images between each plane.

g Buttons to perform jogs with the z-stage and to move the z-stage to a chosen position, with the respective position reader.

h Scanning parameters (window size, number of pixels, pixel dwell time, and device), initiating and cancelling scanning buttons.

i Field to choose the fitting algorithm for the data analysis.

j Threshold for fitting and bin width of the distribution lifetime histogram.

k Fields to perform lifetime splitting and initiating button for FLIM.

l Fields to choose the image to observe in real-time during the scanning.

m Fields to choose the colour bar for real-time intensity imaging.

n Button to change lifetime ranges for the lifetime splitting in real-time.

o Coordinates and button to obtain the histogram from a chosen pixel.

p Button to load recorded data for subsequent data analysis.

4.1 Device Control Software

4.1.1 Control functions for the shutter

The laser path from the source to the sample needs to be blocked while not measuring, therefore minimizing the exposure and photobleaching, which is assured by placing the shutter in the path. The user can open or close the shutter on or off by using the computer, as depicted in Figure 4.2.

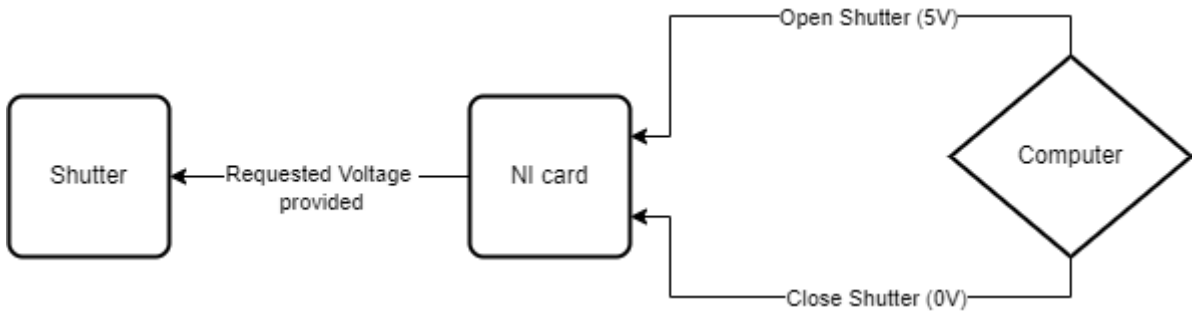

Figure 4.2: Schematic view of the connections between the computer and the shutter.

There are two functions in the software to control the shutter (Table 4.1). Since the voltage to open the shutter is always the same, Booleans were used, which open or close the shutter by providing a fixed voltage (5V) when True or no voltage at all when False. It typically takes 4.8ms (jitter+ open/close time) to control the shutter. The interface window has the button “Shutter” (Figure 4.1c) that changes the Boolean and then executes the associated function to that Boolean value, each time it is clicked.

Table 4.1: Functions that control the Shutter

Accessibility	Function Name	Description
Boolean (True)	Open_Shutter	NI card sends a TTL signal (5V) to the shutter, opening it.
Boolean (False)	Close_Shutter	NI card does not send any signal to the shutter, keeping it closed.

4.1.2 Control functions for the galvanometric mirrors

Two galvanometric mirrors are used to perform the scanning pattern for beam scanning, one for the fast axis and the other for the slow axis. During a scan, the fast axis repeats its movement path multiple times while the slow axis only executes the scanning once at a time (additional explanations of the fast axis and slow axis are given in Subsection 4.1.8). The fast axis is represented on the X-axis and the slow axis is represented on the Y-axis in images.

Each galvanometric mirror accepts an input voltage in the range of –10 to 10 V, which defines the range of maximum deflection angles. Zero volt corresponds to the middle position when the laser is perpendicular to the back aperture of the microscope objective and therefore to the sample. The deflection of the galvanometric mirrors is controlled by a conversion factor of 0.8V/º.

A conversion between the angles of the galvanometric mirrors and the position of the sample is required to move the excitation spot to the desired position in the focal plane. Equation (4.1) performs that conversion. The voltage to be provided to the galvanometric mirrors (V_0) to move the laser the desired distance in millimetres (D_d) equals the voltage being currently provided (V_c) plus that desired distance multiplied by a calibration factor constant, cf:

$$V_0 = V_c + D_d \times cf. \tag{4.1}$$

The calibration factor cf can be obtained either by performing calculations that are dependent on multiple factors such as the magnification of the telescope and the focal length the used microscope objective. Or by imaging a known structure size in the focal plane and relate the voltage needed for a deflection angle of the galvo mirrors able to image the structure. A structure of 50 um diameter was imaged and from the experiments with a 40x objective, a $cf = 10$ was used. Figure 4.3 shows scans using the x-y microscanner a), which is used as a reference, and galvanometric mirrors b), respectively, on the two-photon polymerization structure with 50 µm of diameter. As seen from the obtained images a conversion factor cf of 10 is not completely accurate, and a slightly decreased area is scanned, and therefore the conversion factor should be further and more accurately calibrated. If the microscope objective is changed to another magnification, a new calibration test should be conducted.

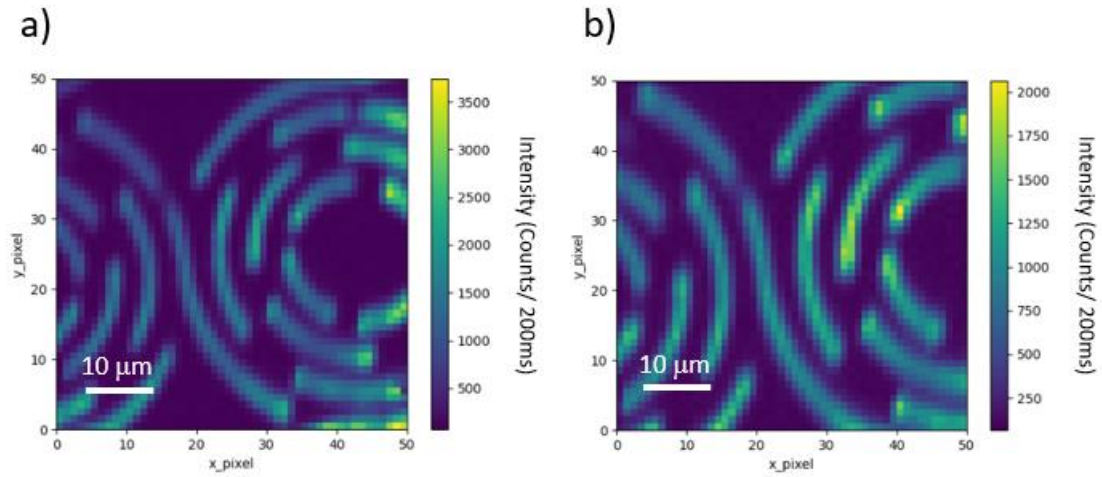

Figure 4.3: Intensity image scans, using the (a) x-y microscanner, and (b) galvanometric mirrors on a two-photon polymerization structure sample with 50 µm of diameter in a window of 50x50 µm, 50x50 pixels, and a pixel dwell time of 200ms.

The control of the galvanometric mirrors is schematically represented in Figure 4.4.

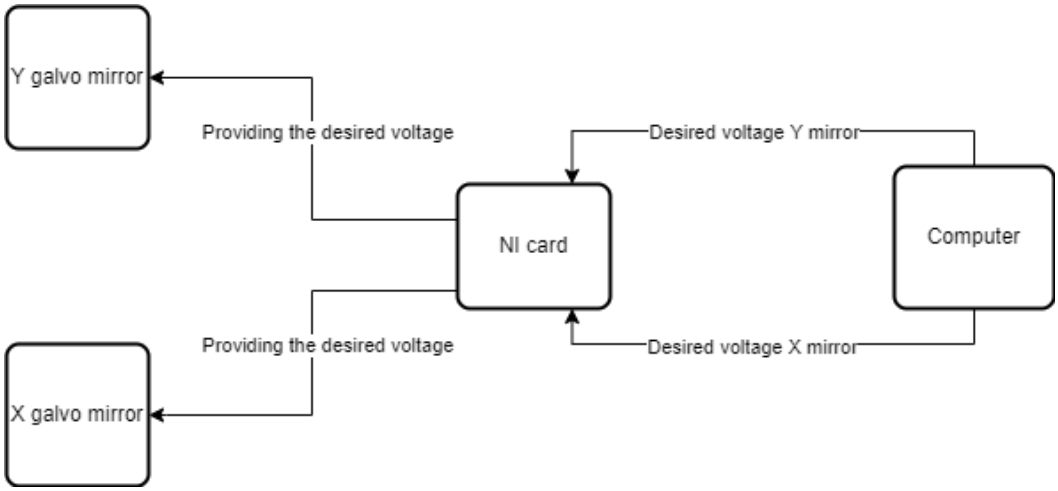

Figure 4.4: Schematic representation of the control of the galvanometric mirrors.

The interface main window has the fields to provide the x and y coordinates in volts, which sets the galvanometric mirrors to the desired angle, after clicking the “Park Laser” button (see Figure 4.1d). The unit of voltage is chosen to be independent of the used objective and therefore the conversion factor. In the current version of the software the conversion factor is coded in, but should in future implementation be accessible for the user, either by direct input of numbers or a selection switch for pre-calibrated objectives.

4.1.3 Control functions for the x-y microscanner

The x-y microscanner serves multiple purposes on the microscope, it moves the sample before scanning, and it is used to perform scanning.

The x-y microscanner sends messages to the software, indicating whether it is in motion or at rest. A new task can be initiated when the software receives a message indicating that the motion has stopped. This is useful to avoid time losses between the x-y microscanner movement and the beginning of a measurement process.

In the interface main window, the user has control over the x-y microscanner, except for changing velocity and acceleration values (Figure 4.1e). Table 4.2 lists the user accessible functionalities of the x-y microscanner.

Table 4.2: x-y microscanner accessible functionalities.

Accessibility	Function name	Description
Value (µm)	Jog size	Selects the distance of the move by the x-y microscanner each jog.
Value	Jog factor	Selects the factor to be multiplied by each jog when performing a big jog
Button with pre-defined values	Jog (x or y) (+/-)	Moves the x-y microscanner a selected jog size.
Button with pre-defined values	Big Jog (x or y) (++/--)	Move the x-y microscanner a selected jog size, times the jog factor.
Button with pre-defined values	Home/ Center x/ Center y	Moves the x-y microscanner to the central position
Value (µm)	Move to (x or y)	Being the position (0,0) µm the bottom left corner, moves to a selected position

It was noticed that the jog function in the x-y microscanner library is not very accurate, usually missing the desired position with an error of 1 µm (see Figure 4.22 in Subsection 4.2.3). Hence, the jog function from the library was not used, instead, I created a very accurate jog function based on the “Move to” function. A scan with this problem fixed is shown in Figure 4.17 for comparison. This self-made jog function considers that each jog is equal to the current position of the x-y microscanner plus the desired distance.

The x-y microscanner moves according to its own specific units but the user can specify the intended window size of a scan or movement steps using S.I. units. The software converts these units into device units. Table 4.3 presents the correspondence between these two measures.

Table 4.3: Correspondence between x-y microscanner units and S.I. units.

Features	Device units	S.I. units
Position	20000	1 mm
Velocity	13744	100 mm/s
Acceleration	13421769	1000 mm/s^2

The buttons used to move and the reader of the position of the x-y microscanner are presented in Figure 4.1e.

4.1.4 Control functions for the z-scanning stage

It is important to have a precise focusing in the sample for obtaining optimal results, which is found by finding the best place for the sample in the z-axis. The movement of the sample in the z-plane is also crucial to acquiring 3D images. Therefore, the sample needs to be moved in the z-plane until it reaches the focal plane, which is determined by finding the position which the MultiHarp record the most counts.

The z-piezo axis is defined from -1 mm to 1 mm, such that the centre of stage is in the middle of the traveling range. To ease the user experience, the definition of the centre of the axis was changed, such that 1mm now equals 0mm, which is the lowest point possible, and -1mm equals 2mm, which is the farthest point possible. This new axis scale is applied to all features of the z-piezo.

The z-piezo provides a library with all the functions to turn the stage on and off and sets the velocity and movement to a defined position. To activate the z-stage, a check box is used (Figure 4.1b). 3D images are acquired by checking the “Activate z-stack” checkbox (Figure 4.1f). The laser focus is found by moving the z-piezo (Figure 4.1g).

4.1.5 Control functions for the TCSPC electronics

In TCSPC, the photon arrival time histogram must be collected for each pixel. TCSPC electronics records the arrival time of each detected photon in relation to the corresponding excitation pulse. The counting hardware is the PicoQuant's MultiHarp 150N, which has a resolution of 80ps and has the following modes: TTTR 2, TTTR 3, and histogram mode. The TTTR 2 mode records the time of detection for each photon from the start of the measurement. To do TCSPC in this mode, two channels are used in the current implementation, but more detection channels can be added in the future. The first channel detects the synchronization signal generated by the reflection of the laser and measured the fast photo diode. This signal will reflect the true repetition rate of the laser. The other input channel counts the detected fluorescence photons. For this mode, the times recorded for the laser and the photons are correlated. For TTTR 3, the correlation is previously established, providing the photon times after the excitation pulse.

A histogram can be created with the photon times using the TTTR modes.

When using the histogram mode, the MultiHarp provide that histogram already created to the software. Hereafter, we focus on the histogram mode, which provides all the necessary information needed for data analysis and thereby after appropriate fitting create a FLIM image.

Figure 4.5 shows a histogram from a spot where a fluorophore was present.

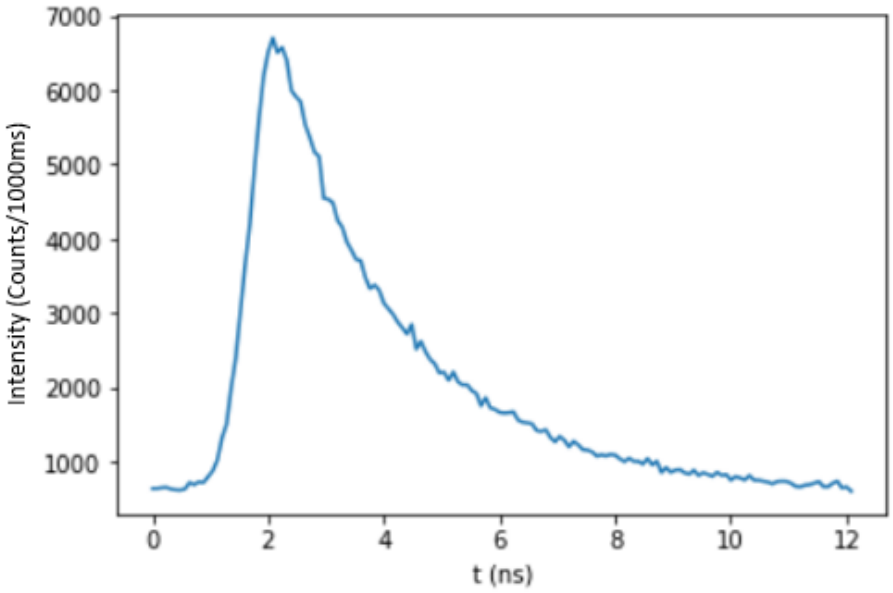

Figure 4.5: Lifetime histogram of a fluorophore provided by the MultiHarp 150N,

The MultiHarp library is very extensive, and to simplify the device usage, there was developed two main functions. One is used to connect the software to the MultiHarp, which is executed when pressing the button in Figure 4.1a. This function verifies if the MultiHarp was successfully connected, providing information about the model used and if there is a synchronous rate from the laser received. Only after the MultiHarp is connected, the second function can be executed. This function has the objective to clean the memory from the previously recorded histogram, and to then start recording a new one, being executed every time a pixel is measured.

At the end of each scan, the software produces a text file containing the histograms of all pixels, with a header with useful information (number of pixels, window size, pixel dwell time, synchronous rate, z position and number of bins). The MultiHarp can use up to 4 detectors simultaneously, hence the software can save a text file for each detector, identifying in each file the channel to which the data corresponds. The data in each file is constituted by histograms from all the pixels. Each pixel histogram is in a different line, with the bin values of the histogram separated by a comma. Since the synchronization pulse rate of the laser, $Sync$, is around 80MHz, the detection interval is 12.5ns, as given by Equation (4.2).

$$Detection\ interval\ = \frac{1}{Sync} \simeq 1.25 \times 10^{-8} = \ 12.5\ \text{ns}. \tag{4.2}$$

As the MultiHarp has a minimum time resolution of 80ps, and with a 12.5 ns between laser pulses there is a maximum of 156 bins with detected fluorescence photons:

$$N_{bins} = \frac{Detection\ interval}{Resolution} = \frac{12.5\ ns}{0.08\ ns} \simeq 156. \tag{4.3}$$

Even though the MultiHarp can create histograms with 65,500 bins, only 156 bins are used and contain information, resulting in smaller and more tractable files (the file size of a scan is reduced from 3 Gigabytes to 3 Megabytes!).

4.1.6 Development of scanning patterns

Before starting the scanning with either the x-y microscanner or the galvanometric mirrors, it is necessary to define several parameters, such as the window size, number of pixels, and pixel dwell time. These parameters are inputs in the interface main window (Figure 4.1h) and must be selected before starting a scan. During the scan, the “stop scan” button can

be activated and stops the scan and moves the x-y microscanner or the galvanometric mirrors (depending on which device is being used) to the initial position where they were when the scan started (Figure 4.1h).

4.1.7 Slow and fast axis

When doing a measurement, the sample is fully scanned when all pixels are measured for the duration of the pixel dwell time. The fast axis (x-axis), as the name suggests, is the axis that moves faster. So, when measuring, the laser stays in the first line (y-axis) until all pixels in that line are measured. After the fast axis completes its movement in the corresponding line, the slow axis moves to the next line, where the fast axis starts repeating its movement. This process is iterated until all pixels are measured. Figure 4.6 illustrates this sequence.

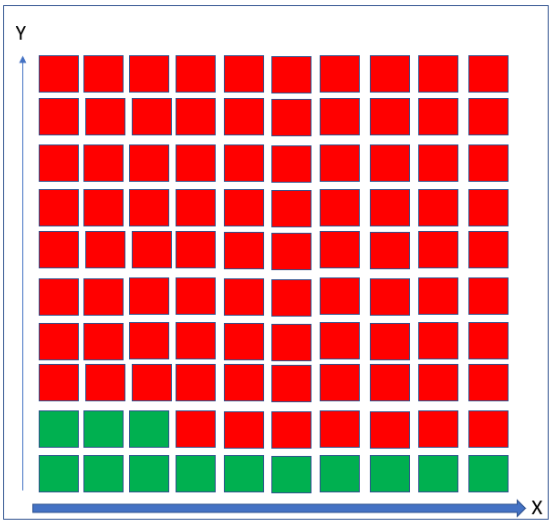

Figure 4.6: Schematic view of a scan. The green pixels are already scanned. All pixels from the first y line were scanned before going to the next y line. x is the fast axis and y is the slow axis.

4.1.8 x-y microscanner and galvanometric mirrors scanning pattern

The basic concept behind the x-y microscanner and galvanometric mirror scanning patterns is the same. The concept is to measure the sample sequentially, point-by-point, i.e., pixel-by-pixel, and not continuously. After knowing the number of pixels and the scanning area, it is possible to know the position of each pixel, aligning the laser with the sample. In each line, the first pixel is the starting position of the respective line scan, the position of the second pixel

equals the position of the first pixel plus the distance from the second to the first pixel, the position of the third pixel equals the position of the second pixel plus the distance from the third pixel to the second pixel, and so on. The distance between each pixel is the row size in S.I. units, divided by the number of pixels in each row. How each pixel position is obtained is represented in Figure 4.7, and Figure 4.8.

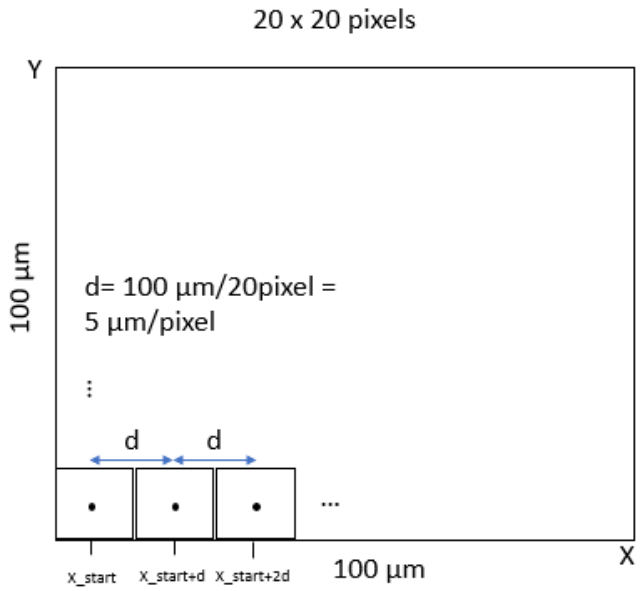

Figure 4.7: Schematic view of the pixel position in S.I. units in a scanning area of 100x100 µm with 20x20 pixels, and the relation between the number of pixels and resolution.

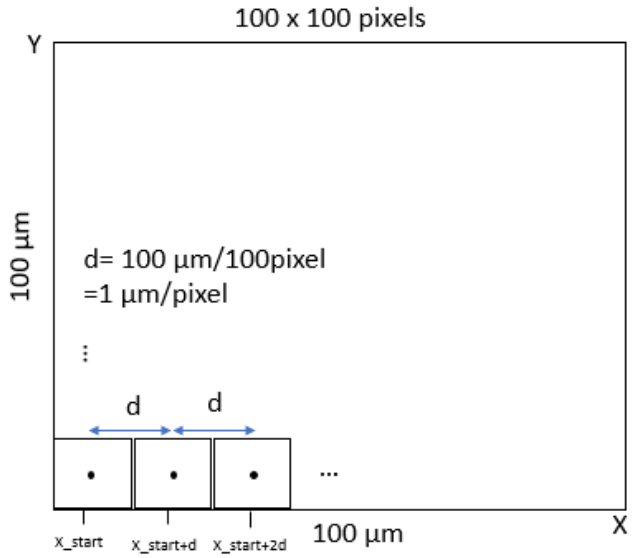

Figure 4.8: Schematic view of the pixel position in S.I. units in a scanning area of 100x100 µm with 100x100 pixels, and the relation between the number of pixels and resolution

However, the devices receive different information sets. The instructions are given in device units and voltage for the x-y microscanner and galvanometric mirrors, respectively. Scanning using the galvanometric mirrors is faster, as it needs almost no settling time due to its low weight compared to the stage and reduced movement to change the laser position. This was verified while doing experiments and controlling the final time for each device with the same scanning parameters.

The continuous scan was also studied but not yet implemented. It could be applied by setting a continuous movement with a given velocity, depending on the window size, number of pixels, and the pixel dwell time. The velocity that a stage had to achieve would be the size of each pixel divided by the pixel dwell time. In a continuous scan, the distance between each pixel is computed as the window size divided by the number of pixels. Since the x-y microscanner takes some time to achieve the desired velocity, it would stay more time in the first pixel than it is supposed to achieve a continuous scan. For the galvanometric mirrors, this would not be a problem, but since the software has the possibility of interacting with the data analysis tasks in real-time, this would require a computing thread between the galvanometric movement functions, the user interface, and the data analysis.

No matter the resolution, i.e., number of pixels in the image chosen by the user, the laser spot size in the focus does not change. So, scanning, with an image pixel size greater than the laser spot size means that the actual measuring points are far apart and does not gather information from the entire image area. If a pixel size is chosen that is comparable to the spot size, the information from the entire image area is collected. If a pixel size smaller than the spot is chosen, they will be overlap between the individual pixels in the image. It must be notes that an increased number of pixels would also potentially lead to an enhanced possibility of photobleaching,

When positioning the sample before a scan, the position of the laser is in the centre of the scanning window. However, to start scanning, the laser must start scanning from the bottom left corner of the scanning window, as seen in Figure 4.9. To do this, either the x-y microscanner or the galvanometric mirror (depending on which is doing the scan) must move halfway the window size of x and y in the negative direction on both axis.

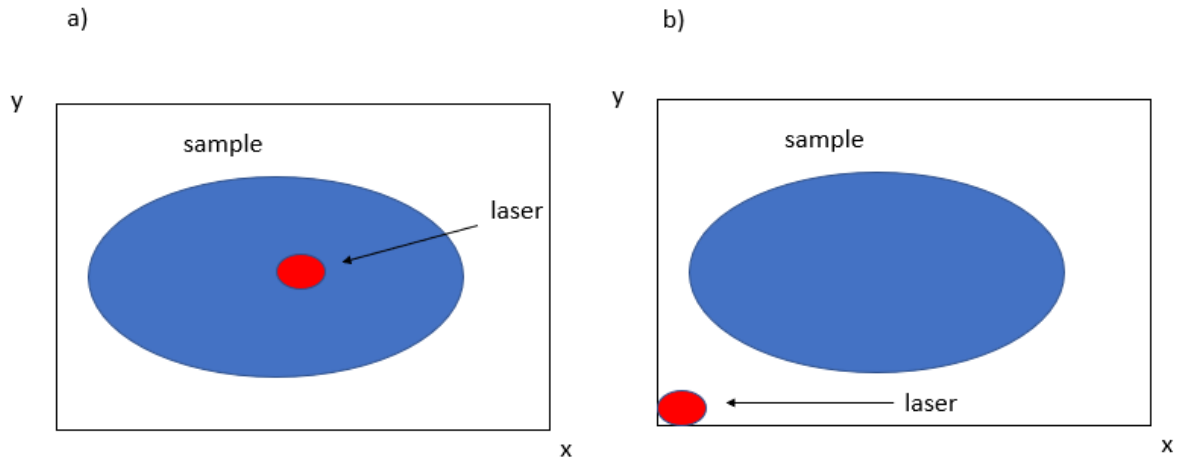

Figure 4.9: Schematic view of the start of the scan after finding a location to get imaging. a) Desired centre of image was found. b) Scan starts from the chosen place.

4.1.9 Pixel dwell time

.

The pixel dwell time is the time during which the software acquires information on each pixel.

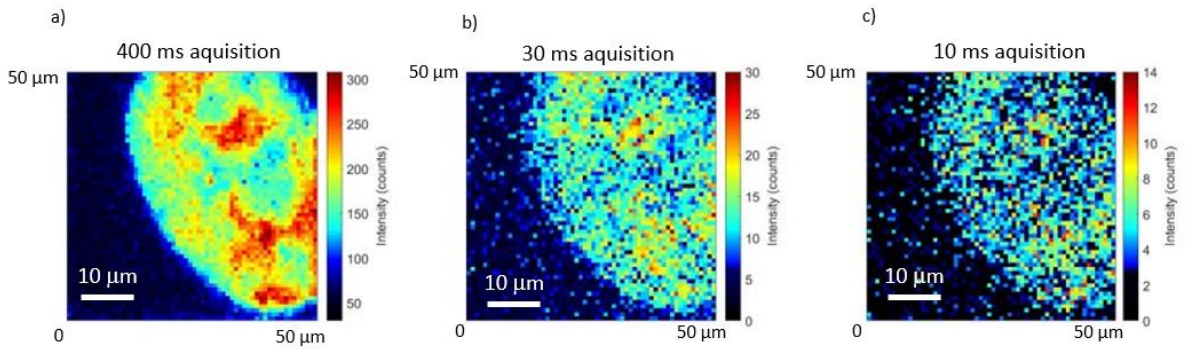

Figure 4.10: Intensity image of a cell with a different pixel dwell times in a 50 µm by 50 µm. The dwell times are a) 400 ms, b) 30 ms, and c) 10 ms.

With higher pixel dwell time, it is obtained a better signal-to-noise data, but the scanning time increases. If the data has bad signal-to-noise ratio, resulted of low pixel dwell time, it might not be able to be used it on data analysis. Figure 4.10 clearly shows the effect of dwell time on the image contrast. Hence, it is critical to select a pixel dwell time that allows enough information in the fastest time possible.

4.1.10 Z-piezo for 3D imaging

The z-piezo moves the sample in the z-axis, allowing the scanning on different depths, and therefore creating the ability to produce 3D images. These images are constructed by stacking recorded xy images at different z-positions. To obtain 3D imaging, the range in the z-axis and the distance between each z-plane must be selected. The sequence starts by acquiring the data from the lowest z-plane and then moves the z-piezo the computed distance each time until reaching the last feasible z-plane.

The number of scans done corresponds to the selected depth to do the 3D image, divided by the distance chosen between each slice plus one for the zero or focus position chosen as the starting position. This number is rounded down to the nearest integer to not exceed the chosen limit. The software developed during this thesis' work is not able to recreate the final 3D images of the scanned xy slices after the scan. To obtain the 3D image, an external software must be used. There is several external software that can combine the xy data from the text files saved by the ExtreMed software, using additional information on the z-plane positions. To obtain 3D data, the user must have the check box in Figure 4.1f checked, with the total 3D height and interval between slices.

Figure 4.11 shows two 3D images from the same sample. The left one is a stack of intensity images, and the right one is a stack of FLIM images of the same sample. The data from each stack was uploaded to an external software named NanoPhotonicsToolbox, developed by Ricardo Adão in the ultrafast Bio- and Nanophotnics group at INL. Obtaining this data and the loading it into the NanoPhotonicsToolbox software, was performed by Leonor Ribeiro. It is demonstrative of how flexible and complete the recording features of the developed software for the SyncRGB-FLIM microscope is.

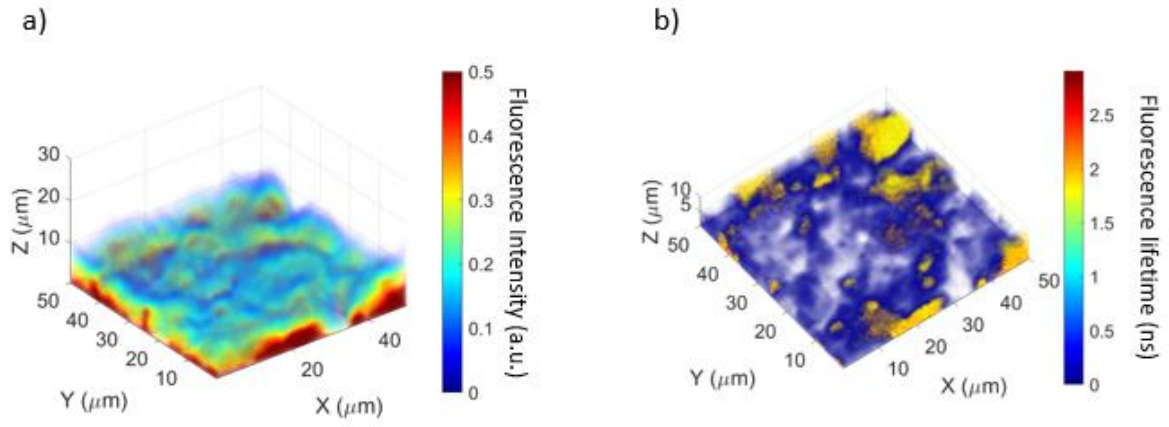

Figure 4.11: 3D images of dorsal root ganglion in collagen from stacking z-planes with different depths using an external software. a) z-stack of intensity images. B) z-stack of FLIM images using a multiexponential fitting algorithm. The images have a 3D height of 10 µm and a step size of 1 µm.

4.1.11 Discussion

The development of adapted functions based on the initial device libraries simplified the device control and creates the possibility to easily change the existing functions or add new device control functions without the need to study the initial documentation.

A problem with the x-y microscanner scan, due to line shifting, was fixed successfully. Understanding that the problem was related to the jog function in the device library was not a trivial task and took some time. This problem was detected when looking at the reader of the x-y microscanner it was noticed that the final position after a jog was not the correct one.

Devices without libraries, such as the galvanometric mirrors and shutter, have now a mechanised control, avoiding the need for the user to know the necessary voltage to manage these devices. To convert the galvanometric mirrors units into voltage, a calibration factor was successfully found using a trial-and-error procedure in a two-photon polymerization structure. This factor is not 100% accurate, showing a decrease in the area scanned.

Scanning using the galvanometric mirrors showed to be faster than the x-y microscanner when timing the total time, due to its reduced movement to change the laser position and its low weight/mass. This is important for point-by-point, i.e., pixel-by-pixel, scanning.

Even though continuous scanning would remove the delay times between pixels, in the scope of this thesis the disadvantages of this technique would be more than its benefits. Since

the x-y microscanner takes some time to achieve the desired velocity, in continuous scanning it would stay more time in the first pixel than the supposed integration time. This could be fixed by increasing the scanning window with a margin that is not scanned, to assure that the x-y microscanner reaches the desired velocity. For the galvanometric mirrors, this would not be a problem, but since the software has the possibility of real-time interaction with the data that is being measured, this would require a computing thread between the galvanometric movement functions and all the other real-time functionalities. This is because the function that moves the galvanometric mirrors, the function that records the data, and the window would need to be all running simultaneously. After the implementation of the threading and giving a margin to scans with the x-y microscanner, the software should adapt to continuous scanning, which is the most common in multiphoton microscopy, increasing the performance (Pilger et al., 2021; Kumazaki et al., 2007).

The data recorded from the MultiHarp 150N is well identified and easy to use since it is recorded in text format. This data was already used in external software to create 3D images of intensity and FLIM. Since the actual number of bins where optimized to the actual repetition rate of the laser, the files size were reduce drastically, easing its management. The option to produce 3D data was also implemented with success, with the flexibility required for producing 3D imaging in external software afterward.

4.2 Data Analysis Software

Besides the software for device control, the thesis' work was also on the development of data analysis software, aiming at providing direct feedback on the imaging quality. The main features are the following: lifetime ranges, which can be changed in real-time, lifetime splitting information, which can be used to assess the quality of the scan, intensity image, and histogram of chosen pixels.

The data analysis can be performed on recorded data or even on external data loaded to the software. All figures are saved in the Portable Network Graphic (PNG) format.

4.2.1 Fitting algorithm for pixel histogram

To obtain the lifetime of a fluorophore for a given pixel, the initial task is the fitting of the curve of the corresponding histogram. This is done by using the Python package named scipy.

When fitting the decaying curve, the curve maximum and the corresponding time bin is chosen as the first bin in the decaying curve, while all-time bins above 12 nanoseconds, the repetition rate of the laser are excluded.

The user can, in the interface main window (Figure 4.1i), chooses the number of exponential functions for the fitting procedure. The user may choose single (one lifetime), double (two lifetimes), or triple (three lifetimes) exponential fitting. More than three exponentials could have been implemented, but it is very rare to find more than three distinct lifetimes in a pixel and difficult to have enough signal intensity for each of the lifetimes to distinguish them simultaneously.

Fitting is performed using an exponential function. Equation (4.4) shows the fitting function, where $\tilde{A}(t)$ is the position of the decay curve at t nanoseconds, C is the number of components such that $C = 1$ for single, $C = 2$ for double, and $C = 3$ for triple fitting. The weights of the different lifetimes are represented by m_i, and the decay rate for each component is given by Γ_i.

$$\tilde{A}(t) = \sum_{i=1}^{C} m_i e^{-t \times \Gamma_i} \tag{4.4}$$

The lifetimes present in the decay curve, τ_c, with $c = \{1\}$ for single fitting, $c = \{1, 2\}$ for double fitting, $c = \{1, 2, 3\}$ for triple fitting, is obtained by:

$$\tau_c = \frac{1}{\Gamma_c} \tag{4.5}$$

The normalized weight of each lifetime, mn_c, is given by:

$$mn_c = \frac{m_c}{\sum_{i=1}^{C} m_i} \tag{4.6}$$

The weights of the lifetimes are normalized to facilitate the data analysis, providing a better perception of the weight of each lifetime.

The fitting of the decay curve of the histogram shown in Figure 4.5 by a single exponential is highlighted in Figure 4.12.

The fitted curve has a lifetime of 2.16 nanoseconds, which means that the fluorophore has a lifetime of 2.16 nanoseconds. The fitting optimization algorithm requires a lifetime initial

guess, which is always assumed to be 1 ns for all fitting types (this was the value that produced the most consistent fitting results in all experiments).

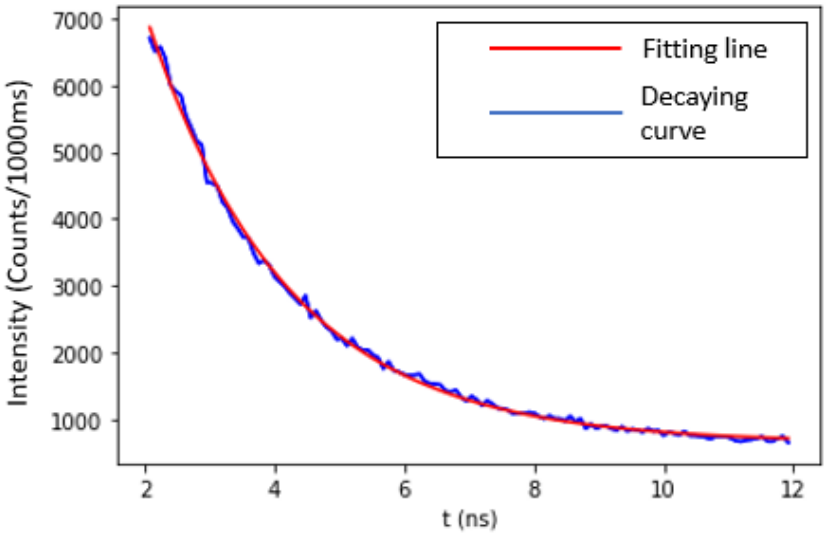

Figure 4.12: Decaying curve (blue line) and corresponding fitting curve (red line), extracting a lifetime of 2.16 ns

If the algorithm tries to fit every pixel in the image, then the algorithm potential also tries to fit very low intensity pixels or background pixels, resulting in non-existing or wrong lifetimes. This can be avoided by defining an intensity threshold, according to which any pixel with an intensity below that threshold is skipped in the fitting procedure (Figure 4.1j).

The current fitting procedure does not benefit from the deconvolution of the IRF but could be incorporated in the software at a later time point. Since the IRF is not taken into consideration, the current fitting of decaying curves with shorter lifetimes below 0.5 ns would be more influenced by the missing IRF deconvolution than curves with longer lifetimes.

4.2.2 Fluorescence lifetime imaging (FLIM).

After the fitting, the lifetimes are separated into the user defined lifetime intervals, selected before the scan, helping the observation of regions of interest (ROI) in the sample associated with a specified lifetime. For effective grouping, it is necessary to select a range around the lifetime, taking in consideration the influence of external factors on the fitting differences, resulting in small differences in the extracted lifetime value. Consequently, the lifetime of a fluorophore is not constant but follow a narrowed distribution around the theoretical limit of 5ns. This distribution needs to be manually selected by the user (Figure 4.1k).

After choosing the distribution limits for the fluorophore, these can be changed during the scanning by changing the image in real-time. FLIM can also be performed at the end of the scanning or on recorded data by pressing the “Calculate FLIM” button after selecting the desired distributions.

When doing the lifetime splitting, one lifetime distribution is represented in red, another in green, and the last one in blue. Lifetimes that are not in the selected intervals or are removed due to low intensity are represented in pink coloured image. After the splitting of the fluorescence lifetimes, an image is created for each chosen lifetime interval where the pixels in each individual image are identified by just one colour, where the brightness ranges from 0 to 1. The brightness is an indicator of the normalized weights, mn_c, of the lifetime present in the pixels. Hence, pixels in the same group have the same colour but can be presented with a different level of brightness. So, a pixel in a chosen lifetime interval with a higher weight will have a brighter colour than the lifetime of a pixel with a lower weight. The weight of the lifetimes in the excluded group, i.e. in the pink image, is equal to the sum of the weight of lifetimes that are not inside any of the selected distributions. So, when no lifetime in that pixel has a colour associated, the excluded pixel will have the maximum brightness or a value of 1 in the pink image.

To illustrate what has been described, let us consider that there is an image with three pixels. Let’s say that ach pixel has a decay curve that is fitted by the triple exponential algorithm.

Firstly, the fitting components are obtained for each pixel (presented in Table 4.4).

Table 4.4: Parameters of the fitting curves.

	$\tau_1(ns)$	mn_1	$\tau_2(ns)$	mn_2	$\tau_3(ns)$	mn_3
Pixel 1 (p1)	2.1	0.80	3.4	0.10	5.8	0.10
Pixel 2 (p2)	1.1	0.50	1.9	0.20	4.4	0.30
Pixel 3 (p3)	2.2	0.65	3.6	0.25	4.0	0.10

Secondly, these pixels are split into groups, by comparing the resulting lifetimes of the fitting with the desired distributions of lifetimes defined by the user, which are: Red [1.8; 2.3] ns, Green [3.3; 3.6] ns, Blue [5.6; 5.9] ns, and Pink (for the rest of the lifetimes). Table 4.5 shows the lifetime split.

Table 4.5: Lifetimes split

	Red (τ, mn)	Green (τ, mn)	Blue (τ, mn)	Pink (τ, mn)
Pixel 1 (p1)	(2.1, 0.80)	(3.4, 0.10)	(5.8, 0.10)	(NaN, 0)
Pixel 2 (p2)	(1.9, 0.20)	(NaN, 0)	(NaN, 0)	(1.1, 4.4); (0.5+0.3)
Pixel 3 (p3)	(2.2, 0.65)	(3.6, 0.25)	(NaN, 0)	(4.0, 0.10)

Note: In pixel 2, two lifetimes were outside of the selected distributions, meaning that the brightness in the excluded group is the sum of the weight of both lifetimes. When NaN there is no lifetime present in that colour range.

The first pixel has a lifetime that lies within the range of colour red. This means that p1 of that group image will have the weight associated with the lifetime that is inside that range. After doing the splitting for all the lifetime ranges, the lifetimes that were left out are associated with the excluded group. If a pixel has more than one lifetime associated, the weights are added up. After the lifetimes splitting, an image for each group is created using the weights as the colour intensity, as seen in Figure 4.13.

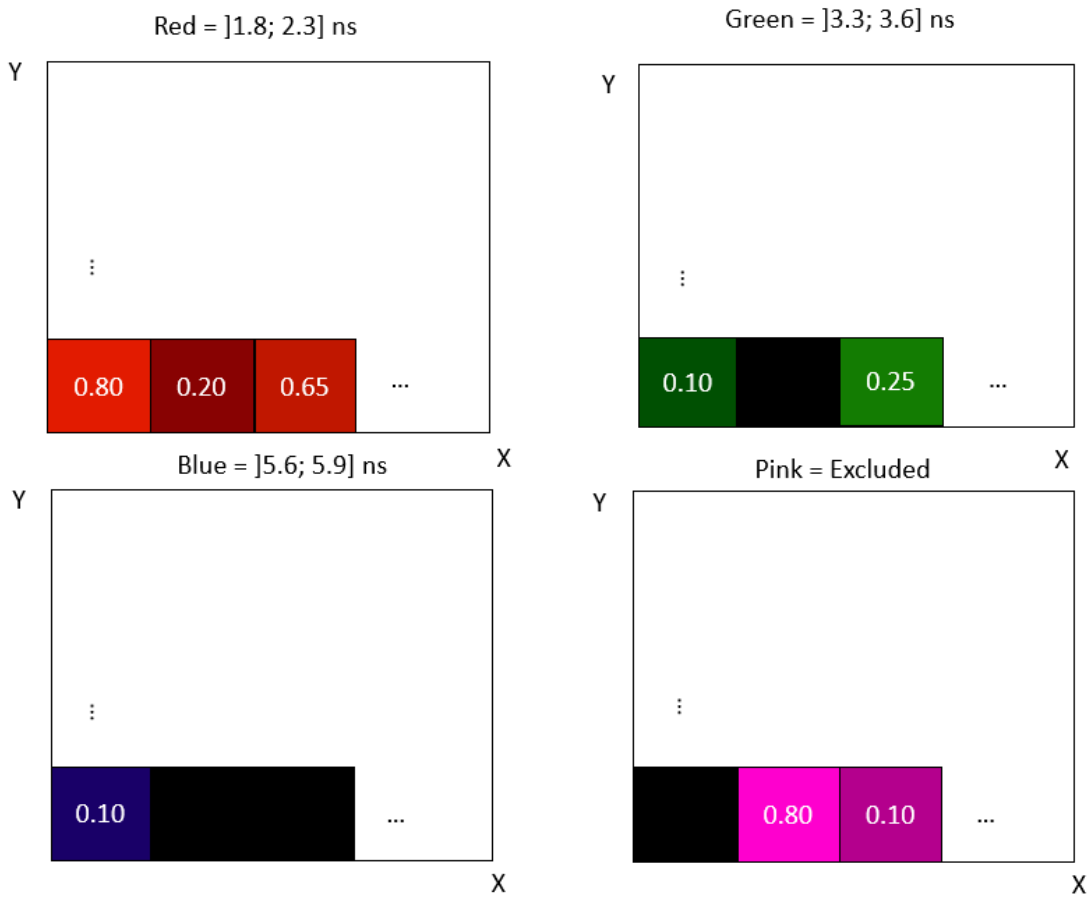

Figure 4.13: A schematic example of lifetime splitting.

A FLIM of a cell from an experiment can be seen in Figure 4.14 and Figure 4.15, with intensity thresholds of 100 and 500 counts, respectively.

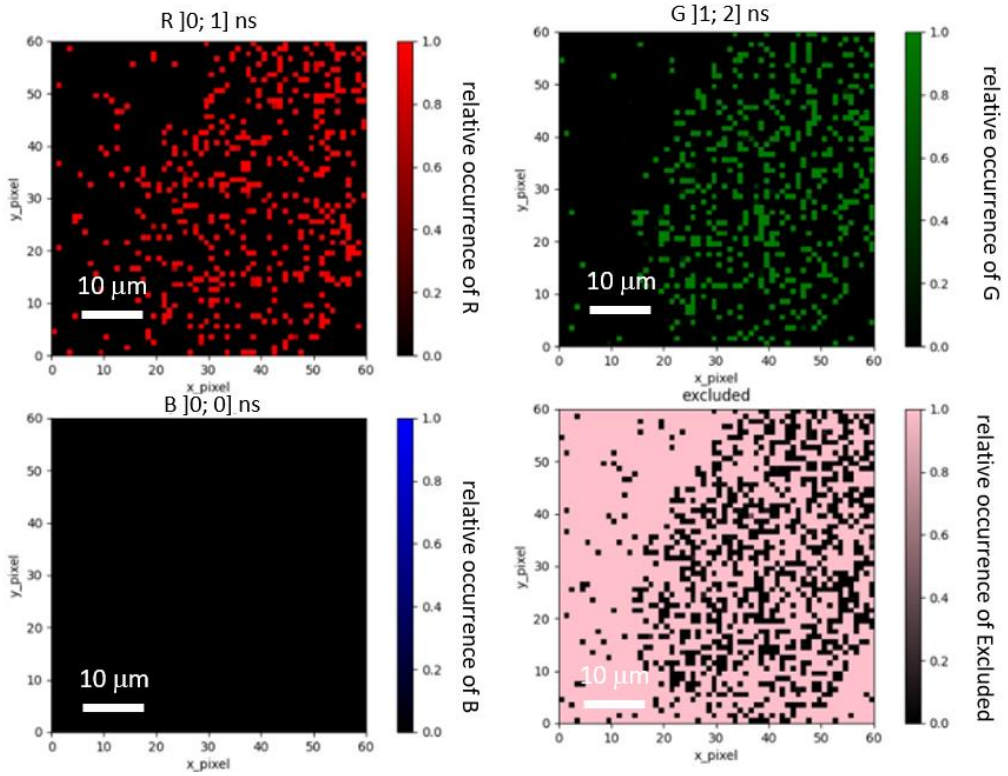

Figure 4.14: Lifetime splitting results for a double exponential fitting, with a filtering threshold of 100 counts. The relative occurrence of fluorescence lifetimes components tau1, within an interval of]0; 1] ns (shown in red), and tau 2]1; 2] ns (shown in green) per pixel.

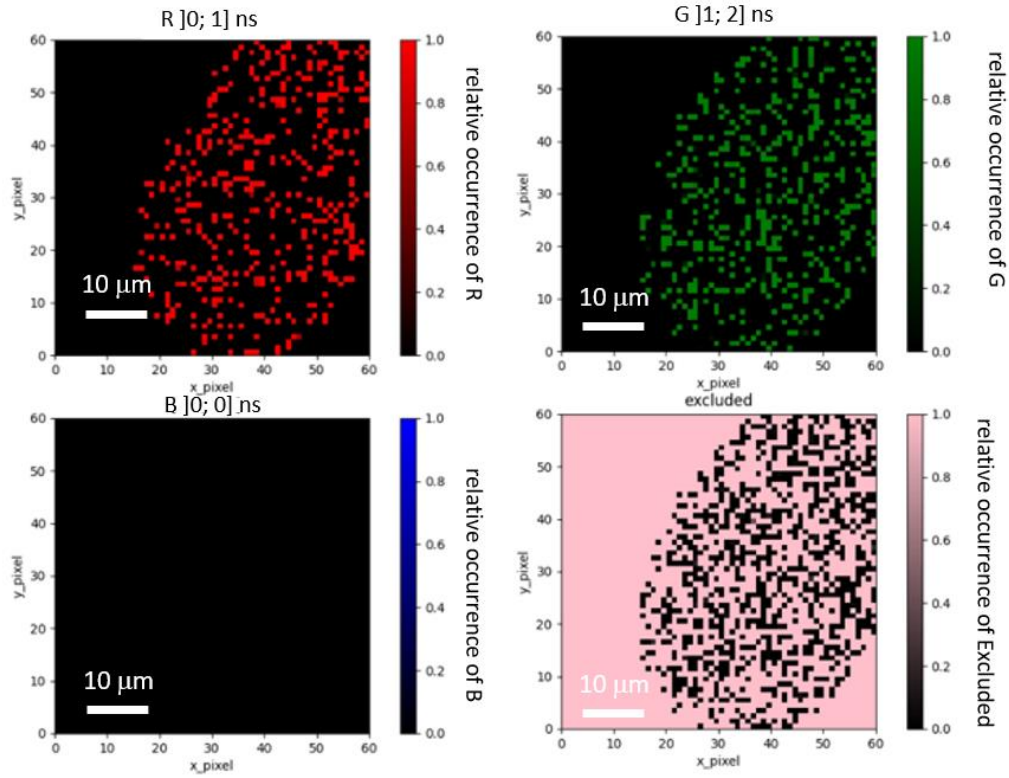

Figure 4.15: Lifetime splitting results for a double exponential fitting, with a filtering threshold of 500 counts. The relative occurrence of fluorescence lifetimes components tau1, within an interval of]0; 1] ns (shown in red), and tau 2]1; 2] ns (shown in green) per pixel.

When scanning samples with unknown lifetimes, it is difficult to select the lifetimes when doing the lifetime splitting. So, after a scan, the user has a frequency histogram of all the lifetimes, from where it can be observed which lifetimes are most present. This works as an aid for deciding which lifetime ranges to use in the lifetime splitting, for post processing the image or for additional scans of the same area. In the frequency histogram all lifetimes are saved with the same weight, even though the weights associated with the fitting might be different. Furthermore, the time bin width of the histogram can be adjusted by the user. An important condition for the creation of a histogram is only accepting lifetimes lower than 12.5 ns, which is the maximum that can be obtained based on the repetition rate of the laser used. If this boundary condition is not present, then the algorithm can potentially try to fit noisy curves returning lifetimes longer than the repetition rate of the laser

An example of a histogram can be seen in Figure 4.16, which was obtained from the data used to create Figure 4.14. The histogram bin dimension can be chosen based on the parameter present in Figure 4.1j.

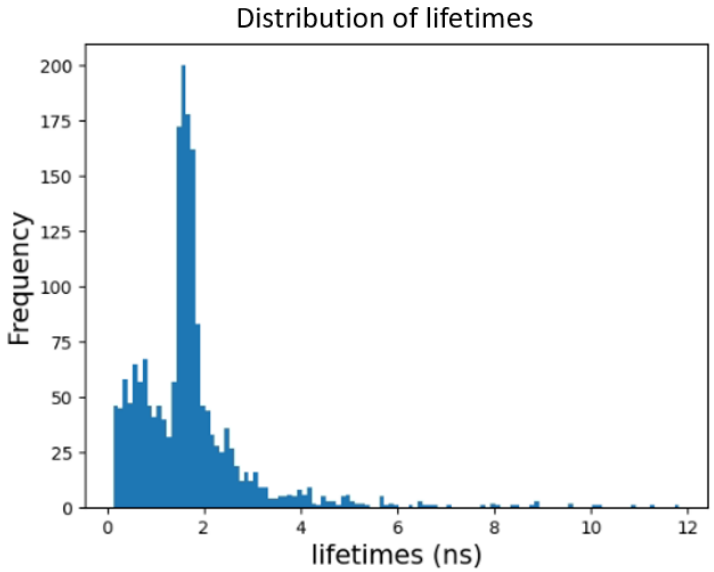

Figure 4.16: Distribution of fluorescence lifetimes of a sample using double exponential fitting and a bin width of 0.10 ns.

A scan was performed in a cell with unknown lifetimes. The fitting algorithm chosen was the single exponential. Based on the intensity image obtained from the scan, presented in Figure 4.17, the background pixels are around 200 counts. With this information, all pixels below 200 counts are not fitted. After fitting all the pixels above 200 counts, a histogram with the distribution of the lifetimes is provided (see Figure 4.18). Based on that histogram, the chosen lifetime ranges were]0, 1] for red,]1, 2] for green, and]2, 3] for blue.

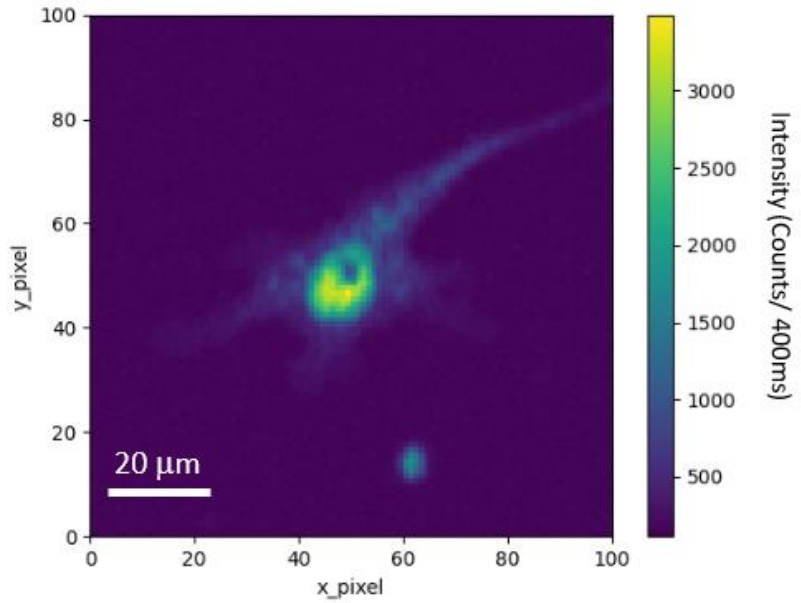

Figure 4.17: Fluorescence intensity image with 100x100 pixels and 100x100 µm showing fluorescence emission of a multi-colour labelled cell.

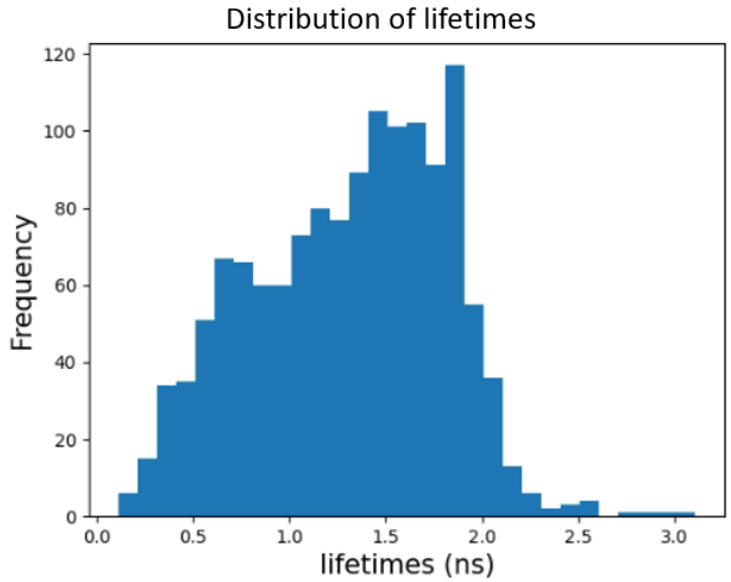

Figure 4.18: Distribution of fluorescence lifetimes, with a bin width of 0.1 ns, determined by single exponential fit of all pixels above a selected intensity threshold of 200 counts, which preselected the pixels belonging to the cell.

The FLIM image with the chosen parameter is presented in Figure 4.19. From the comparison between intensity (Figure 4.17) and FLIM (Figure 4.19), it is clear that the silhouette of the cell has a shorter lifetime than the inside of the cell.

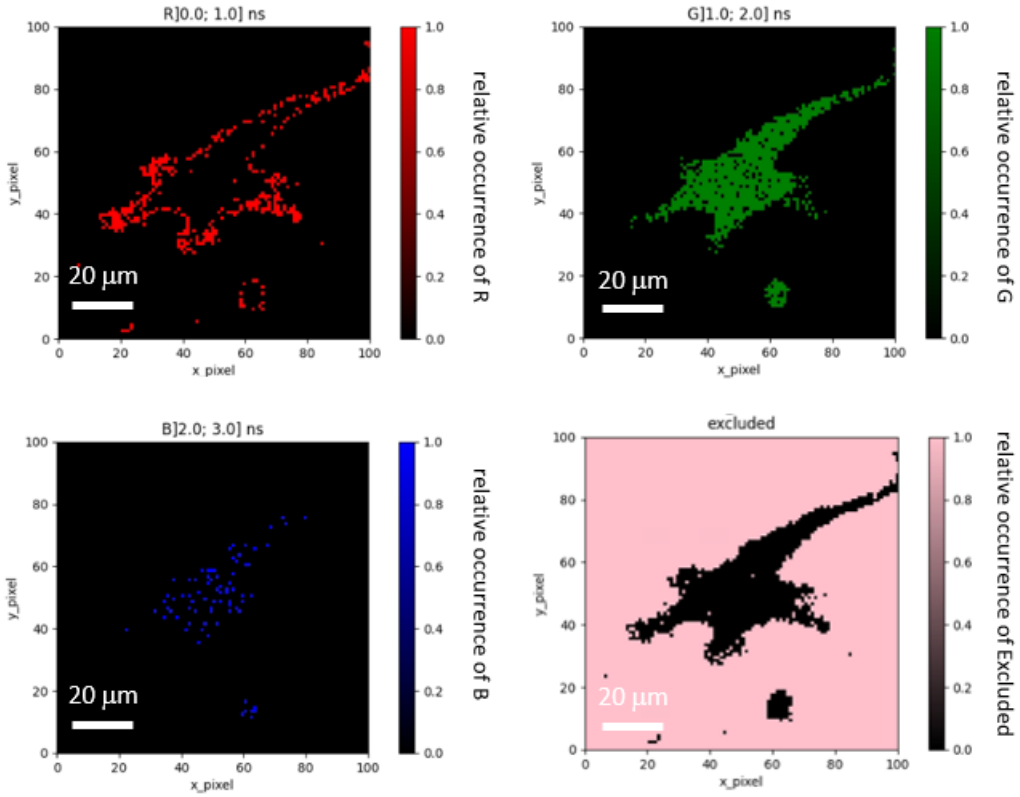

Figure 4.19: FLIM of a cell with unknown lifetimes, using single exponential fitting algorithm.

4.2.3 Real-time imaging of intensity and FLIM

The scanning time for a full image can take from several minutes to hours. So, it is crucial to know in real-time if the scanning was well designed. Real-time imaging allows the observation of the part of the sample already scanned. This is illustrated in Figure 4.20. The user can choose to observe the evolution of the scanning using the intensity image or FLIM (Figure 4.1l). The interaction with the window during a scan is possible due to the line of code “app.processEvents()”. The problem with this approach is the bad performance of the microscope software since this line of code only flushes all queued events, meaning that when the user clicks on something from the window, the information provided by the interaction is instantly sent to the code, pausing the scan until the new command is executed.

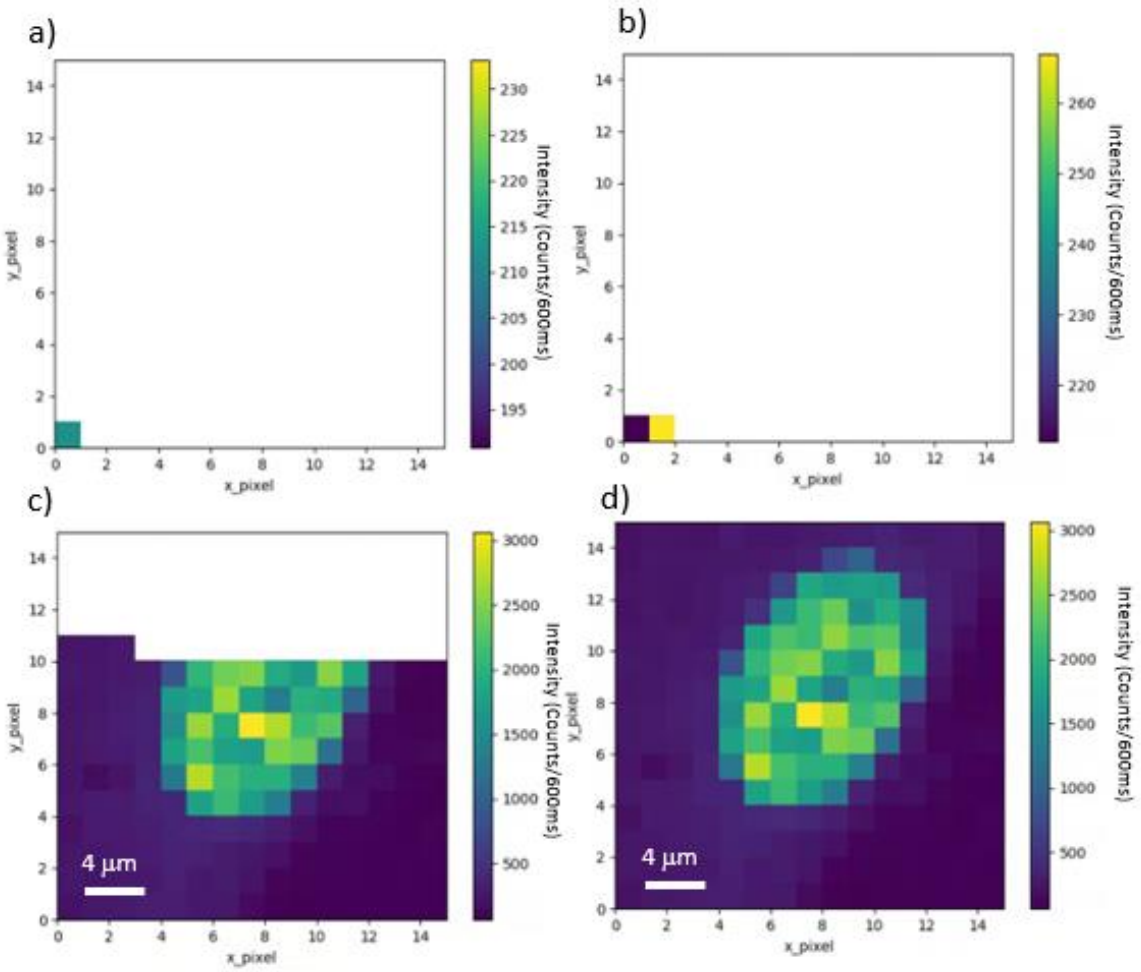

Figure 4.20: Real-time imaging of the intensity, using automatic colour bar (15x15 pixel, 30x30 µm). From top left to bottom right the advances in collected image data during an image scan is shown with the total sum of counts, implemented for both sample scanning and bean scanning modes. Automatic colour bar: a) to b).

Intensity image

For intensity imaging, a matrix is used to record the sum of the values of the time bins for each histogram of the corresponding pixel. This matrix is then plotted in an image where each number has an associated colour, defined by the colour bar. The user decides between an automatic colour bar or a colour bar with chosen limits (Figure 4.1m). The automatic colour bar updates the limits following the minimum and maximum intensity pixels, updating each time a new pixel is scanned. With a chosen colour bar, all pixels with intensity bellow (above) the selected lower (higher) limit of the colour bar are associated with the lowest (highest) colour selected. This means that by selecting a colour bar, the contrast is removed for pixels outside the selected range. This is illustrated by Figure 4.21.

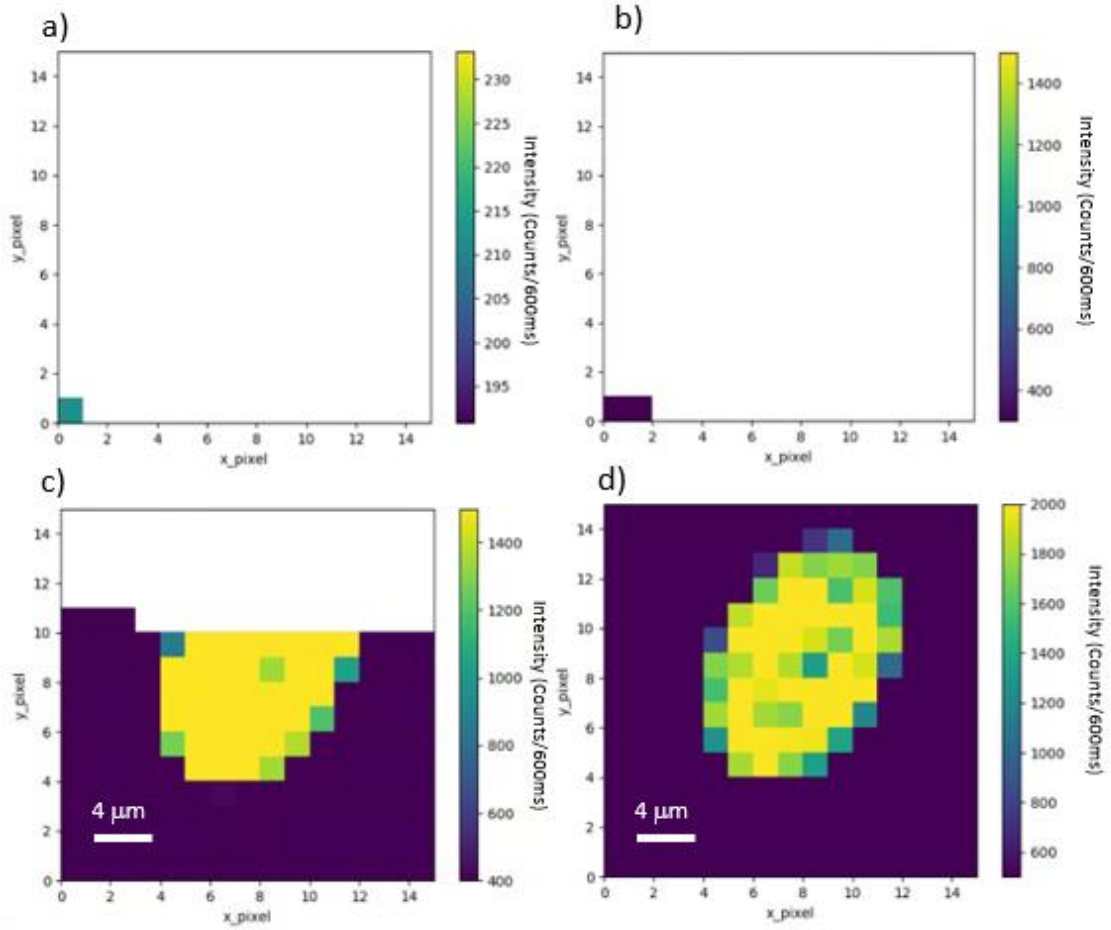

Figure 4.21: Real-time imaging of the intensity, changing the colour bar during the scan. (15x15 pixel, 30x30 µm). From top left to bottom right the advances in collected image data during an image scan is shown with the sum of counts per photon arrival time histogram, and as implemented for both sample scanning and bean scanning modes. Automatic color bar: a) Selected color bar: b) – [300; 1500] counts; c) – [400; 1500] counts; d) – [500; 2000] counts

The automatic colour bar has the advantage of providing a general overview of the intensity image of the sample. With this information, it is possible to put all the background in the same colour by setting the lowest value of the new colour bar to the maximum intensity observed in the background of the image. This procedure is illustrated in Figure 4.22.

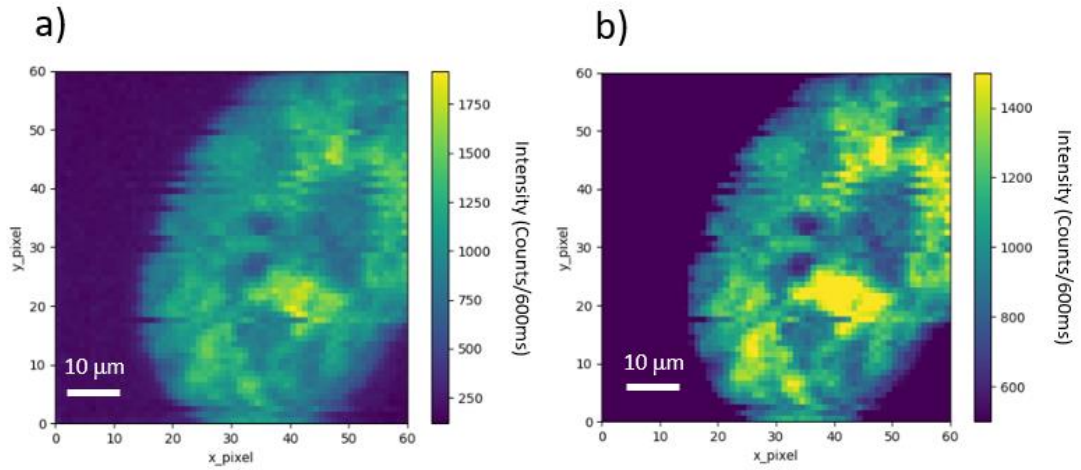

Figure 4.22: Intensity image of 60 x 60 µm and 60 by 60 pixels, using: a) automatic colour bar, with a range of [230,1800] counts, b) a selected colour bar with a range of [500; 1500] counts.

Sometimes, instability in the laser intensity resulted in higher intensity levels than expected for some pixels. This creates a problem in the automatic colour bar image, with a huge colour separation between the new pixel with high intensity and the rest of the pixels in the image, resulting in poor contrast. The image contrast can be recovered by setting the maximum of the colour bar to the maximum value before the faulty pixel.

Selecting between automatic and user-chosen colour bars can be done during or after a scan. This is important because the faulty pixel may be identified during the scan, and fixing it is fundamental to properly observe the data in real-time.

FLIM image

For FLIM imaging, the lifetime ranges for the splitting can be changed in real-time during the scan, updating the FLIM image with new ranges when clicking on the button “Change lifetimes” (see Figure 4.1n).

Both Figure 4.23 and Figure 4.24 were obtained during one single scan. The lifetime ranges were changed during the scanning by updating the lifetimes, which updated the image in real-time.

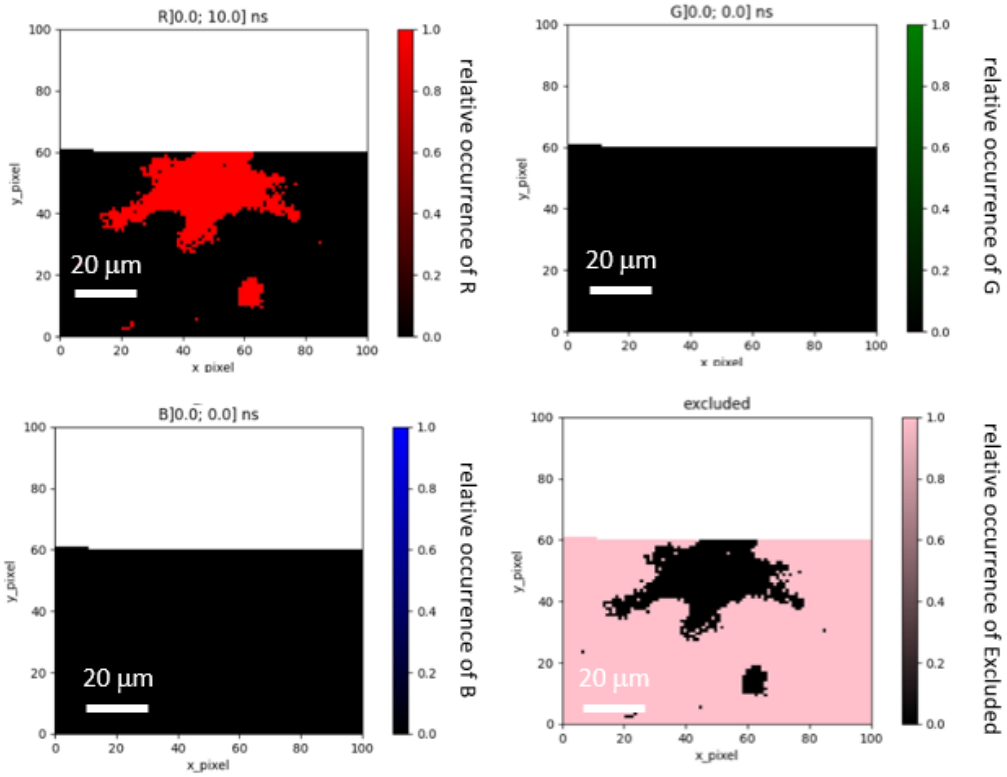

Figure 4.23: Real-time FLIM of a cell, using single exponential fitting, and splitting the lifetimes between]0; 10] ns for the colour red.

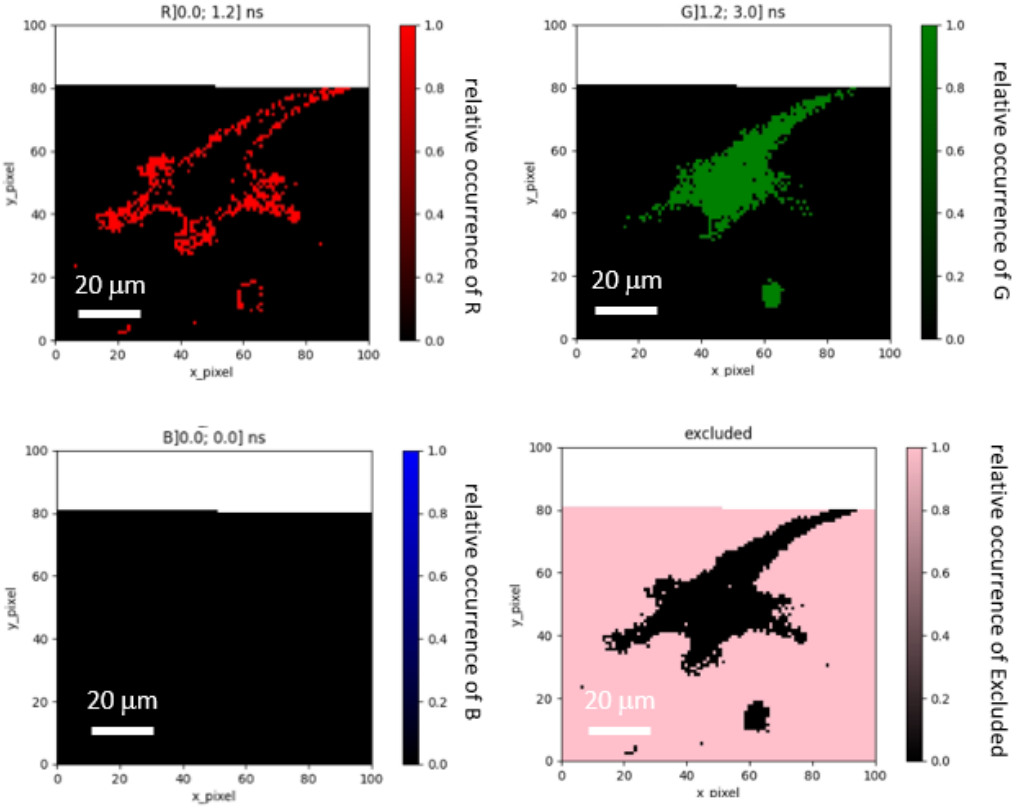

Figure 4.24: Real-time FLIM of a cell, using single exponential fitting, and splitting the lifetimes between]0; 1.2] ns and]1.2; 3] ns.

4.2.4 Histogram of a selected pixel

In the software there is the possibility to choose a pixel by selecting the corresponding position in the image matrix and then provide the associated histogram after pressing the "Histogram" button (Figure 4.1o). The histogram may be shown with the fitting curve and the corresponding lifetimes, as seen in Figure 4.25, or alone when the algorithm cannot obtain a fitting curve, as seen in Figure 4.26.

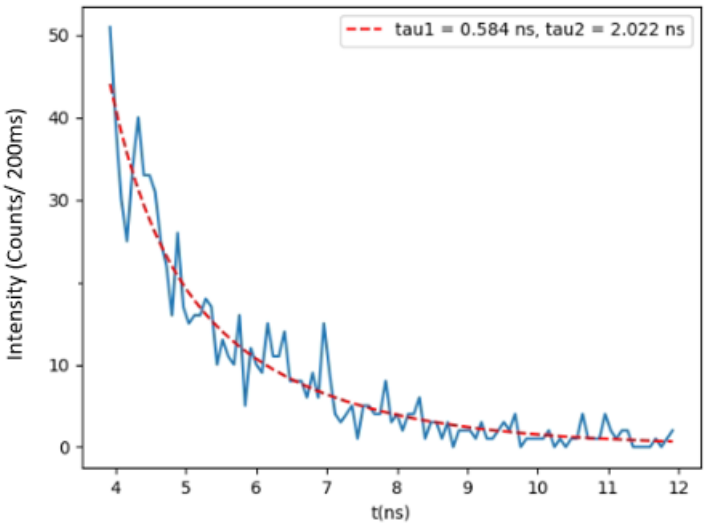

Figure 4.25: Histogram of a chosen pixel with a double exponential fitting curve, having present the lifetime 0.584 ns and 2.022 ns.

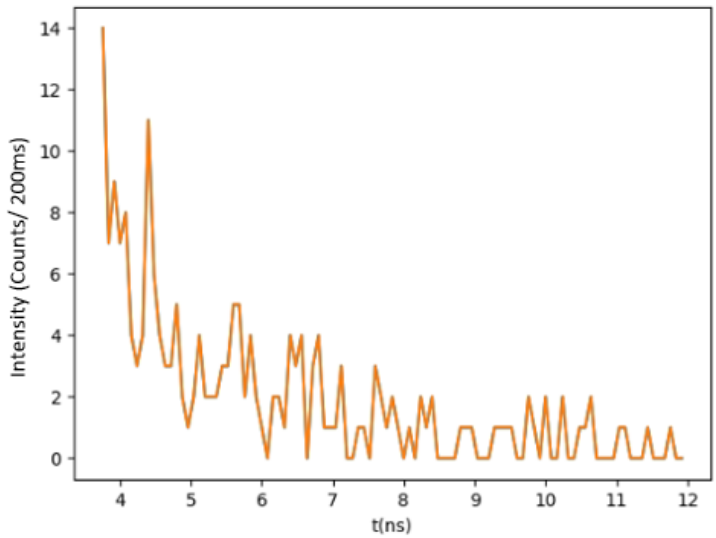

Figure 4.26: Histogram of a chosen pixel in a situation where the algorithm could not obtain a double exponential fit, and no values for the tau´s are returned.

Low pixel dwell time or background signals tend to result in very noisy data, with very low photon counts. In these situations, the fitting algorithm has difficulty in fitting the data even if there are fluorophores present. These curves are a good indication of the fitting quality of the data.

4.2.5 Effective scanning time

When doing the scanning pattern with the x-y microscanner or the galvanometric mirrors there is a short period taken by the hardware and software before the measurement of each pixel starts. In terms of hardware, this is due to the time that the scanner takes to move in the fast axis and slow axis, which also accounts for the resetting of the fast axis when the scanner moves on the low axis. Data analysis and scanning simultaneously have the advantage of seeing the results in real-time. However, it has the disadvantage of introducing a delay due to the time needed by the software to perform data analysis and device control, which take place before measuring the next pixel.

The effective scanning time, $t_{effective}$, is the time that a scan would take without any delays. The effective time corresponds to the number of pixels, N_{pixels}, times the pixel dwell time. Knowing the effective scan time allows the computation of the delay for each pixel. For example, a 4x4 pixel scan with a 1-second dwell time per pixel results in an effective scan time of 16 seconds. If the scan takes a total of 17.6 seconds (t_{total}), then 1.6 seconds are used for the scanner movement and data analysis. The delay between each pixel, $Delay_{pixel}$, is computed by dividing the extra time by N_{pixels}, as seen in Equation (4.7). In this example $Delay_{pixel} = 0.1$ seconds.

$$Delay_{pixel} = \frac{\left(t_{total} - t_{effective}\right)}{N_{pixels}} \tag{4.7}$$

The delay time was important to assess which code segments needed to be optimized. To obtain the time that a scan took to finish, a timer measured the time between the beginning and the end of several scanning experiences. The tests were conducted on a scan with 4x4 pixels, in a window of 100x100 µm, with a pixel dwell time of 1 second, with an effective scan time of 16 seconds.

To study the delays resulting from device control, all features of data analysis were turned off. Table 4.6 shows that the galvanometric mirrors are faster, taking 0.187 seconds to move between pixels, while the x-y microscanner takes 0.484 seconds, almost half a second.

Table 4.6: Total scanning time for the x-y microscanner and galvanometric mirrors.

Scan device	Time (seconds)	Delay per pixel (seconds)
x-y microscanner	23.744	0.484
Galvanometric mirrors	19.006	0.187

Notes: These are the total scanning time for an image with 4x4 pixels, in a window of 100x100 µm, with a pixel dwell time of 1 second.

Galvanometric mirrors were used to study the delays associated with data analysis. The results are shown in Table 4.7.

Table 4.7: Total times for different data analysis, using the galvanometric mirrors.

Features of the scan	Time (seconds)	Delay per pixel (seconds)
Fitting	19.200	0.013
Intensity (image) + fitting	20.919	0.120
FLIM (image) + fitting.	27.859	0.554

Notes: These are the total times for an image with 4x 4 pixels, in a window of 100x 100 µm, with a pixel dwell time of 1 second. The delay per pixel equals the total delay minus the delay of the galvanometric mirror movement, which is –0.187 sec.

The time taken to perform the fitting in each pixel is quite low. However, in real-time imaging of intensity and FLIM the time taken per pixel might reach between 0.741 seconds (0.554+0.187) and 0.307 seconds (0.187+0.120). This is not optimal and might be due to the creation of a new plot after each pixel is scanned. Python has some options to improve the imaging performance by keeping the plot open and just updating the data.

4.2.6 Load data

The software allows the loading of recorded data for subsequent data analysis. After pressing the “Load File” button (Figure 4.1p), a window opens, allowing the user to select a text file from a previous scan or even external data (if the data follows the rules that allow it to be imported), as shown in Figure 4.27.

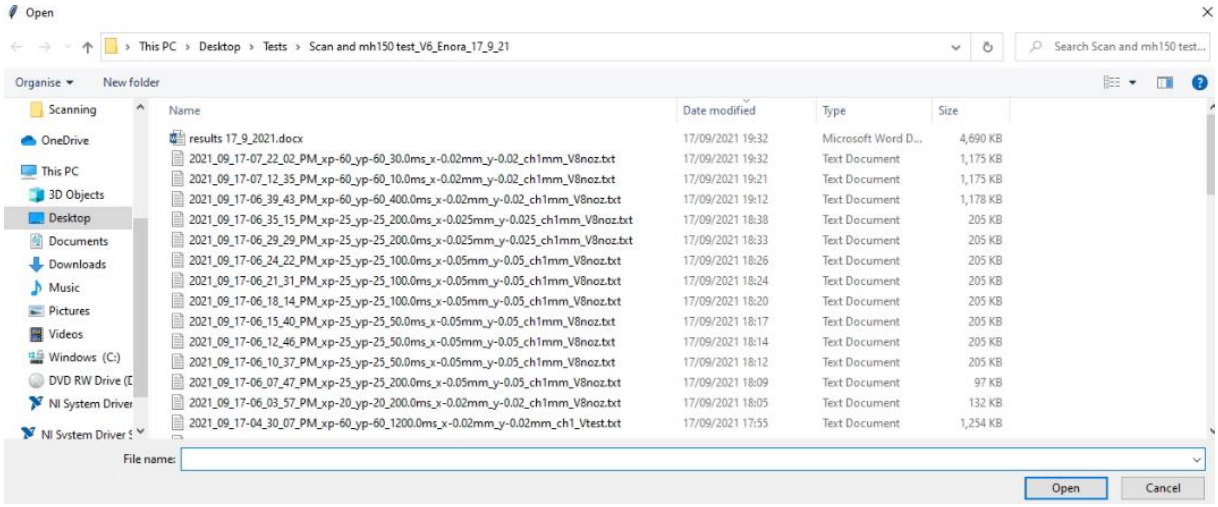

Figure 4.27: Load file button in the interface main window that opens a new window with recorded text files. These files are importable to the software to perform data analysis.

When loading the data, the software will read information from the header, such as the number of pixels, pixel dwell time, and window size. With this information, it is known where each histogram in the file is placed in the corresponded scanning matrix. More specifically, this is done by using the number of pixels that corresponds to the matrix dimension of the location of the associated histograms. The histograms are then associated with the matrix in the order that the scan was performed. So, even though the histograms do not have a marker indicating that a new slow axis position has started, this is known since all the pixels in the previous slow axis have histograms associated. After having the completed matrix, all data analysis tasks are appliable, except the histogram itself, which is already in the text file.

4.2.7 Discussion

Since device control and data analysis are done in Python, which is an open-source language, any researcher may use the software. Python is also compatible with many operating systems, increasing the compatibility of the software. This computing language is drawing attention from many researchers looking into developing new microscopy software (Barabas et al., 2016). A possible drawback from using Python is that the initial libraries of the devices are written in C^{++}, hence there is the need to use specific wrapper functions to read them.

One feature that makes this software remarkable is the successful implementation of real-time feedback and data interaction. The user may obtain information in real-time, change

the lifetime ranges for the splitting and the colour bar in the intensity image, updating and change the image while the scanning is being performed. This possibility is governed by the app.processEvents() function. The problem with this approach is the non-optimal performance of the code because it makes the real-time interaction possible only by flushing all queued events, meaning that when the user clicks on something in the interface main window the information provided by the interaction is instantly sent to the code, not waiting for the scan to finish. Real-time interaction can also be achieved with higher performance by using multithreading, being more used, which allows the window code and the scan code to run simultaneously by different threads (Nguyen et al., 2006). However, multithreading codes require someone with expertise, being difficult to implement, and bad implementation would miss the start times for each part of the software, providing wrong data.

It is also possible to observe in real-time the histogram of the desired pixel while scanning, providing information on the performance of the decay curves. There is also the possibility to choose which fitting algorithm to use, selecting from one (single exponential fitting), two (double exponential fitting), or three decay components (triple exponential fitting), which are an option to find the fluorescence decaying components (Sejung et al., 2015). The fitting algorithms may be changed while scanning. The FLIM was implemented correctly, highlighting the importance of the weights in the data brightness. The distribution of the lifetimes allows the selection of the lifetimes ranges for the FLIM and splits them in equal ways in accordance with the lifetimes present. The extraction of the decay curve from the histogram of the MultiHarp 150N was implemented successfully, fitting only the relevant data.

The use of a 7 fs ultra-broadband laser has multiple advantages, some of them were not properly addressed in this thesis due to time restrictions. In some experiments, the 7 fs laser provided low-intensity pulses and required long pixel dwell times to avoid low signal-to-noise ratios and to assure the fitting of the decay curves by the algorithm. This problem can be detected earlier in a scan by using the real-time feedback of the histogram or intensity map. This allows the user to tune the parameters without having to wait for the end of the scan. The fitting algorithm does not work on very noisy curves. In these situations, there are low-intensity data, which results in bad FLIM imaging.

The option to load recorded data enables the observation and study of the data whenever desired by the user. The data analysis, present in this thesis, was performed by me on the software using this option.

The imaging produced by the software reduces the scanning performance drastically. This problem is due to the opening of a new plot with all the data each time a pixel is measured. To overcome this problem, while a different solution is not found, the option is to disable the imaging, therefore allowing the software to focus only on the device control. Some solutions to reduce the delay from the imaging were studied but the implementation is not yet completed.

The most promising solution is the option of keeping the plot open and updating the data plotted each time a new pixel is measured. Another solution is to implement animations. The computing threading would allow the device control to execute without any delays from the imaging, even though, the imaging information and the device control measurement gap would continuously increase.

5 Conclusion

In sum, the software developed meets all the expectations and requirements, providing successful device control and accurate data analysis. The software saves the data from the scans with all relevant data for future studies, along with the figures from the data analysis. The data saved in text files allows researchers to implement their own data analysis tools. The Graphical User Interface, which allows the interaction with all features of the microscope is intuitive and user-friendly. The real-time interaction features were also implemented successfully, allowing the user to perform data analysis on ongoing scanning data. Functions to control devices without libraries, such as the galvanometric mirrors and shutter, were successfully created, not requiring the user to know the necessary voltage to manage these devices. To convert the galvanometric mirrors units into voltage, a calibration factor was successfully found using a trial-and-error procedure. This factor is not 100% accurate, showing a small decrease in the area scanned. The way in which the software was developed allowed me to continuously add and improve its features.

For future work, the real-time interaction performance can be increased by applying computing threads in the software, allowing the data analysis and device control to run simultaneously. With the implementation of the thread, the scanning patterns could be changed to a continuous setup, improving the scan performance. The deconvolution of the IRF by the decay curve can be implemented to increase the accuracy of the fitting algorithm. One opportunity to deal with noisy data, where fitting becomes more challenging, lies in the application of machine learning techniques, which becomes more interesting once a considerable image database is available. There is also room to improve the existing features of the software, as well as add new ones. Since the objective limits the scanning area when using beam scanning, adding a mosaic architecture would allow the creation of better imaging. This can be done by selecting an area and dividing it into smaller scanning areas. In the end, all the scan data can be staked together in a single image. An option to see the histogram of a pixel by clicking on it would be better than the user having to input the pixel coordinates.

Even though there is room for improvements, the software has shown promising results in the lab tests, meaning that is it ready to further explore the advantages of the 7 fs Ti:Sa laser, namely by opening a new line of research on labelled or label-free imaging of samples. Despite the thesis not showing any results with the excitation of multiple fluorophores by the 7 fs laser, the software is fully ready to be used in this type of research.

References

Baryshnikov, G., Minaev, B., & Ågren, H. (2017). Theory and calculation of the phosphorescence phenomenon. *Chemical Reviews*, *117*(9), 6500–6537. https://doi.org/10.1021/acs.chemrev.7b00060

Becker & Hickl. GmbH (2021, November). Non-descanned FLIM detection in multiphoton microscopes*. Application Note.* https://www.becker-hickl.com/wp-content/uploads/2018/12/ndd-flim-v03.pdf. [Accessed on: 12/09/2021]

Bestvater, F., Spiess, E., Stobrawa, G., Hacker, M., Feurer, T., Porwol, T., Berchner-Pfannschmidt, U., Wotzlaw, C., & Acker, H. (2002). Two-photon fluorescence absorption and emission spectra of dyes relevant for cell imaging. *Journal of Microscopy*, *208*(2), 108–115. https://doi.org/10.1046/j.1365-2818.2002.01074.x

Bower, A. J. (2019). *Label-free multiphoton microscopy for imaging transient metabolic dynamics in living cells and tissue*. (Doctoral dissertation, University of Illinois at Urbana-Champaign). https://www.ideals.illinois.edu/bitstream/handle/2142/105131/BOWER-DISSERTATION-2019.pdf

Boyd, R. W. (2003). The nonlinear optical susceptibility. In Boyd, R. W. (Ed.), *Nonlinear Optics* (2nd ed., pp. 1–65). Elsevier. https://doi.org/10.1016/B978-012121682-5/50002-X

Cha, J. W., Yew, E. Y. S., Kim, D., Subramanian, J., Nedivi, E., & So, P. T. C. (2015). Non-descanned multifocal multiphoton microscopy with a multianode photomultiplier tube. *AIP Advances*, *5*(8), 084802. https://doi.org/10.1063/1.4916040

Cheng., P.-C. (2006). The contrast formation in optical microscopy. In Pawley, J. B. (Ed.) *Handbook of biological confocal microscopy* (3rd ed., pp. 162-206). Springer Science +Business Media, LLC, New York. https://doi.org/10.1007/978-0-387-45524-2

Gavin S. Dawe, Jan-Thorsten Schantz, Mortimer Abramowitz, Michael W. Davidson and Dietmar W. Hutmacher (2006). Light microscopy. In Dokland, T., Hutmacher, D. W., Ng, M. M. L., & Schantz, J.-T. (Ed). *Techniques in microscopy for biomedical applications.* Manuals in Biomedical Research - Vol. 2, Schantz, J.-T. (Ed.). World Scientific. https://doi.org/10.1142/5911

Faria, A. R., Silvestre, O. F., Maibohm, C., Adão, R. M. R., Silva, B. F. B., & Nieder, J. B. (2019). Cubosome nanoparticles for enhanced delivery of mitochondria anticancer drug elesclomol and therapeutic monitoring via sub-cellular NAD(P)H multi-photon fluorescence lifetime imaging. *Nano Research*, *12*(5), 991–998. https://doi.org/10.1007/s12274-018-2231-5

Firefly Conservation & Research (2022, February). Will the real firefly please stand up? Bugs that get confused with fireflies - And how to tell the difference. https://www.firefly.org/bioluminescent-insects.html [Accessed on: 10/01/2022]

Fujimoto, J. G., & Farkas, D. L. (2009). *Biomedical optical imaging*. Oxford University Press.

Gibson, E., Masihzadeh, O., Lei, T., Ammar, D., & Kahook, M. (2011). Multiphoton microscopy for ophthalmic imaging. *Journal of Ophthalmology*, *2011*, 870879. https://doi.org/10.1155/2011/870879

Görlitz, F. (2018). *Development and application of fluorescence lifetime imaging and super-resolution microscopy*. (Doctoral dissertation, Imperial College London). https://doi.org/10.25560/75118

Hell, S. W. (2007). Far-field optical nanoscopy. *Science*, *316*(5828), 1153–1158. https://doi.org/10.1126/science.1137395

Hell, S. W., Bahlmann, K., Schrader, M., Soini, A., Malak, H. M., Gryczynski, I., & Lakowicz, J. R. (1996). Three-photon excitation in fluorescence microscopy. *Journal of Biomedical Optics*, *1*(1), 71–74. https://doi.org/10.1117/12.229062

Huang, B., Bates, M., & Zhuang, X. (2009). Super-resolution fluorescence microscopy. *Annual Review of Biochemistry*, *78*, 993–1016. https://doi.org/10.1146/annurev.biochem.77.061906.092014

Jarvis, S. P., Sweetman, A. M., Kantorovich, L., McGlynn, E., & Moriarty, P. (2015). Pauli's principle in probe microscopy. In Moriarty, P. and Gauthier, S. (Eds.) *Imaging and Manipulation of Adsorbates Using Dynamic Force Microscopy* (pp. 1–24). Proceedings from the AtMol Conference Series, Springer International Publishing, Nottingham. https://doi.org/10.1007/978-3-319-17401-3

Kozma, A., & Kelly, D. L. (1965). Spatial filtering for detection of signals submerged in noise. *Applied Optics*, *4*(4), 387–392. https://doi.org/10.1364/AO.4.000387

Dobrucki J., & Kubitscheck, U. (2017) Fluorescence Microscopy. In Kubitscheck, U. (Ed.). *Fluorescence microscopy: From principles to biological applications*. (2nd ed., pp 85-132). Wiley-VCH, Weinheim. https://onlinelibrary.wiley.com/doi/book/10.1002/9783527687732

Naredi-Rainer, N., Prescher, J., Hartschuh, A., Lamb, D. (2017) Confocal Microscopy. In Kubitscheck, U. (Ed.). *Fluorescence microscopy: From principles to biological applications*. (2nd ed., pp 165-200). Wiley-VCH, Weinheim. https://onlinelibrary.wiley.com/doi/book/10.1002/9783527687732

Nienhaus, U., Nienhaus, K. (2017) Fluorescence Labeling. In Kubitscheck, U. (Ed.). *Fluorescence microscopy: From principles to biological applications*. (2nd ed., pp 133-162). Wiley-VCH, Weinheim. https://onlinelibrary.wiley.com/doi/book/10.1002/9783527687732

Li, F., Park, Y., & Azaña, J. (2007). Complete temporal pulse characterization based on phase reconstruction using optical ultrafast differentiation (PROUD). *Optics Letters*, *32*(22), 3364–3366. https://doi.org/10.1364/OL.32.003364

Liu, Z., Pouli, D., Alonzo, C. A., Varone, A., Karaliota, S., Quinn, K. P., Münger, K., Karalis, K. P., & Georgakoudi, I. (2018). Mapping metabolic changes by noninvasive, multiparametric, high-resolution imaging using endogenous contrast. *Science Advances*, *4*(3), eaap9302. https://doi.org/10.1126/sciadv.aap9302

Maibohm, C., Silva, F., Figueiras, E., Guerreiro, P. T., Brito, M., Romero, R., Crespo, H., & Nieder, J. B. (2019). SyncRGB-FLIM: synchronous fluorescence imaging of red, green and blue dyes enabled by ultra-broadband few-cycle laser excitation and fluorescence lifetime detection. *Biomedical Optics Express*, *10*(4), 1891. https://doi.org/10.1364/boe.10.001891

Microscopy U (2021, November 16). Fundamentals and applications in multiphoton excitation microscopy. Nikon. https://www.microscopyu.com/techniques/multi-photon/multiphoton-microscopy. [Accessed on: 04/06/2021]

Miller, D. R., Jarrett, J. W., Hassan, A. M., & Dunn, A. K. (2017). Deep tissue imaging with multiphoton fluorescence microscopy. *Current Opinion in Biomedical Engineering*, *4*, 32–39. https://doi.org/10.1016/j.cobme.2017.09.004

Miranda, M., Arnold, C. L., Fordell, T., Silva, F., Alonso, B., Weigand, R., L'Huillier, A., & Crespo, H. (2012). *Characterization of broadband few-cycle laser pulses with the d-scan technique. Optics Express*, *20*(17), 18732-18743. https://doi.org/10.1364/OE.20.018732

Momcilovic, M., & Shackelford, D. B. (2018). Imaging cancer metabolism. *Biomolecules & Therapeutics*, *26*(1), 81–92. https://doi.org/10.4062/biomolther.2017.220

Mullen, A. D., Treibitz, T., Roberts, P. L. D., Kelly, E. L. A., Horwitz, R., Smith, J. E., & Jaffe, J. S. (2016). Underwater microscopy for in situ studies of benthic ecosystems. *Nature Communications*, *7*(1), 12093. https://doi.org/10.1038/ncomms12093

Murthy, K. V. R., & Virk, H. S. (2014). Luminescence phenomena: An introduction. *Defect and Diffusion Forum*, *347*, 1–34. https://doi.org/10.4028/www.scientific.net/DDF.347.1

Phizicky, E. M., & Fields, S. (1995). Protein-protein interactions: Methods for detection and analysis. *Microbiological reviews*, *59*(1), 94-123. https://doi.org/10.1128/mr.59.1.94-123.1995

Physik Instrumente Group. (2022, February). ND72Z2LAQ PIFOC Objective Scanning System 2000 µm *[Brochure]*. https://www.physikinstrumente.com/en/products/nanopositioning-piezo-flexure-stages/pifoc-objective-pinano-sample-scanners-for-microscopy/nd72z2laq-pifoc-objective-scanning-system-2000m-1100210/#specification [Accessed on: 26/01/2022]

Scientific Volume Imaging. (2021, November). *Fluorescence*. https://svi.nl/Fluorescence [Accessed on: 12/11/2021]

Semwogerere, D., & Weeks, E. R. (2005). Confocal microscopy. *Encyclopedia of Biomaterials and Biomedical Engineering*, *23*, 1–10. https://doi.org/10.1081/E-EBBE-120024153

Sevick-Muraca, E. M., Houston, J. P., & Gurfinkel, M. (2002). Fluorescence-enhanced, near infrared diagnostic imaging with contrast agents. *Current Opinion in Chemical Biology*, *6*(5), 642–650. https://doi.org/10.1016/S1367-5931(02)00356-3

Thorlabs, GmbH. (2022d, February). BBD202: User manual. Retrieved from https://www.thorlabs.com/thorproduct.cfm?partnumber=BBD202 [Accessed on: 10/01/2022]

Thorlabs, GmbH (2022b, February). GVSM002-EC/M: User manual. Retrieved from https://www.thorlabs.com/thorproduct.cfm?partnumber=GVSM002-EC/M [Accessed on: 10/01/2022]

Thorlabs, GmbH (2022c, February). MLS203-1: User manual. Retrieved from https://www.thorlabs.de/thorproduct.cfm?partnumber=MLS203-1 [Accessed on: 10/01/2022]

Thorlabs, GmbH (2022a, February). SHB025T - Ø1/4" Low-Reflectance Diaphragm Optical Beam Shutter: User manual. Retrieved from

https://www.thorlabs.com/thorproduct.cfm?partnumber=SHB025T#ad-image-0 [Accessed on: 10/01/2022]

PicoQuant (2021, November). Multiharp 150 N. https://www.picoquant.com/products/category/tcspc-and-time-tagging-modules/multiharp-150-high-throughput-multichannel-event-timer-tcspc-unit#description [Accessed on: 13/11/2021]

Becker, W. (2021). *The bh TCSPC handbook* (9th ed.). https://www.becker-hickl.com/literature/documents/flim/the-bh-tcspc-handbook/ [Accessed on: 19/04/2021]

Walmsley, I. A., & Dorrer, C. (2009). Characterization of ultrashort electromagnetic pulses. *Advances in Optics and Photonics.*, *1*(2), 308–437. https://doi.org/10.1364/AOP.1.000308

Worbs, T. (2006). *Chemokine receptor CCR7 contributes to intranodal T cell motility and functional organization of the intestinal immune system*. (Doctoral dissertation, Hannover, Med. Hochsch., Diss., 2007). https://web.archive.org/web/20211202192849id_/https://mhh-publikationsserver.gbv.de/servlets/MCRFileNodeServlet/mhh_derivate_00000901/diss-worbs_a.pdf

Yoshida, S., Tanaka, S., Hirata, M., Mouri, R., Kaneko, I., Oka, S., Yoshihara, M., & Chayama, K. (2007). Optical biopsy of GI lesions by reflectance-type laser-scanning confocal microscopy. *Gastrointestinal Endoscopy*, *66*(1), 144–149. https://doi.org/https://doi.org/10.1016/j.gie.2006.10.054

Pilger C, Pospíšil J, Müller M, Ruoff M, Schütte M, Spiecker H, Huser T. Super-resolution fluorescence microscopy by line-scanning with an unmodified two-photon microscope. Philos Trans A Math Phys Eng Sci. 2021 Jun 14;379(2199):20200300. doi: 10.1098/rsta.2020.0300. Epub 2021 Apr 26. PMID: 33896201; PMCID: PMC8072199.

Kumazaki S, Hasegawa M, Ghoneim M, Shimizu Y, Okamoto K, Nishiyama M, Oh-Oka H, Terazima M. A line-scanning semi-confocal multi-photon fluorescence microscope with a simultaneous broadband spectral acquisition and its application to the study of the thylakoid membrane of a cyanobacterium Anabaena PCC7120. J Microsc. 2007 Nov;228(Pt 2):240-54. doi: 10.1111/j.1365-2818.2007.01835.x. PMID: 17970923.

Barabas FM, Masullo LA, Stefani FD. Note: Tormenta: An open source Python-powered control software for camera based optical microscopy. Rev Sci Instrum. 2016 Dec;87(12):126103. doi: 10.1063/1.4972392. PMID: 28040938.

Nguyen QT, Tsai PS, Kleinfeld D. MPScope: a versatile software suite for multiphoton microscopy. J Neurosci Methods. 2006 Sep 30;156(1-2):351-9. doi: 10.1016/j.jneumeth.2006.03.001. Epub 2006 Apr 18. PMID: 16621010.

Sejung Yang, Joohyun Lee, Youmin Lee, Minyung Lee, Byung-Uk Lee, "Estimation of multiexponential fluorescence decay parameters using compressive sensing," J. Biomed. Opt. 20(9) 096003 (3 September 2015) https://doi.org/10.1117/1.JBO.20.9.096003

Appendix A - Code for Controllable Components and Data Analysis

A.1 Code for Controllable Components

 # Connecting all devices

```
def Connect_Stage(self):
    self.connectmh150()
    ss.ConnectDevice()
    ss.velocity_Change(1, 1000, 100)
     ss.velocity_Change(2, 1000, 100)
if self.ActivateZstage.isChecked():
    try:
        self.zstage = 1
        self.Connect_Z_Stage()
        sleep(1)
        self.z_Readout()
        print('Z-stage connected')
     except:
        print('z-stage is not turned on')
   self.LCDReadout()
```

 # Control functions for the shutter

```
class shutter():
    def ConnectionToNICard(self):
        task = nidaqmx.Task()
        task.do_channels.add_do_chan("Dev1/port0/line7")
        task.start()
    def Reset(self):
        task = nidaqmx.Task()
        task.stop()
        task.close()
        task.do_channels.add_do_chan("Dev1/port0/line7")
        task.start()
    def Open_Shutter(self):
        value = True # gives a voltage, if false, no voltage output
        task.write(value)
    def Close_Shutter(self):
        value = False # gives a voltage, if false, no voltage output
        task.write(value)
```

 # Control functions for the galvanometric mirrors

```
#Park laser function for the galvo mirrors
def Parklaser(self):
    y_task = nidaqmx.Task('y_galvo')
    y_task.ao_channels.add_ao_voltage_chan("PXI1Slot2_2/ao1", '', -10,\
10)
    y_task.start()
    x_task = nidaqmx.Task('x_galvo')
    x_task.ao_channels.add_ao_voltage_chan("PXI1Slot2_2/ao0", '', -10,\
10)
    x_task.start()
    xsp = float(self.OffsetXVoltage.value())
    ysp = float(self.OffsetYVoltage.value())
```

```
        x_task.write(xsp)
        y_task.write(ysp)
        x_task.stop()
        y_task.stop()
        x_task.close()
        y_task.close()
#Addfunction that records histogram on single point
        print('laser parked')
#Complete scan function for the Galvo mirrors
    def Galvo_Scan(self):
        start=time.time()
        ynt = 1
        xnt = 1
        cspx = ss.Readout(1)
        cspy = ss.Readout(2)
        overlap = self.Overlap.value() / 100
        yws = float(self.Y_Window_Size.value()/1000)
        xws = float(self.X_Window_Size.value()/1000)
        oy = yws * overlap
        ox = xws * overlap
        oyws = yws - oy
        oxws = xws - ox
        self.scanstop=False
        self.autocolorbarbol = True
        self.galvocalibration=10 # calibration factor for the galvo
        z = self.znumbersteps()
        for f in range(z):
            self.Parklaser()
            from datetime import datetime
            date = datetime.now().strftime("%Y_%m_%d-%I_%M_%S_%p")
            xnp = int(self.X_Number_of_Pixels.value())
            ynp = int(self.Y_Number_of_Pixels.value())
            yws = float(self.Y_Window_Size.value())/1000
            xws = float(self.X_Window_Size.value())/1000 # conversion from
um to mm
            xsp = float(self.OffsetXVoltage.value())
            ysp = float(self.OffsetYVoltage.value())
            vssx = float(xws * self.galvocalibration / xnp)  # set voltage
on window but once we know the correct voltage for this lens system we
can use a conversion formula
            vssy = float(yws * self.galvocalibration / ynp)
            pdt = float(self.Pixel_Dwell_Time.value())  # Record histogram
already changes the value to miliseconds
            hwsx = xws / 2 #half window size x
            hwsy = yws / 2 #half window size y
                ###Galvos task start###
            y_task = nidaqmx.Task('y_galvo')
            y_task.ao_channels.add_ao_voltage_chan("PXI1Slot2_2/ao1", '', \
-10, 10)
            y_task.start()
            x_task = nidaqmx.Task('x_galvo')
            x_task.ao_channels.add_ao_voltage_chan("PXI1Slot2_2/ao0", '', \
-10, 10)
            x_task.start()
            nchannels = self.NChannels.value()
            if nchannels == 2:
               channel1 = True
               channel2 = True
               channel3 = False
               channel4 = False
             elif nchannels == 3:
```

```
                channel1 = True
                channel2 = True
                channel3 = True
                channel4 = False
            elif nchannels == 4:
                channel1 = True
                channel2 = True
                channel3 = True
                channel4 = True
            else:
                channel1 = True
                channel2 = False
                channel3 = False
                channel4 = False
                channels = [channel1, channel2, channel3, channel4]
                outputfile = [0, 0, 0, 0]
            global histogramlist, intensitymatrix
            intensitymatrix = np.zeros((xnp, ynp))
            intensitymatrix[:] = np.nan
            histogramlist = np.zeros((xnp, ynp), dtype=object)
            histogramlist[:] = np.nan
            taumedio = np.zeros((xnp, ynp))
            taumedio[:] = np.nan
            weight1 = np.zeros((xnp, ynp))
            weight1[:] = np.nan
            weight2 = np.zeros((xnp, ynp))
            weight2[:] = np.nan
            weight3 = np.zeros((xnp, ynp))
            weight3[:] = np.nan
            tau1 = np.zeros((xnp, ynp))
            tau1[:] = np.nan
            tau2 = np.zeros((xnp, ynp))
            tau2[:] = np.nan
            tau3 = np.zeros((xnp, ynp))
            tau3[:] = np.nan
            weightR = np.zeros((xnp, ynp))
            weightR[:] = np.nan
            weightG = np.zeros((xnp, ynp))
            weightG[:] = np.nan
            weightB = np.zeros((xnp, ynp))
            weightB[:] = np.nan
            trash = np.zeros((xnp, ynp))
            trash[:] = np.nan
            listoflifetimes = []
            self.changelifebolean=False
            if self.ActivateZstack.isChecked():
               zpositionheader=(-1*self.cspz)+1+f*self.zss
            else:
               zpositionheader=0
            for i in range(len(channels)):
                if channels[i] == True:
#title of file
                filepath = rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests  \
\Scan and mh150... test_V6_Enora_Tests/{date}__Z-{f}_xp-{xnp}_yp  \
{ynp}_{pdt}\ ms_x-{xws}mm_y-{yws}_ch{i +... 1}_GSV9.txt"
# Header File creation
                outputfile[i] = open(filepath, "w+")
                outputfile[i].write("Channel                : %s\n" % str(i\
+ 1))
                outputfile[i].write("Binning                : %d\n" %\
self.binning)
```

```
                outputfile[i].write("Offset                : %d\n" % \
self.offset)
                outputfile[i].write("collectionTime(ms)    : %s\n" % pdt)
                outputfile[i].write("SyncDivider           : %d\n" % \
self.syncDivider)
                outputfile[i].write("SyncEdgeTrg           : %d\n" % \
self.syncEdgeTrg)
                outputfile[i].write("InputEdgeTrg          : %d\n" % \
self.inputEdgeTrg)
                outputfile[i].write("BaseResolution(ps)    : %d\n" % \
self.resolution.value)
                outputfile[i].write("blockLength           : %d\n" % \
self.blocklength)
                outputfile[i].write("syncRate              : %d\n" % \
self.syncRate.value)
                outputfile[i].write("sizeX                 : %d\n" % xnp)
                outputfile[i].write("sizeY                 : %d\n" % ynp)
                outputfile[i].write("startX(mm)            : %d\n" % 0)
                outputfile[i].write("endX(mm)              : %s\n" % \
str(xws))
                outputfile[i].write("startY(mm)            : %d\n" % 0)
                outputfile[i].write("endY(mm)              : %s\n" % \
str(yws))
                outputfile[i].write("TAC_gain              : %d\n" % 1)
                outputfile[i].write("z(mm)                 : %s\n" % \
str(zpositionheader))
                outputfile[i].write('counts:\n')
                flyback = 0.005  # seconds, needed for y to have some time
to return to start position
                i = 0
                j = 0
                .
                initialx=xsp-hwsx*self.galvocalibration
                initialy=ysp+hwsy*self.galvocalibration
                x_task.write(initialx)
                y_task.write(initialy)
                yv = initialy
                while i != xnp and j != ynp:
                      if i == 0:
                         x_task.write(initialx)
                         xv = initialx
                         xv = xv + vssx
                         x_task.write(xv)
                      for c in range(len(channels)):
                         if channels[c] == True:
                            intensitysum, histogram = \
self.recordhist(int(pdt), outputfile, c)
                            intensitymatrix[j, i] = intensitysum
                            histogramlist[j][i] = histogram
                            taumedio[j, i], weight1[j, i], weight2[j, i],\
weight3[j, i], tau1[j, i], tau2[j, i], tau3[j, i]= \
self.fittingrt(histogram)
                            weightR[j, i], weightG[j, i], weightB[j, i], \
trash[j, i] = self.lifetimesplitting(tau1, tau2, tau3, weight1, \
weight2, weight3, j, i ,intensitysum)
                            threshold = \
self.intensitythresholdvalue.value()
                           self.drawintensitymatrix(intensitymatrix, \
xnp,  ynp)
                           self.lifetimesplittingimaging(weightR, \
weightG, weightB, trash)
```

```
                        self.changerangeflim.clicked.connect(lambda: \
self.changeLifetimes())
                        if self.changelifebolean == True:
                            for k in range(j):
                                for l in range(xnp):
                                    weightR[k, l], weightG[k, l], \
weightB[k, l], trash[k, l] = self.lifetimesplitting(tau1, tau2, tau3, \
weight1, weight2, weight3, k, l, intensitysum)
                        self.changelifebolean = False
                        self.autocolorbar.clicked.connect(lambda: \
self.autocolorbarfunc(intensitymatrix, xnp, ynp))
                        self.chosencolorbar.clicked.connect(lambda: \
self.chosencolorbarfunc(intensitymatrix, xnp, ynp))
                        if tau1[j, i] != 0 and intensitymatrix[j, i] \
>  threshold and tau1[j, i] < 12.5 and tau1[j, i]>=0:
                            listoflifetimes.append(tau1[j, i])
                        if tau2[j, i] != 0 and intensitymatrix[j, i] \
> threshold and tau2[j, i] < 12.5 and tau2[j, i]>=0:
                            listoflifetimes.append(tau2[j, i])
                        if tau3[j, i] != 0 and intensitymatrix[j, i] \
> threshold and tau3[j, i] < 12.5 and tau3[j, i]>=0:
                            listoflifetimes.append(tau3[j, i])
                i = i + 1
                if i == xnp:
                    yv = yv - vssy
                    y_task.write(yv)
                    sleep(flyback)
                    xv = xsp
                    i = 0
                    j = j + 1
                app.processEvents()
                self.stop.clicked.connect(self.stopScan)
                if self.scanstop == True:
                    x_task.stop()
                    y_task.stop()
                    x_task.close()
                    y_task.close()
                    self.Parklaser()
                    if self.ActivateZstack.isChecked():
                        self.centerZ()
                    print('scan stoped sucessfuly')
                    return
                    x_task.stop()
                    y_task.stop()
                    x_task.close()
                    y_task.close()
            for l in range(len(channels)):
                if channels[l] == True:
                    outputfile[l].close()
            if self.ActivateZstack.isChecked()==False:
                self.lifetimesplittingimaging(weightR, weightG, \
weightB, trash)
                print('Scan Done')
#stage.close()
                try:
                    self.histogramlifetimes(listoflifetimes)
                except:
                    print('no lifetimes')
            self.autocolorbar.clicked.connect(lambda: \
self.autocolorbarfunc(intensitymatrix, xnp, ynp))
```

```
            self.chosencolorbar.clicked.connect(lambda: \
self.chosencolorbarfunc(intensitymatrix, xnp, ynp))
            self.flim.clicked.connect(lambda: self.FLIM( \
histogramlist,  intensitymatrix))
            self.pushButtonhist.clicked.connect(lambda: \
self.drawhistograms(histogramlist))
            if self.ActivateZstack.isChecked():
                self.z_step(self.zss)
            # self.zmovetoscanstart()
        if self.ActivateZstack.isChecked():
            self.centerZ()
        end=time.time()
        print(end-start)
        print('Scan Complete')
```

 # Control functions for xy microscanner

```
#functions to control the xy microscanner
def MoveToX(self):
    mtx = self.Move_To_X.value()
    x_position = mtx
    ss.moveTo(1, x_position)
    self.LCDReadout()
    print('Moved to Position!')
def MoveToY(self):
    mty = self.Move_To_Y.value()
    y_position = mty
    ss.moveTo(2, y_position)
    self.LCDReadout()
    print('Moved to Position!')
def Homing(self):
    ss.moveTo(1,55)
    ss.moveTo(2,37.5)
    self.LCDReadout()
    print('Device Homed')
def XHoming(self):
    ss.moveTo(1,55)
    self.LCDReadout()
    print('X axis Homed')
def YHoming(self):
    ss.moveTo(2,37.5)
    self.LCDReadout()
    print('Y axis Homed')
def Big_Jog(self, channel, size, mf):
    size = size * mf
    if self.D_Limits(channel, size) == True:
       ss.Jog(channel, size)
       self.LCDReadout()
def YPStep(self):
    js = self.Jog_Size.value()/1000
    ss.Jog(2, js)
    self.LCDReadout()
def YPPStep(self):
    js = self.Jog_Size.value()/1000
    jxf = self.Jog_X_Factor.value()
    self.Big_Jog(2, js, jxf)
    self.LCDReadout()
def YNStep(self):
    js = self.Jog_Size.value()/1000
    ss.Jog(2, -js)
    self.LCDReadout()
```

```
def YNNStep(self):
    js = self.Jog_Size.value()/1000
    jxf = self.Jog_X_Factor.value()
    self.Big_Jog(2, -js, jxf)
    self.LCDReadout()
def XPStep(self):
    js = self.Jog_Size.value()/1000
    ss.Jog(1, js)
    self.LCDReadout()
def XPPStep(self):
    js = self.Jog_Size.value()/1000
    jxf = self.Jog_X_Factor.value()
    self.Big_Jog(1, js, jxf)
    self.LCDReadout()
def XNStep(self):
    js = self.Jog_Size.value()/1000
    ss.Jog(1, -js)
    self.LCDReadout()
def XNNStep(self):
    js = self.Jog_Size.value()/1000
    jxf = self.Jog_X_Factor.value()
    self.Big_Jog(1, -js, jxf)
    self.LCDReadout()
#Scan function using the xy microscanner
    start=time.time()
    cspx = ss.Readout(1)
    cspy = ss.Readout(2)
    yws = float(self.Y_Window_Size.value())
    xws = float(self.X_Window_Size.value())
# current scan frame is top corner
# stage at end of scan moves to the left and jst enough to do another
frame with some overlap, repeat for all tiles
# stage moves entire window size - overlap percentage
    a = 0
    b = 0
    self.scanstop = False
    self.autocolorbarbol = True
    z=self.znumbersteps()
    for f in range(z):
        from datetime import datetime
        date = datetime.now().strftime("%Y_%m_%d-%I_%M_%S_%p")
        l = []
        xvel = self.Xvel.value()
        yvel = self.Yvel.value()
        xacell = self.Xaccel.value()
        yacell = self.Yaccel.value()
        xnp = int(self.X_Number_of_Pixels.value())
        ynp = int(self.Y_Number_of_Pixels.value())
        yws = self.Y_Window_Size.value()/1000
        xws = self.X_Window_Size.value()/1000
        hxws = xws / 2
        hyws = yws / 2
        cspx = ss.Readout(1)
        cspy = ss.Readout(2)
        l.append(cspx)
        l.append(cspy)
        scsx = cspx + hxws
        scsy = cspy + hyws
        pdt = self.Pixel_Dwell_Time.value()  # record hist later
converts this value from seconds to milliseconds
# position of the stage for beginning of scan
```

```
            ssx = - (xws / xnp)
            ssy = - (yws / ynp)
            ss.velocity_Change(1, xacell, xvel)  # Stage can possibly go
    faster but would require experimental optimization
            ss.velocity_Change(2, yacell, yvel)
    # max v 400 mm/s
    # max a 2480 mm/s2
    # which channels are on.
            nchannels = self.NChannels.value()
            if nchannels == 2:
               channel1 = True
               channel2 = True
               channel3 = False
               channel4 = False
            elif nchannels == 3:
               channel1 = True
               channel2 = True
               channel3 = True
               channel4 = False
            elif nchannels == 4:
               channel1 = True
               channel2 = True
               channel3 = True
               channel4 = True
            else:
               channel1 = True
               channel2 = False
               channel3 = False
               channel4 = False
               channels = [channel1, channel2, channel3, channel4]
               outputfile = [0, 0, 0, 0]
    # Z-axis starts at current position and scans down to prevent crashing
    into objective
            global histogramlist, intensitymatrix
            intensitymatrix = np.zeros((xnp, ynp))
            intensitymatrix[:] = np.nan
            histogramlist = np.zeros((xnp, ynp), dtype=object)
            histogramlist[:] = np.nan
            taumedio = np.zeros((xnp, ynp))
            taumedio[:] = np.nan
            weight1 = np.zeros((xnp, ynp))
            weight1[:] = np.nan
            weight2 = np.zeros((xnp, ynp))
            weight2[:] = np.nan
            weight3 = np.zeros((xnp, ynp))
            weight3[:] = np.nan
            tau1 = np.zeros((xnp, ynp))
            tau1[:] = np.nan
            tau2 = np.zeros((xnp, ynp))
            tau2[:] = np.nan
            tau3 = np.zeros((xnp, ynp))
            tau3[:] = np.nan
            weightR = np.zeros((xnp, ynp))
            weightR[:] = np.nan
            weightG = np.zeros((xnp, ynp))
            weightG[:] = np.nan
            weightB = np.zeros((xnp, ynp))
            weightB[:] = np.nan
            trash = np.zeros((xnp, ynp))
            trash[:] = np.nan
            listoflifetimes = []
```

```
            self.changelifebolean = False
            if self.ActivateZstack.isChecked():
               zpositionheader=(-1*self.cspz)+1+f*self.zss
            else:
               zpositionheader=0
            for i in range(len(channels)):
                if channels[i] == True:  # Header File creation
                            filepath =
                rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests\Scan and mh150 \
                test_V6_Enora_Tests/{date}__z{f}_position_xp-{xnp}_yp-{ynp}_ \
                {pdt}ms_x-{xws} mm_y-{yws}_ch{i + 1}_SSV9.txt"
        #stage = open(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests\Scan and mh150
        test_V6_Enora_Tests/stagepos.txt", "w+")
            outputfile[i] = open(filepath, "w+")
            outputfile[i].write("Channel              : %s\n" % str(i + 1))
            outputfile[i].write("Binning              : %d\n" % self.binning)
            outputfile[i].write("Offset               : %d\n" % self.offset)
            outputfile[i].write("collectionTime(ms)   : %s\n" % pdt)
            outputfile[i].write("SyncDivider          : %d\n" % \
            self.syncDivider)
            outputfile[i].write("SyncEdgeTrg          : %d\n" % \
            self.syncEdgeTrg)
            outputfile[i].write("InputEdgeTrg         : %d\n" % \
            self.inputEdgeTrg)
            outputfile[i].write("BaseResolution(ps)   : %d\n" % \
            self.resolution.value)
            outputfile[i].write("blockLength          : %d\n" % \
            self.blocklength)
            outputfile[i].write("syncRate             : %d\n" % \
            self.syncRate.value)
            outputfile[i].write("sizeX                : %d\n" % xnp)
            outputfile[i].write("sizeY                : %d\n" % ynp)
            outputfile[i].write("startX(mm)           : %d\n" % 0)
            outputfile[i].write("endX(mm)             : %s\n" % str(xws))
            outputfile[i].write("startY(mm)           : %d\n" % 0)
            outputfile[i].write("endY(mm)             : %s\n" % str(yws))
            outputfile[i].write("TAC_gain             : %d\n" % 1)
            outputfile[i].write("z(mm)                : %s\n" % \
            str(zpositionheader))
            outputfile[i].write('counts:\n')
            # Start of scan motion
                i = 0
                j = 0
                yposition=scsy
                ss.moveTo(2, scsy)
                print('Scanning...')
                while j != ynp and i != xnp:
                      if i == 0:
                         ss.moveTo(1, scsx)
                         xpostition = scsx
            #stage.write(str(ss.Readout(1)) + "; " + str(ss.Readout(2)) + "
            ")
                         for c in range(len(channels)):  # records counts
            for pixel dwell time
                               if channels[c] == True:
                                  intensitysum, histogram = \
            self.recordhist(int(pdt), outputfile, c)
                                  intensitymatrix[j, i] = intensitysum
                                  histogramlist[j][i] = histogram
```

```
                    taumedio[j, i], weight1[j, i], \
weight2[j, i], weight3[j, i], tau1[j, i], tau2[j, i], \
tau3[j,i]  = self.fittingrt(histogram)
                    weightR[j, i], weightG[j, i], \
weightB[j, i], trash[j, i] = self.lifetimesplitting(tau1,\
tau2, tau3, weight1, weight2, weight3, j, i, intensitysum)
                    self.drawintensitymatrix(intensitymatrix,\
xnp, ynp)
                    self.lifetimesplittingimaging(weightR, \
weightG, weightB, trash)

self.chosencolorbar.clicked.connect(lambda: \
self.chosencolorbarfunc(intensitymatrix, xnp, ynp))

self.changerangeflim.clicked.connect(lambda: \
self.changeLifetimes())
                    if self.changelifebolean == True:
                        for k in range(j):
                            for l in range(xnp):
                                weightR[k, l], weightG[k, l], \
weightB[k, l], trash[k, l] = self.lifetimesplitting( tau1, \
tau2, tau3, weight1, weight2, weight3, k, l, intensitysum)
                                self.changelifebolean = False
# self.lifetimesplittingimaging(weightR, weightG, weightB,
trash)
                    threshold = \
self.intensitythresholdvalue.value()
                    if tau1[j, i] != 0 and \
intensitymatrix[j, i] > threshold and tau1[j, i] < 12.5 \
and  tau1[j, i]>=0:
                        listoflifetimes.append(tau1[j, i])
                    if tau2[j, i] != 0 and \
intensitymatrix[j, i] > threshold and tau2[j, i] < 12.5 \
and  tau2[j, i]>=0:
                        listoflifetimes.append(tau2[j, i])
                    if tau3[j, i] != 0 and \
intensitymatrix[j, i] > threshold and tau3[j, i] < 12.5 \
and tau3[j, i]>=0:
                        listoflifetimes.append(tau3[j, i])
                xpostition=xpostition+ssx
                ss.moveTo(1, xpostition)
                i = i + 1
                if i == xnp:
                    yposition=yposition+ssy
                    ss.moveTo(2, yposition)
                    i = 0
                    j = j + 1
                app.processEvents()
                self.stop.clicked.connect(self.stopScan)
                if self.scanstop == True:
                    print('scan stoped sucessfuly')
                    ss.moveTo(1, cspx)
                    ss.moveTo(2, cspy)
                    if self.ActivateZstack.isChecked():
                        self.centerZ()
                    return
            ss.moveTo(1, ss.Readout(1)+xws/2)
            ss.moveTo(2, ss.Readout(2)+yws/2)
            self.LCDReadout()
            for l in range(len(channels)):
                if channels[l] == True:
```

```
                            outputfile[l].close()
                        if self.ActivateZstack.isChecked()==False:
                            self.lifetimesplittingimaging(weightR, \
        weightG, weightB, trash)
                        print('Scan Done')
                        try:
                            self.histogramlifetimes(listoflifetimes)
                        except:
                            print('no lifetimes')
                    self.flim.clicked.connect(lambda: \
        self.FLIM(histogramlist, intensitymatrix))
                    self.pushButtonhist.clicked.connect(lambda: \
        self.drawhistograms(histogramlist))
                    self.autocolorbar.clicked.connect(lambda: \
        self.autocolorbarfunc(intensitymatrix, xnp, ynp))
                    self.chosencolorbar.clicked.connect(lambda: \
        self.chosencolorbarfunc(intensitymatrix, xnp, ynp))
                    if self.ActivateZstack.isChecked():
                        self.z_step(self.zss)
                        #self.zmovetoscanstart()
                if self.ActivateZstack.isChecked():
                    self.centerZ()
                end=time.time()
                print(end-start)
                print('Scan Complete')
```

 # Control functions for z-piezo

```
# Zstagemovement
def Connect_Z_Stage(self):
    self.pi_device = GCSDevice('E-861')  # Load PI Python Libraries
    self.pi_device.ConnectUSB('0120043695')  # Connect to the
controller via USB
    self.pi_device.SVO('1', 1)  # Turn on servo control of axis "1"
    self.z_Readout()
def z_step(self, size):
    self.pi_device.SVO('1', 1)  # Turn on servo control of axis "1"
    position = self.z_Readout()  # Query current position of axis "1"
    new_position = float(position - size)
    sleep(0.2)
    if new_position <= -1 or new_position >= 1:
       print('Outside of stage limit')
    else:
       self.pi_device.MOV('1', new_position)  # Command axis "1" to
position x units in mm
       position = self.z_Readout()
    print('z Jog done')
def z_home(self):
    self.pi_device.MOV('1', 0)
    sleep(2)
    self.z_Readout()
def Big_Jog_Z(self, size, mf):
    size = size * mf
    self.z_step(size)
    sleep(0.5)
    sleep(0.5)
    self.z_Readout()
def z_Readout(self):
    position = float(self.pi_device.qPOS('1')['1'])  # Query current
position of axis "1"
    sleep(0.5)
    self.Z_Position.display((-position*1000)+1000)
```

```
        return position
    def centerZ(self):
        position = 0
        self.pi_device.MOV('1', position)  # Command axis "1" to position x
    units in mm
        sleep(0.5)
        self.z_Readout()
    def Z_Move_To(self):
        position=self.Move_To_Z.value()
     if position <= 0:
        print('Outside of stage limit')
        self.pi_device.MOV('1', -1)
        self.z_Readout()
    elif position >= 2000:
         print('Outside of stage limit')
         self.pi_device.MOV('1', 1)
         self.z_Readout()
    else:
         position= -1*(position-1000)/1000
         self.pi_device.MOV('1', position)  # Command axis "1" to position
    x units in mm
         sleep(0.5)
         self.z_Readout()
    def ZPStep(self):
         js = self.Jog_Size.value()/1000
         self.z_step(js)
         sleep(0.5)
         self.z_Readout()
    def ZPPStep(self):
         js = self.Jog_Size.value()/1000
         jxf = self.Jog_X_Factor.value()
         self.Big_Jog_Z(js, jxf)
         sleep(0.5)
         self.z_Readout()
    def ZNStep(self):
         js = self.Jog_Size.value()/1000
         self.z_step(-js)
         sleep(0.5)
         self.z_Readout()
    def ZNNStep(self):
        js = self.Jog_Size.value()/1000
        jxf = self.Jog_X_Factor.value()
        self.Big_Jog_Z(-js, jxf)
        sleep(0.5)
        self.z_Readout()
```

 # Control functions for Multiharp 150 N

```
# Multiharp code to create histograms
# From mhdefin.h
def mh150variables(self):
    self.LIB_VERSION = "2.0"
    self.MAXDEVNUM = 8
    self.MODE_HIST = 0
    self.MAXLENCODE = 6
    self.MAXINPCHAN = 16
    self.MAXHISTLEN = 65536
    self.FLAG_OVERFLOW = 0x0001
# Measurement parameters, these are hardcoded since this is just a demo
    self.binning = self.Binning.value()  # you can change this
    self.offset = 0
```

```
    self.tacq = 1000  # Measurement time in millisec, you can change
this
    self.syncDivider = int(self.SyncDivider.value())# you can change
this
    self.syncEdgeTrg = int(self.SyncEdgeTrg.value())# you can change
this (in mV)
    self.syncChannelOffset = int(self.SyncSignalOffset.value())# you
can change this (in ps, like a cable delay)
    self.inputEdgeTrg = int(self.InputEdgeTrg.value())  # you can
change this (in mV)
     self.inputChannelOffset =int(self.InputChannelOffset.value())
# you can change this (in ps, like a cable delay)
     self.cmd = 0
# Variables to store information read from DLLs
     self.counts = [(ct.c_uint * self.MAXHISTLEN)() for i in
range(0,... self.MAXINPCHAN)]
     self.dev = []
     self.libVersion = ct.create_string_buffer(b"", 8)
     self.hwSerial = ct.create_string_buffer(b"", 8)
     self.hwPartno = ct.create_string_buffer(b"", 8)
     self.hwVersion = ct.create_string_buffer(b"", 8)
     self.hwModel = ct.create_string_buffer(b"", 24)
     self.errorString = ct.create_string_buffer(b"", 40)
     self.numChannels = ct.c_int()
     self.histLen = ct.c_int()
     self.resolution = ct.c_double()
     self.syncRate = ct.c_int()
     self.countRate = ct.c_int()
     self.flags = ct.c_int()
     self.warnings = ct.c_int()
     self.warningstext = ct.create_string_buffer(b"", 16384)
     if os.name == "nt":
        self.mhlib = ct.WinDLL("mhlib64.dll")
     else:
        self.mhlib = ct.CDLL("libmh150.so")
def closeDevices(self):  # Closes the multiharp
     for i in range(0, self.MAXDEVNUM):
         self.mhlib.MH_CloseDevice(ct.c_int(i))
         exit(0)
def tryfunc(self, retcode, funcName):  # Starts up and Connects to the
Multiharp
     if retcode < 0:
        self.mhlib.MH_GetErrorString(self.errorString,
ct.c_int(retcode))
     print("MH_%s error %d (%s). Aborted." % (funcName, retcode, \
self.errorString.value.decode("utf-8")))
        self.closeDevices()
def connectmh150(self):
    print("\nSearching for MultiHarp devices...")
    print("Devidx     Status")
    for i in range(0, self.MAXDEVNUM):
        retcode = self.mhlib.MH_OpenDevice(ct.c_int(i), self.hwSerial)
        if retcode == 0:
           print("  %1d        S/N %s" % (i,
self.hwSerial.value.decode("utf-8")))
           self.dev.append(i)
        else:
            if retcode == -1:  # MH_ERROR_DEVICE_OPEN_FAIL
               print("  %1d        no device" % i)
            else:
```

```
                self.mhlib.MH_GetErrorString(self.errorString, \
ct.c_int(retcode))
                print("  %1d        %s" % (i, \
self.errorString.value.decode("utf8")))
        if len(self.dev) < 1:
            print("No device available.")
            exit(0)
        print("Using device #%1d" % self.dev[0])
        print("\nInitializing the device...")
# Histo mode with internal clock
        self.tryfunc(self.mhlib.MH_Initialize(ct.c_int(self.dev[0]), \
ct.c_int(self.MODE_HIST), ct.c_int(0)), "Initialize")
# Only for information
        self.tryfunc(self.mhlib.MH_GetHardwareInfo(self.dev[0], \
self.hwModel, self.hwPartno, self.hwVersion), "GetHardwareInfo")
        print("Found Model %s Part no %s Version %s" % (
            self.hwModel.value.decode("utf-8"),
self.hwPartno.value.decode("utf-8"), \
self.hwVersion.value.decode("utf-8")))
self.tryfunc(self.mhlib.MH_GetNumOfInputChannels(ct.c_int(self.dev[0])\
, byref(self.numChannels)), "GetNumOfInputChannels")
        print("Device has %i input channels." % self.numChannels.value)
        self.tryfunc(self.mhlib.MH_SetSyncDiv(ct.c_int(self.dev[0]), \
ct.c_int(self.syncDivider)), "SetSyncDiv")
    self.tryfunc(self.mhlib.MH_SetSyncEdgeTrg(ct.c_int(self.dev[0]), \
ct.c_int(self.syncEdgeTrg), ct.c_int(1)), "SetSyncEdgeTrg")
self.tryfunc(self.mhlib.MH_SetSyncChannelOffset(ct.c_int(self.dev[0]),\
ct.c_int(self.syncChannelOffset)), "SetSyncChannelOffset")
# we use the same input settings for all channels, you can change this
        for i in range(0, self.numChannels.value):
   self.tryfunc(self.mhlib.MH_SetInputEdgeTrg(ct.c_int(self.dev[0]),\
ct.c_int(i), ct.c_int(self.inputEdgeTrg), ct.c_int(1)), \
"SetInputEdgeTrg")
self.tryfunc(self.mhlib.MH_SetInputChannelOffset(ct.c_int(self.dev[0])\
, ct.c_int(i), ct.c_int(self.inputChannelOffset)), \
"SetInputChannelOffset")
self.tryfunc(self.mhlib.MH_SetHistoLen(ct.c_int(self.dev[0]), \
ct.c_int(self.MAXLENCODE), byref(self.histLen)), "SetHistoLen")
        print("Histogram length is %d" % self.histLen.value)
        self.tryfunc(self.mhlib.MH_SetBinning(ct.c_int(self.dev[0]), \
ct.c_int(self.binning)), "SetBinning")
        self.tryfunc(self.mhlib.MH_SetOffset(ct.c_int(self.dev[0]), \
ct.c_int(self.offset)), "SetOffset")
     self.tryfunc(self.mhlib.MH_GetResolution(ct.c_int(self.dev[0]), \
byref(self.resolution)), "GetResolution")
        print("Resolution is %1.1lfps" % self.resolution.value)
       self.tryfunc(self.mhlib.MH_GetSyncRate(ct.c_int(self.dev[0]), \
byref(self.syncRate)), "GetSyncRate")
        print("Syncrate=%1d/s" % self.syncRate.value)
        time.sleep(0.2)
# Size of Histogram
        if self.syncRate.value == 0:
           self.blocklength = int((1 / 80e6) / (80 * 10 ** (-12))) + 10
# in case we are not connected to a sync signal the script still runs
        else:
          self.blocklength = int((1 / self.syncRate.value) / (80 * 10 \
** (-12))) + 10
    def recordhist(self, tacq, outputfile, channel):
        histogram=[]
# clearing last record
```

```
        self.tryfunc(self.mhlib.MH_ClearHistMem(ct.c_int(self.dev[0])),\
"ClearHistMem")
# start measuring
        self.tryfunc(self.mhlib.MH_StartMeas(ct.c_int(self.dev[0]), \
ct.c_int(tacq)), "StartMeas")
# print("\nMeasuring for %1d milliseconds..." % tacq)
        ctcstatus = ct.c_int(0)
        while ctcstatus.value == 0:
        self.tryfunc(self.mhlib.MH_CTCStatus(ct.c_int(self.dev[0]), \
byref(ctcstatus)), "CTCStatus")
# stop measuring
        self.tryfunc(self.mhlib.MH_StopMeas(ct.c_int(self.dev[0])), \
"StopMeas")
# return information
      self.tryfunc(self.mhlib.MH_GetHistogram(ct.c_int(self.dev[0]), \
byref(self.counts[channel]), ct.c_int(channel)), "GetHistogram")
        self.tryfunc(self.mhlib.MH_GetFlags(ct.c_int(self.dev[0]), \
byref(self.flags)), "GetFlags")
        if self.flags.value & self.FLAG_OVERFLOW > 0:
            print("  Overflow.")
        intensitysum = 0
# record of information
        for k in range(0, self.blocklength):
# regist the bins height of each channel
            outputfile[channel].write(str(self.counts[channel][k]))
            outputfile[channel].write(',')
# regist the intensity (so far for one single channel)
            histogram.append(str(self.counts[channel][k]))
            intensitysum = intensitysum + self.counts[channel][k]
        outputfile[channel].write('\n')
        return intensitysum, histogram
```

A.2 Code for Data Analysis

 # FLIM

```
def fittingrt(self, histogram):
#Fitting in real-time
    ti=[]
    tadd=0
    stop=0
    for i in range(len(histogram)):
        ti.append(tadd)
        tadd = tadd + 80*10**(-3)
        max_val = max(histogram)
        max_index = histogram.index(max_val)
        while ti[stop] < 12:
            stop = stop + 1
        ydata = np.array(histogram[max_index:stop])
        xdata = np.array(ti[max_index:stop])
        fittype=str(self.fittingtype.currentText())
        if fittype == 'Single Fitting':
            def monoExp(x, m1, t1, b):
                return m1 * np.exp(-t1 * x) + b
            try:
                popt, pcov = scipy.optimize.curve_fit(monoExp, xdata, \
ydata, p0=(1, 1e-9, 1))
                m1, t1, b = popt
                tmedio = t1
```

```
                    weight1 = m1 / m1
                    weight2=0
                    weight3=0
                    tau1 = 1 / t1
                    tau2=0
                    tau3=0
                    taumedio = (1 / tmedio)
                except:
                    tmedio=0
                    weight1=1
                    weight2=0
                    weight3=0
                    tau1=0
                    tau2=0
                    tau3=0
                    taumedio = 0
            if fittype == 'Double Fitting':
                def monoExp(x, m1, t1, m2, t2, b):
                    return m1 * np.exp(-t1 * x) + m2 * np.exp(-t2 * x) + b
                ydata = np.array(histogram[max_index:stop])
                xdata = np.array(ti[max_index:stop])
                try:
                    popt, pcov = scipy.optimize.curve_fit(monoExp, xdata, \
ydata, p0=(1, 1e-9, 1, 1e-9, 1))
                    m1, t1, m2, t2, b = popt
                    tmedio=t1+t2/2
                    weight1=m1/(m1+m2)
                    weight2= m2/(m1+m2)
                    weight3=0
                    tau1=1/t1
                    tau2=1/t2
                    tau3=0
                    taumedio = (1 / tmedio)
                except:
                    tmedio=0
                    weight1=1
                    weight2=0
                    weight3=0
                    tau1=0
                    tau2=0
                    tau3=0
                    taumedio = 0
            if fittype == 'Triple Fitting':
                def monoExp(x, m1, t1, m2, t2, m3, t3, b):
                    return m1 * np.exp(-t1 * x) + m2 * np.exp(-t2 * x) + \
m3  * np.exp(-t3 * x) + b
                ydata = np.array(histogram[max_index:stop])
                xdata = np.array(ti[max_index:stop])
                try:
                    popt, pcov = scipy.optimize.curve_fit(monoExp, xdata, \
ydata, p0=(1, 1e-9, 1, 1e-9, 1, 1e-9, 1))
                    m1, t1, m2, t2, m3, t3, b = popt
                    tmedio=t1+t2+t3/3
                    weight1=m1/(m1+m2+m3)
                    weight2=m2/(m1+m2+m3)
                    weight3= m3/(m1+m2+m3)
                    tau1=1/t1
                    tau2=1/t2
                    tau3=1/t3
                    taumedio = (1 / tmedio)
                except:
```

```
                tmedio=0
                weight1=1
                weight2=0
                weight3=0
                tau1=0
                tau2=0
                tau3=0
                taumedio = 0
        return taumedio, weight1, weight2, weight3, tau1, tau2, tau3
    def lifetimesplitting(self, tau1, tau2, tau3, weight1, weight2, \
weight3, j, i, intensity):
#Lifetimes splitting
        threshold = self.intensitythresholdvalue.value()
        fittype = str(self.fittingtype.currentText())
        minred = self.minred.value()
        maxred = self.maxred.value()
        mingreen = self.mingreen.value()
        maxgreen = self.maxgreen.value()
        minblue = self.minblue.value()
        maxblue = self.maxblue.value()
        trashval=0
        retval = [0, 0, 0, 0]
        if fittype == 'Single Fitting':
            tau1 = tau1[j, i]
            weight1 = weight1[j, i]
            if tau1 > minred and tau1 <= maxred and \
intensity>threshold:
                retval[0] = weight1
            elif tau1 > mingreen and tau1 <= maxgreen and \
intensity>threshold:
                retval[1] = weight1
            elif tau1 > minblue and tau1 <= maxblue and \
intensity>threshold:
                retval[2] = weight1
            else:
                trashval = trashval + weight1
            retval[3]=trashval
            return retval
        if fittype == 'Double Fitting':
            tau1 = tau1[j, i]
            tau2 = tau2[j, i]
            weight1 = weight1[j, i]
            weight2 =weight2[j, i]
            if tau1 > minred and tau1 <= maxred and \
intensity>threshold:
                retval[0] = weight1
            elif tau1 > mingreen and tau1 <= maxgreen and \
intensity>threshold:
                retval[1] = weight1
            elif tau1 > minblue and tau1 <= maxblue and \
intensity>threshold:
                retval[2] = weight1
            else:
                trashval = trashval + weight1
            if tau2 >= minred and tau2 <= maxred and \
intensity>threshold:
                retval[0] = retval[0]+weight2
            elif tau2 >= mingreen and tau2 <= maxgreen and \
intensity>threshold:
                retval[1] = retval[1]+weight2
```

```
            elif tau2 >= minblue and tau2 <= maxblue and \
intensity>threshold:
                retval[2] = retval[2]+weight2
            else:
                trashval = trashval + weight2
            retval[3]=trashval
            return retval
        if fittype == 'Triple Fitting':
            tau1 = tau1[j, i]
            tau2 = tau2[j, i]
            tau3 = tau3[j, i]
            weight1 = weight1[j, i]
            weight2 =weight2[j, i]
            weight3 = weight3[j, i]
            if tau1 > minred and tau1 <= maxred and \
intensity>threshold:
                retval[0] = weight1
            elif tau1 > mingreen and tau1 <= maxgreen and \
intensity>threshold:
                retval[1] = weight1
            elif tau1 > minblue and tau1 <= maxblue and \
intensity>threshold:
                retval[2] = weight1
            else:
                trashval = trashval + weight1
            if tau2 >= minred and tau2 <= maxred and \
intensity>threshold:
                retval[0] = retval[0]+weight2
            elif tau2 >= mingreen and tau2 <= maxgreen and \
intensity>threshold:
                retval[1] = retval[1]+weight2
            elif tau2 >= minblue and tau2 <= maxblue and \
intensity>threshold:
                retval[2] = retval[2]+weight2
            else:
                trashval = trashval + weight2
            if tau3 >= minred and tau3 <= maxred and \
intensity>threshold:
                retval[0] = retval[0]+weight3
            elif tau3 >= mingreen and tau3 <= maxgreen and \
intensity>threshold:
                retval[1] = retval[1]+weight3
            elif tau3 >= minblue and tau3 <= maxblue and \
intensity>threshold:
                retval[2] = retval[2]+weight3
            else:
                trashval = trashval + weight3
            retval[3] = trashval
            return retval
    def lifetimesplittingimaging(self,weightR, weightG, weightB, trash):
    #Plot the matrixes from the split
        if self.flimimg_check.isChecked():
            minred = self.minred.value()
            maxred = self.maxred.value()
            mingreen = self.mingreen.value()
            maxgreen = self.maxgreen.value()
            minblue = self.minblue.value()
            maxblue = self.maxblue.value()
            xnp = int(self.X_Number_of_Pixels.value())
            ynp = int(self.Y_Number_of_Pixels.value())
            fig = plt.figure(3,figsize=(10, 7))
```

```
        plt.clf()
        fig.add_subplot(2, 2, 1)
        normR = plt.Normalize(0, 1)
        cmapR = matplotlib.colors.LinearSegmentedColormap.from_list \
("", ["black","red"])
        plt.imshow(weightR, interpolation='nearest', origin='lower', \
extent=(0,... xnp, 0, ynp), cmap=cmapR , norm=normR)
        plt.colorbar()
        plt.xlabel('x_pixel')
        plt.ylabel('y_pixel')
        if xnp < 8:
           plt.xticks(np.arange(0, xnp+1, 1))
            plt.yticks(np.arange(0, ynp+1, 1))
            plt.title('R'+']'+str(minred)+'; '+ str(maxred)+ '] ns')
            fig.add_subplot(2, 2, 2)
            normG = plt.Normalize(0, 1)
            cmapG = \
matplotlib.colors.LinearSegmentedColormap.from_list("", \
["black", "green"])
             plt.imshow(weightG, interpolation='nearest', \
origin='lower', extent=(0, xnp, 0, ynp), cmap=cmapG , norm=normG)
            plt.colorbar()
            plt.xlabel('x_pixel')
            plt.ylabel('y_pixel')
        if xnp < 8:
           plt.xticks(np.arange(0, xnp+1, 1))
           plt.yticks(np.arange(0, ynp+1, 1))
           plt.title('G'+']'+str(mingreen)+'; '+ str(maxgreen)+ '] ns')
           fig.add_subplot(2, 2, 3)
           normB = plt.Normalize(0, 1)
           cmapB = \
matplotlib.colors.LinearSegmentedColormap.from_list("", ["black", \
"blue"])
           plt.imshow(weightB, interpolation='nearest', origin='lower',\
extent=(0, xnp, 0, ynp), cmap=cmapB , norm=normB)
           plt.colorbar()
           plt.xlabel('x_pixel')
           plt.ylabel('y_pixel')
         if xnp < 8:
            plt.xticks(np.arange(0, xnp+1, 1))
            plt.yticks(np.arange(0, ynp+1, 1))
            plt.title('B'+']'+str(minblue)+'; '+ str(maxblue)+ '] ns')
            fig.add_subplot(2, 2, 4)
            normt = plt.Normalize(0, 1)
            cmapt = \
matplotlib.colors.LinearSegmentedColormap.from_list("", ["black", \
"pink"])
            plt.imshow(trash, interpolation='nearest', origin='lower', \
extent=(0, xnp, 0, ynp), cmap=cmapt , norm=normt)
            plt.colorbar()
            plt.xlabel('x_pixel')
            plt.ylabel('y_pixel')
         if xnp < 8:
            plt.xticks(np.arange(0, xnp+1, 1))
            plt.yticks(np.arange(0, ynp+1, 1))
            plt.title('excluded')
            plt.savefig(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests\Scan \
and mh150 test_V6_Enora_Tests/lifetime")
            plt.show()
            plt.pause(1e-3)
def FLIM(self, histogramlist, intensitymatrix):
```

```
#Repeat the FLIM analysis with different fitting and ranges
        xnp = int(self.X_Number_of_Pixels.value())
        ynp = int(self.Y_Number_of_Pixels.value())
        self.flimimg_check.setChecked(True)
        taumedio = np.zeros((xnp, ynp))
        taumedio[:]=np.nan
        weight1 = np.zeros((xnp, ynp))
        weight1[:]=np.nan
        weight2 = np.zeros((xnp, ynp))
        weight2[:]=np.nan
        weight3 = np.zeros((xnp, ynp))
        weight3[:]=np.nan
        tau1 = np.zeros((xnp, ynp))
        tau1[:]=np.nan
        tau2 = np.zeros((xnp, ynp))
        tau2[:]=np.nan
        tau3 = np.zeros((xnp, ynp))
        tau3[:]=np.nan
        weightR = np.zeros((xnp, ynp))
        weightR[:]=np.nan
        weightG = np.zeros((xnp, ynp))
        weightG[:]=np.nan
        weightB = np.zeros((xnp, ynp))
        weightB[:]=np.nan
        trash = np.zeros((xnp, ynp))
        trash[:] = np.nan
        listoflifetimes=[]
        i = 0
        j = 0
        self.changelifebolean=False
        while j != ynp and i != xnp:
            histogram = histogramlist[j][i]
            intensity = intensitymatrix[j,i]
            taumedio[j, i], weight1[j, i], weight2[j, i], \
weight3[j, i], tau1[j, i], tau2[j, i], tau3[j, i] = \
self.fittingrt(histogram)
            weightR[j, i], weightG[j, i], weightB[j, i], trash[j, i] \
=self.lifetimesplitting(tau1, tau2, tau3, weight1, weight2, weight3, \
j, i, intensity)
            threshold = self.intensitythresholdvalue.value()
            if tau1[j, i] != 0 and intensitymatrix[j, i] > threshold \
and tau1[j, i] < 12.5 and tau1[j, i] >= 0:
                listoflifetimes.append(tau1[j, i])
            if tau2[j, i] != 0 and intensitymatrix[j, i] > threshold \
and tau2[j, i] < 12.5 and tau2[j, i] >= 0:
                listoflifetimes.append(tau2[j, i])
            if tau3[j, i] != 0 and intensitymatrix[j, i] > threshold \
and tau3[j, i] < 12.5 and tau3[j, i] >= 0:
                listoflifetimes.append(tau3[j, i])
            i = i + 1
            if i == xnp:
                i = 0
                j = j + 1
        self.lifetimesplittingimaging(weightR, weightG, weightB, trash)
        try:
            self.histogramlifetimes(listoflifetimes)
        except:
            print('no lifetimes')
```

 # Histograms

```
    def drawhistograms(self,data):
```

```
        fig2=plt.figure(2)
        plt.clf()
        xpos=self.xposhistogram.value()-1
        ypos=self.yposhistogram.value()-1
        fittype = str(self.fittingtype.currentText())
        x=[]
        y=[]
        ti=0
        stop=0
        for i in range(len(data[xpos][ypos])):
            x.append(ti)
            ti=ti+80*10**(-3)
        for i in data[xpos][ypos]:
            y.append(i)
        max_val = max(y)
        max_index = y.index(max_val)

        while x[stop] < 12:
            stop = stop + 1
        y = np.array(y[max_index:stop])
        x = np.array(x[max_index:stop])
        plt.plot(x,y)
        if fittype == 'Single Fitting':
            def monoExp(x, m1, t1, b):
                return m1 * np.exp(-t1 * x) + b
            try:
                popt, pcov = scipy.optimize.curve_fit(monoExp, x, y, \
p0=(1, 1e-9, 1))
                m1, t1, b = popt
                tau1= 1/t1
                plt.plot(x, monoExp(x, *popt),'r--', label= "tau1 = \
%5.3f ns" % tau1)
                plt.legend(loc="best")
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
             plt.savefig(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests \
\Scan  and mh150... test_V6_Enora_Tests/histogram of pixel"+
str(xpos)+", "+str(ypos))
                plt.show()
                plt.pause(1e-3)
            except:
                print('cant fit curve')
                plt.plot(x, y)
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
                plt.show()
                plt.pause(1e-3)
        if fittype == 'Double Fitting':
            def monoExp(x, m1, t1, m2, t2, b):
                return m1 * np.exp(-t1 * x) + m2 * np.exp(-t2 * x) + b
            try:
                popt, pcov = scipy.optimize.curve_fit(monoExp, x, y, \
p0=(1, 1e-9, 1, 1e-9, 1))
                m1, t1, m2, t2, b = popt
                plt.plot(x, monoExp(x, *popt),"r--",label= "tau1 = \
%5.3f ns, tau2 = %5.3f ns" % (1/t1, 1/t2))
                plt.legend(loc="best")
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
```

```
            plt.savefig(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests \
\Scan and mh150 test_V6_Enora_Tests/histogram of pixel" + \ str(xpos) \
+ ", " + str(ypos))
                plt.show()
                plt.pause(1e-3)
            except:
                print('cant fit curve')
                plt.plot(x, y)
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
                plt.show()
                plt.pause(1e-3)
        if fittype == 'Triple Fitting':
            def monoExp(x, m1, t1, m2, t2, m3, t3, b):
                return m1 * np.exp(-t1 * x) + m2 * np.exp(-t2 * x) + \
m3  * np.exp(-t3 * x) + b
            try:
                popt, pcov = scipy.optimize.curve_fit(monoExp, x, y, \
p0=(1, 1e-9, 1, 1e-9, 1, 1e-9, 1))
                m1, t1, m2, t2, m3, t3, b = popt
                plt.plot(x, monoExp(x, *popt),"r--", label= "tau1 = \
%5.3f ns, tau2 = %5.3f ns, tau3 = %5.3f ns" % (1/t1, 1/t2, 1/t3))
                plt.legend(loc="best")
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
            plt.savefig(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests \
\Scan and mh150 test_V6_Enora_Tests/histogram of pixel" + str(xpos) \
+ ", " + str(ypos))
                plt.show()
                plt.pause(1e-3)
            except:
                print('cant fit curve')
                plt.plot(x, y)
                plt.xlabel('t(ns)')
                plt.ylabel('intensity')
                plt.show()
                plt.pause(1e-3)
```

 # Lifetime Histograms

```
def histogramlifetimes(self,listoflifetimes):
    #consider threshold
    fig4=plt.figure(4)
    plt.clf()
    binwidth=float(self.binwidthval.value())
    bins=[]
for i in np.arange( float(min(listoflifetimes)),
float(max(listoflifetimes))+ binwidth, binwidth):
    bins.append(i)
    plt.hist(listoflifetimes,bins)
    plt.xlabel('lifetimes (ns)', fontsize=15)
    plt.ylabel('Frequency', fontsize=15)
    plt.title('Normal Distribution of lifetimes', fontsize=15)
    plt.savefig(rf"C:\Users\ExtreMedSyncRGB\Desktop\Tests\Scan and \
mh150 test_V6_Enora_Tests/histogram of lifetimes")
    plt.show()
    plt.pause(1e-3)
```

 # Intensity

```
def drawintensitymatrix(self, data, xnp, ynp):
    if self.intensityimg_check.isChecked():
       if self.autocolorbarbol==True:
```

```
#Automatic colorbar
          fig1=plt.figure(1)
          plt.clf()
          plt.imshow(data, interpolation='nearest', origin='lower', \
extent=(0, xnp, 0, ynp))
          plt.colorbar()
          plt.xlabel('x_pixel')
          plt.ylabel('y_pixel')
          if xnp<8:
             plt.xticks(np.arange(0, xnp+1, 1))
             plt.yticks(np.arange(0, ynp+1, 1))
              plt.show()
              plt.pause(1e-3)
           else:
#Choosen colorbar
              min=int(self.minintensitycolorbar.value())
              max=int(self.maxintensitycolorbar.value())
              fig1=plt.figure(1)
              plt.clf()
              plt.imshow(data, vmin=min, vmax=max, \
interpolation='nearest', origin='lower', extent=(0, xnp, 0, ynp))
              plt.colorbar()
              plt.xlabel('x_pixel')
              plt.ylabel('y_pixel')
              if xnp < 8:
                 plt.xticks(np.arange(0, xnp+1, 1))
                 plt.yticks(np.arange(0, ynp+1, 1))
                 plt.show()
                 plt.pause(1e-3)
```

 # Load files

```
def loadFile(self):
    Tk().withdraw() # we don't want a full GUI, so keep the root window
from appearing
    filename = askopenfilename()  # show an "Open" dialog box and
return the path to the selected file
    inputfile = open(str(filename), "r")
    inputread = inputfile.read()
    self.autocolorbarbol = True
    listmh = []
    ti = []
    t1 = 0
    x = []
    s = ''
    for i in range(len(inputread)):
        s = s + inputread[i]
        listmh = s.split()
        h = 0
    for i in range(len(listmh)):
        if listmh[i] == 'counts:' and h == 0:
            h = i
    for i in range(len(listmh)):
        if listmh[i] == 'sizeX':
           xnp=int(listmh[i+2])
        if listmh[i] == 'sizeY':
           ynp=int(listmh[i+2])
    self.X_Number_of_Pixels.setValue(xnp)
    self.Y_Number_of_Pixels.setValue(ynp)
    intensitymatrix = np.zeros((xnp, ynp))
    intensitymatrix[:] = np.nan
    histogramlist = np.zeros((xnp, ynp), dtype=object)
```

```
        histogramlist[:] = np.nan
        taumedio = np.zeros((xnp, ynp))
        taumedio[:] = np.nan
        weight1 = np.zeros((xnp, ynp))
        weight1[:] = np.nan
        weight2 = np.zeros((xnp, ynp))
        weight2[:] = np.nan
        weight3 = np.zeros((xnp, ynp))
        weight3[:] = np.nan
        tau1 = np.zeros((xnp, ynp))
        tau1[:] = np.nan
         tau2 = np.zeros((xnp, ynp))
         tau2[:] = np.nan
         tau3 = np.zeros((xnp, ynp))
         tau3[:] = np.nan
         weightR = np.zeros((xnp, ynp))
         weightR[:] = np.nan
         weightG = np.zeros((xnp, ynp))
         weightG[:] = np.nan
         weightB = np.zeros((xnp, ynp))
         weightB[:] = np.nan
         trash = np.zeros((xnp, ynp))
         trash[:] = np.nan
         listoflifetimes = []
         for j in range(xnp):
             for i in range(ynp):
                 h=h+1
                 intensitysum=0
                 histogram = listmh[h].split(',')
                 histogram.pop()
              for l in range(0, len(histogram)):
                  histogram[l] = int(histogram[l])
                  intensitysum=intensitysum+histogram[l]
                  intensitymatrix[j, i] = intensitysum
                  histogramlist[j][i] = histogram
        self.drawintensitymatrix(intensitymatrix, xnp, ynp)
        self.autocolorbar.clicked.connect(lambda: \
    self.autocolorbarfunc( intensitymatrix, xnp, ynp))
        self.chosencolorbar.clicked.connect(lambda: \
    self.chosencolorbarfunc(intensitymatrix, xnp, ynp))
        self.flim.clicked.connect(lambda: self.FLIM(histogramlist, \
    intensitymatrix))
        self.pushButtonhist.clicked.connect(lambda: \
    self.drawhistograms(histogramlist))
```