Università degli studi di Padova

Dipartimento di Fisica e Astronomia 'G. Galilei'

Laurea magistrale in Fisica curriculum sperimentale

3D-reconstruction of connexin32 channels distribution in the myelin of Schwann cells by advanced optical microscopy

Laureando: **Mirko Zanon**

Relatore: **Dr. Mario Bortolozzi**

Controrelatore: **Dr.ssa Tiziana Cesca**

Correlatori: **A. Leparulo, M. Gintoli**

Anno accademico **2015/2016**

Abstract

Connexin 32 (Cx32) is a 32 kDa protein of the connexin family that is expressed in the peripheral nervous system where it localizes in the myelin sheath of Schwann cells. Mutations of Cx32 are the leading cause of the X-linked form of Charcot–Marie–Tooth disease (CMT1X), a peripheral neuropathy for which there is no cure. Alteration in the distribution and function of Cx32 channels are presumed to trigger the neuropathy, but the pathological mechanism is still unknown. In this thesis work we combined two-photon fluorescence microscopy with third harmonic generation of the myelin sheath of Schwann cells to analyze the distribution of Cx32 and 3D render it by a software we developed in Matlab. STED microscopy tests were carried out with the future perspective to obtain high-resolution 3D images of Cx32 distribution in nerve samples of CMT1X patients.

Sommario

La connessina 32 (Cx32) è una proteina della famiglia delle connessine con massa pari a 32 kDa che viene espressa nel sistema nervoso periferico al livello della guaina mielinica delle cellule di Schwann. Le mutazioni della Cx32 sono la principale causa della neuropatia di Charcot-Marie-Tooth X-linked (CMT1X), per la quale non vi è ancora una cura. Si suppone che le alterazioni nella distribuzione e nella funzione dei canali di Cx32 siano alla base dello sviluppo della neuropatia, anche se il meccanismo patologico è ancora sconosciuto. In questo lavoro di tesi sono state utilizzate tecniche di microscopia di fluorescenza a due fotoni e generazione di terza armonica della mielina per analizzare la distribuzione di Cx32 nelle cellule di Schwann ed ottenere una mappa 3D mediante un software sviluppato in Matlab. Sono stati inoltre condotti test di microscopia STED per ottenere in prospettiva futura delle immagini tridimensionali a super risoluzione della distribuzione di Cx32 in campioni di nervo da pazienti CMT1X.

Index

I. Introduction

1. Connexins

Connexins are a family of transmembrane proteins that form gap junctions (GJs), specialized intercellular connections that directly connect the cytoplasm of adjacent cells, allowing molecules, ions and electrical impulses to pass through the cells; in the case of the myelin sheath they may also connect different compartments of the same cell. One gap junction channel is composed of two apposed hemichannels (from opposing cells), each of which is composed by six connexins arranged around a central pore; hemichannels of uniform connexin composition are called homomeric, otherwise heteromeric (Figure 1C). The GJ channel pore measures about 1.2 nm in diameter allowing the passage of molecules smaller than 1000 Da, including ions and second messengers.

Gap junctions show voltage-sensitive gating and are sensitive to pH, calcium ion concentration, and phosphorylation of residues in the carboxy terminus; GJs are also essential for many physiological processes, such as the coordinated depolarization of cardiac muscle, embryonic development and growth control, cellular differentiation: this is why mutations in connexins-encoding genes can lead to functional and developmental abnormalities.

Connexins are found only in vertebrates, while the responsible for GJs in invertebrates are the innexins.

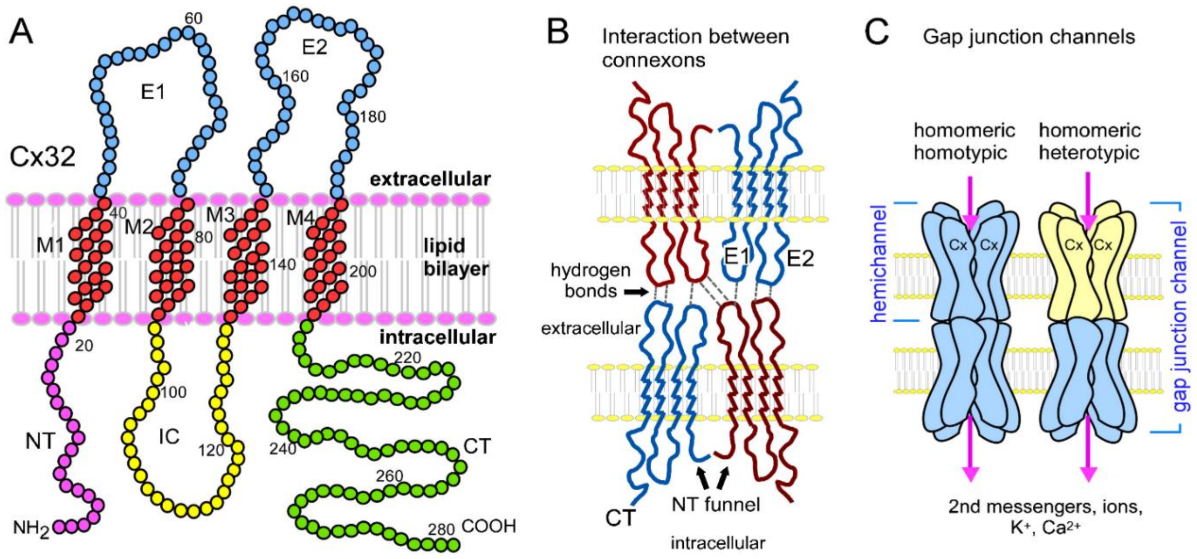


Figure 1 – Connexins structure

The structure of connexins consists basically in a cytoplasmatic amino terminus, four transmembrane domains with alpha helix structure, one intracellular and two extracellular loops, and a cytoplasmic carboxy terminus [1, 2].

The most variable domains are the intracellular loop and the carboxy terminus, which also determine the differences in their molecular mass. Hemichannels interaction with themselves are mediated via hydrogen bonds between apposing extracellular domains (Figure 1A-B).

The most common way to name connexins is accordingly to their molecular weights expressed in kilo Dalton, thus, for example, Cx32 is the connexins protein of 32 kDa. They usually weigh between 26 and 60 kDa and have an average length of 380 amino acids.

Some of the most common connexins are:

- Cx26: mutated in Vohwinkel syndrome as well as Keratitis-Icthyosis-Deafness Syndrome
- Cx29: present in innermost layer of myelin in Schwann cells (but not known to form gap junctions)
- Cx30: mutated in Clouston syndrome
- Cx30.2: expressed in structures of the inner ear
- Cx32: major component of the peripheral myelin, mutated in Charcot-Marie-Tooth disease
- Cx36: pancreatic function mediating insulin release
- Cx37: induced in vascular smooth muscle during coronary arteriogenesis.
- Cx40: responsible for mediating the coordinated electrical activation of atria.
- Cx43: expressed at the surface of vasculator with atherosclerotic plaque and found mainly in ventricular myocardium; associated with oculodentodigital dysplasia
- Cx47: expressed in oligodentrocyte
- Cx50: between A-typ horizontal cells in mouse and rabbit retina
- Cx62: in B-typ horizontal cell in rabbit retina.

Connexin32 (Cx32)

Cx32 is a protein encoded in humans by the GJB1 gene and it was the first connexin to be cloned; it is highly conserved, as the amino acid sequence of the human Cx32 protein is 98% identical to those of the mouse and rat Cx32. It was discovered that mutations affecting this connexins occur in patients with the X-linked Charcot-Marie-Tooth neuropathy (CMT1X, hereditary motor and sensory neuropathy); CMT causes slowly progressive weakness and atrophy of the distal limb muscles, as well as pes cavus, sensory loss, and loss of deep tendon reflexes; this is the second most common CMT form caused by over 400 different mutations in the GJB1 gene. In peripheral nerves, Cx32 is expressed by Schwann cells and forms GJs through non-compact myelin areas.

Both in vitro and in vivo models of CMT1X indicate that most Cx32 mutations cause loss of function and inability of the mutant Cx32 to form functional GJs[3]. GJB1 mutations usually cause only peripheral neuropathy, and occasionally CNS phenotypes; however, Cx32 is expressed in many cell types besides Schwann cells and oligodendrocytes, and in fact most abundantly in the liver. It is generally accepted that co-expression of other connexins in non-neural cell types and tissues may provide at least partial functional redundancy, protecting against the loss of Cx32. Myelinating Schwann cells express Cx29 (the human orthologue is Cx31.3) in addition to Cx32, but Cx29/Cx31.3 does not prevent the development of demyelinating neuropathy and appears to form hemichannels but not GJs. Oligodendrocytes have a partial redundancy for connexins as they all express Cx47 in addition to Cx32, but they are more vulnerable in CMT1X patients, as indicated by CNS phenotypes under stress conditions in CMT1X patients. Nevertheless, loss of both Cx32 and Cx47 in the CNS causes severe demyelination in mice, and loss-of-function mutations in GJC2 encoding Cx47 in humans cause Pelizeaus–Merzbacher-like disease, a severe hypomyelinating leukodystrophy[4-6].

GJB1 mutations have shown that disability increases with age and that the degree of disability was comparable with that observed in patients with a documented GJB1 deletion.

2. Peripheral nervous system

The human nervous system contains a huge number of neurons (about 10^{11}), classifiable in a thousand different types; nevertheless, the cells of the nervous system are amenable to one single basic structure: the complexity of human behavior doesn't derive from the different specialization of these cells, but rather in their specific and particular organization and connection.

In the nervous system there are two main classes of cells: nervous cells (neurons) and glial cells (glia).

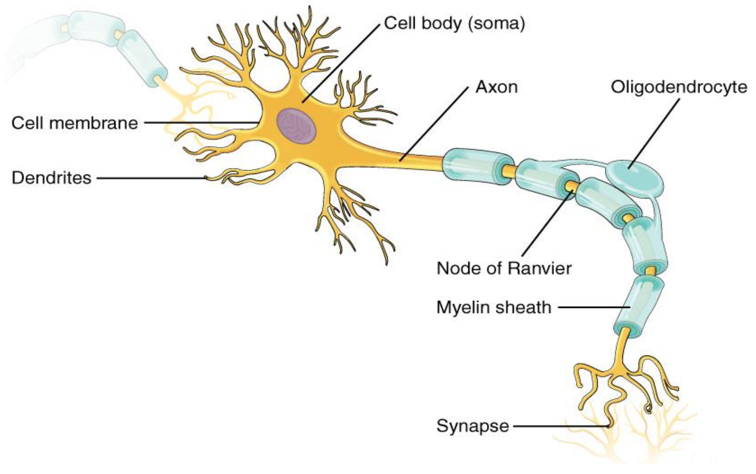


Figure 2 - Scheme of a neuron

In a neuron we can distinguish four morphologically distinct regions (Figure 2):

- the cell body (soma), the metabolic center that contains the nucleus and the endoplasmic reticulum for the protein biosynthesis;
- the dendrites that form a sort of arborization for the reception of signals;
- the axon for the transmission of signals;
- the synapses (terminal or synaptic buttons), biological junctions through which neurons signal to each other and to non-neuronal cells such as those in muscles or glands.

How we have just said, the main element for the conduction of messages between neurons is the axon; the majority of the axons in the central nervous system is very thin (0,2 to 20 µm of diameter) respect to cell body (50 µm or more). Axon extends itself for long distances far from the cell body and it's able to transmit electrical signals to distances from 0,1 mm to 3 m. These signals, fast and

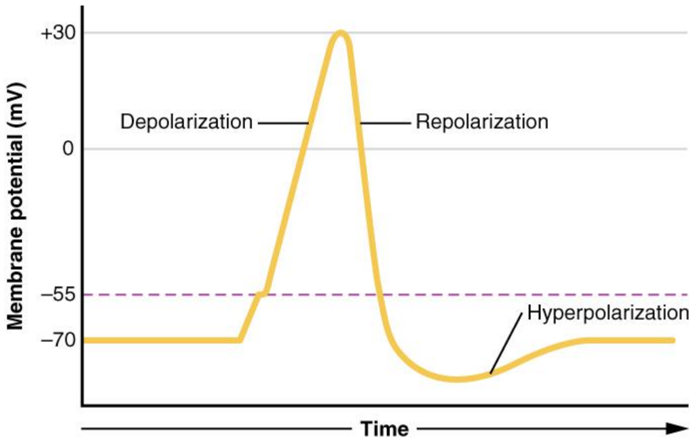


Figure 3 - Example of an action potential

transitory nerve impulses of the order of 100 mV with a duration of 1 ms, are called action potentials (Figure 3). They are highly stereotyped in all the nervous system even if they are originated form a very large variety of events and stimuli: this means that the information transported by the action potential doesn't depend on the morphology of the signal but rather on the path that the signal treads in the brain; it's the brain that analyzes and deciphers the nature of these signals, transforming them in everyday sensations. To increase the velocity of action potentials conduction, the bigger axons are surrounded by a lipid and insulating sheath of myelin; this sheath is interrupted regularly by sites without coating, the Ranvier's nodes, where the energy of the action potential regenerates[7-9].

These last are the other class of cells, represented by *glia*: in vertebrates, they are from 10 to 50 times more numerous than neurons. They surround cell bodies, axons and dendrites; the most important functions are those of supporting to give form and structure to nervous tissue, originating myelin (due to oligodendrocytes and Schwann cells), removing cell fragments created after lesions or death of nervous cells, maintenance for the efficiency of signals transmission, directing the growth of axons (due to radial glia), and insulating brain capillaries preventing the entrance of toxic substances deriving from the blood (due to astrocytes).

While astrocytes and oligodendrocytes are part of the central nervous system, Schwann cells (Gliocytus periphericus, neurolemmocytes), named after physiologist Theodor Schwann, are the principle glial cells of the peripheral nervous system.

There are two types of neurolemmocytes, the myelinating ones, that wrap around axons forming the myelin sheath (Figure 4), and nonmyelinating[10].

But the roles of these glia are even more than the only myelin production and protection of neurons: they are also involved in the conduction of nervous impulses along axons (propagated at nodes of Ranvier), nerve development and regeneration, modulation of neuromuscular synaptic activity, production of the nerve extracellular matrix, among the most important functions.

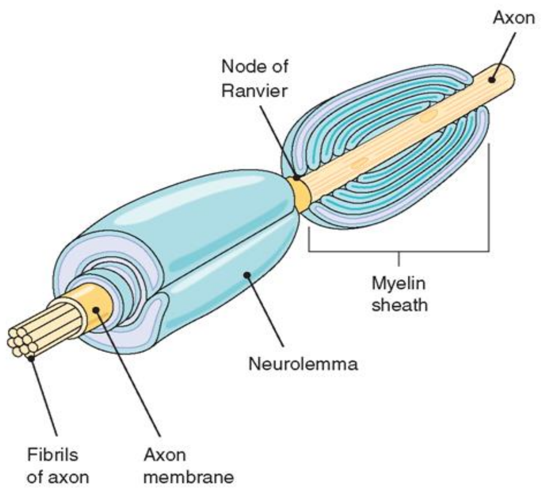


Figure 4 - Scheme of a Schwann cell

In general, at the ultrastructural level, the components of the extracellular matrix are composed by two domains: an internal basal lamina linked to cellular membranes, and an external fibrillary matrix. Originally it was thought that these components had only a static function of separation or filtering in tissues; but now it is also observed that the basal lamina components are cell-specific and moreover they not only maintain the structural integrity of the lamina but also interact with receptors in the cell membrane, that trigger intracellular signals (controlling many functions such as cellular migration, proliferation, polarization, shape). It is interesting to note that these functions of the lamina evolve dynamically with the changes in time and space of its components; at the same time the receptors of the extracellular matrix themselves modify: it's a mutual modification. Specializing the treatment in the case of Schwann cells, the fact that lamina components, like laminin and collagen, are implicated in myelin formation suggest a crucial role for neurolemmocytes basal lamina: correlating the regulation of its components and of their receptors to a unique function (like myelination) gives a view on the specific role and evolution of this lamina[11]. The relation of the lamina to domain formation is for instance at the basis of polarization, responsible for fast propagation of action potentials by myelinated nerve fibers. In addition, despite the lack of direct evidence and the nescience about the mechanisms that might regulate the length of the myelinated segments, internodal distances (between Ranvier's nodes -Figure 4-) have been proposed to affect the velocity of nerve impulse conduction; moreover it was suggested bands of cytoplasm in internodal neurolemmocytes to have a nutritive function; internodal growth in nerves is matched to nerve extension, and disruption of cytoplasmic bands impairs Schwann cell elongation during nerve growth: decreased internodal distances decrease conduction velocities, affecting motor function [12, 13].

But characterize the functions for all ligands and receptors is a very hard work, and in vitro studies have only partially predicted the roles for the extracellular matrix; moreover, the relation between Schwann cell and axon is also very complex. For all these reasons there's a lot to do in this direction, and an important task, how we have seen, could be the detailed study of the cell morphological structure.

Cx32 in Schwann cells

Although Cx32 is most abundant in liver, it is also expressed by many tissues, including kidney, intestine, lung, spleen, stomach, pancreas, uterus, testes, brain, and peripheral nerve. Despite this broad expression pattern, peripheral neuropathy is the sole clinical manifestation of Cx32 mutations: in this case Cx32 is expressed by Schwann cells and forms GJs through the non-compact myelin layers of peripheral myelinated fibers. The reason why the other tissues are not affected is unclear; perhaps the coexpression of another connexin protects them from the loss of Cx32[14-16].

Previous studies have shown that peripheral neuropathy in most X-linked Charcot-Marie-Tooth disease (CMT1X) affected patients results from loss of Cx32 function with rare exceptions in which toxic effects were demonstrated caused, for example, by leaky hemichannel formation of the mutant. CMTX1 results from numerous mutations in the GJB1 gene (encoding the gap junction protein Cx32) and is one of the commonest forms of inherited neuropathy. Owing to the expression of Cx32 not only in Schwann cells but also in oligodendrocytes, a subset of CMT1X patients develops central nervous system (CNS) clinical manifestations in addition to peripheral neuropathy (even if, while most GJB1 mutations appear to cause peripheral neuropathy through loss of Cx32 function, the cellular mechanisms underlying the CNS manifestations remain controversial [17, 18]).

The Cx32 start codon mutation is equivalent to a complete loss of the protein with failure of translation, although transcription is not impaired. Following HeLa cell transient transfection with the two different start codon mutant constructs as well as with WTCx32 and the V140E mutant, immunocytochemistry revealed that in cells transfected with the WT construct, Cx32 was localized at the cell membrane forming GJ-like plaques, while in cells transfected with the V140E mutant Cx32 was retained intracellularly; V140E and WTCx32 showed the predicted Cx32 immunoreactivity in the cytoplasm and on the cell membrane, respectively[19]. In transfected cells, the mutant Cx32 proteins (encoded by some Cx32 mutations) fail to reach the cell surface; other mutant proteins reach the cell surface, but only one of these forms functional gap junctions.

In peripheral nerve, Cx32 is localized to incisures and paranodes, regions of noncompact myelin within the myelin sheat; in fact, to determine where gap junctions might be in the myelin sheath, peripheral nerves were immunolabeled: Cx32 immunoreactivity was found in noncompact myelin, which is found in the paranodal regions and incisures of the myelin sheath (Figure 5)[20].

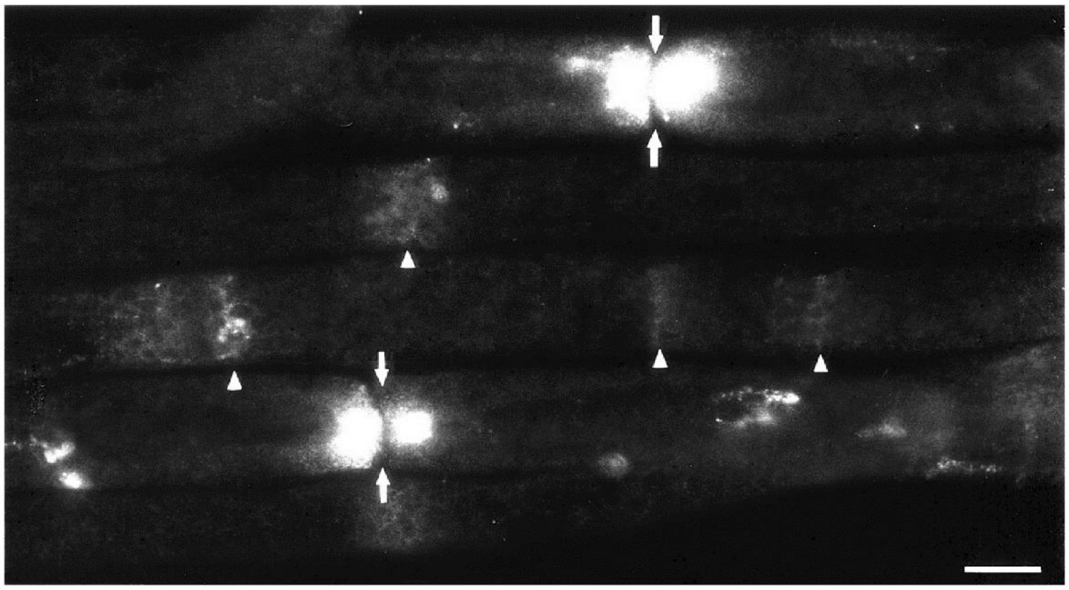


Figure 5 - Localization of Cx32 in the myelin sheath of teased fibers from a rat sciatic nerve immunolabeled with a monoclonal antibody against Cx32. Immunoreactivity is seen at incisures (arrowheads) and nodes of Ranvier (arrows). Scale bar 10µm.

If there were gap junctions in the incisures and paranodes, these could provide a direct pathway for the diffusion of ions and small molecules directly across the myelin sheath. For thick myelin sheaths, this direct, radial pathway would be 1000 times shorter than the circumferential pathway through the Schwann cell cytoplasm. Thus, myelinating Schwann cells may use ‘‘reflexive’’ gap junctions (gap junctions of a cell onto itself) to overcome the limited diffusion of ions and small molecules inherent to their specialized geometry.

If CMTX mutations interrupted the function of these gap junctions, then this could damage myelinating Schwann cells and their axons, leading to demyelination as well as axonal loss[20].

3. Microscopy techniques

In our experiments we are going to use fluorescence microscopy techniques: thanks to its selectivity, with this kind of microscopy it is possible to visualize features that normally are not visible with transmitted light microscopy; furthermore, in general, it has the capability to reduce consistently the noise given by other features or by the background.

Another characteristic is that we use scanner microscopy: this means that the laser beam is focalized and scanned through the sample; in this way it's possible to obtain higher excitation intensities (with a consequent better signal to noise ratio) and at the same time to avoid diffraction effects of a single element to influence the whole image.

It is to remember that the observation of sub-wavelength structures with microscopes is impossible because of the Abbe diffraction limit: Ernst Abbe found in 1873 that light with wavelength λ_0, traveling in a medium with refractive index *n* and converging to a spot with angle θ will make a spot with dimensions

$$\Delta r = \frac{0.61\ \lambda_0}{NA}$$

$$\Delta z = \frac{2\ \lambda_0\ n}{NA^2}$$

with $NA = n sin\theta$ the objective numerical aperture.

A fundamental problem of scanner microscopy is the poor axial confinement of the signal, that is generated from more focal planes, disturbing the vision of the plane one is interested in. To solve this problem confocal microscopy was introduced: in this way the noise is cut and images more resolute can be obtained. The resolution is then improved reaching:

$$\Delta r_{conf} = \frac{0.4\ \lambda_0}{NA} \cong 0.7\ \Delta r$$

$$\Delta z_{conf} = \frac{1.4\ \lambda_0\ n}{NA^2} \cong 0.7\ \Delta z$$

These techniques are surely capable of giving images that are resolute and clear, but in general only for thin samples, because of the presence of scattering and absorption that disrupt image quality after a certain depth (deeper than 20μm); multiphoton microscopy overcomes these problems.

Two-photons (and multi-photons) excitation microscopy (MPE)

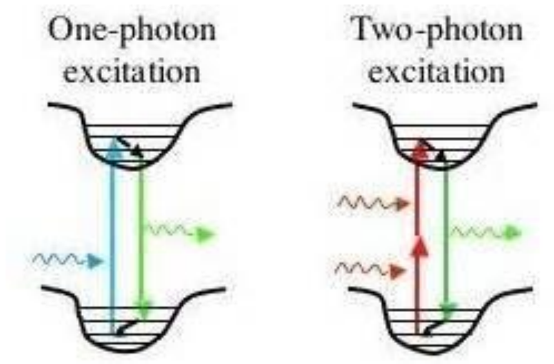


Figure 6 - Mechanism of 2PE

2PE is a fluorescence imaging technique used to study specimens up to about one millimeter in depth. It consists in two main quantum events (Figure 6):

- the absorption of two photons by a fluorophore (through the passage in an intermediate virtual state of lifetime ~10^{-16} s);
- the emission of a fluorescence photon after thermal relaxation upon excitation (with typical lifetime of 1-10ns).

The process is analogous to one photon excitation, but the absorption of two photons of doubled wavelength brings some advantages; first of all, using deep red or IR light minimizes scattering in the tissue and suppresses the background strongly: this leads to an increased penetration depth. Moreover, the axial spread of the point spread function is lower than that of single-photon excitation, improving the z-dimension resolution. Since the absorption processes involved in 2PE are statistically independent events, the total probability density is given by the square of the 1PE probability density h(u/2, v/2) calculated for the wavelength $\lambda_{1P} = \lambda_{2P}/2$, where u and v are the axial and radial optical coordinates. The resulting illumination PSF is

$$\boldsymbol{h}_{2P}(u, v) = \left[\boldsymbol{h}\left(\frac{u}{2}, \frac{v}{2}\right)\right]^2 .$$

which leads to imaging resolutions slightly worse than confocal microscopy (with about a factor 1.3 for lateral and 1.5 for axial resolution), as expected in theory, since they are linearly dependent on light wavelength, which is approximately doubled in 2PE. At the same time though, given the low 2PE probability, only the focal region is invested by a high enough optical density to emit a detectable signal: there's a localization of excitation without the need of a confocal scheme. As the filtering system is removed, both ballistic (not scattered) and scattered fluorescence photons that are collected by the objective are detected, returning a greater signal.

These advantages come at a cost: given the extremely small excitation rate, high photon fluxes (10^{30} photons/s cm^2) are needed to ensure a sufficient level of fluorophore excitation. Such powers are not reachable with continuous wave lasers, so that pulsed lasers have to be used: these concentrate low energies in very small time intervals (mainly of femto-/picoseconds pulse duration), allowing to obtain very high instantaneous irradiances to perform 2PE.

It is also important to note that the resolution limit is always imposed by diffraction.

Two-beam multicolor fluorescence imaging

It can be considered that when a sample is illuminated with two laser beams it can experience 2PE of two different chromophores, furthermore a new chromophore can be excited by the simultaneous absorption of two photons one for each wavelength, creating a third 'virtual' excitation wavelength: this is the so called three-color two-photon (3C2P) mechanism, and in order to obtain it the two pulses must be highly temporally and spatially synchronized (and suitable fluorophores obviously must be available)[21, 22].

Let us consider the energies of photons belonging to the two beams, namely $\epsilon_1 = hc/\lambda_1$ and $\epsilon_2 = hc/\lambda_2$. In order to realize 3C2P imaging, their sum has to be equal to the energy required for one-photon absorption:

$$\epsilon_{3(1p)} = \frac{hc}{\lambda_1} + \frac{hc}{\lambda_2}$$

whereas, for two-photon absorption,

$$\lambda_{(2p)} = 2\lambda_{3(1p)} = \frac{2}{\frac{1}{\lambda_1} + \frac{1}{\lambda_2}}$$

The three emission intensities, assuming Gaussian temporal profiles of the pulses and no overlap between excitation spectra, scale as

$$I_{\lambda_1} \sim (P_1)^2, \qquad I_{\lambda_2} \sim (P_2)^2, \qquad I_{\lambda_3} \sim 2P_1P_2 \exp\left(-\frac{\tau^2}{2\sigma^2}\right)$$

where $P_1$and P_2 are the two average beam powers, τ is the temporal delay and σ the half $1/e^2$ width of the temporal intensity intercorrelation function between the two pulses. Assuming Gaussian profiles for both incident pulses, temporal intensity intercorrelation function can be treated as autocorrelation function, whose $1/e^2$ width measures about $\sqrt{2}$-fold the pulse temporal width.

The described method is more convenient than other multicolor imaging techniques, such as single-beam excitation of multiple chromophores or sequential excitation at different wavelengths. For quantitative multicolor imaging it is crucial to have independent control of different chromophores signals; actually the main advantage in MC2P microscopy is the possibility to adjust simultaneously and independently the intensities of the different emissions, together with a reduction of acquisition time.

STED

The stimulated-emission-depletion (STED) microscopy is a technique that allows to obtain super-resolute images, breaking the limit of diffraction (typically in the order of hundreds of nanometers).

The basic principle is the stimulated emission studied by Einstein: a single photon, matching the gap between an excited state and a vibrational level of the ground state, interacts with a previously excited molecule, bringing it to the ground level and generating in this way a second photon that is indistinguishable from the first. When a population inversion is present, the rate of stimulated emission exceeds that of absorption and an optical amplification can be achieved (the base of the laser system). If we also use a wavelength which is on the emission spectrum of the molecule, but as far as possible from the absorption spectrum tail, we can achieve stimulated emission without the risk of a re-excitation by other photons of the same beam.
This process is conceptually the inverse of absorption, with a probability proportional to the intensity of the light incident on the molecule: increasing the intensity and so the probability, the mean time of the process reduces; when this time becomes shorter than the fluorescence mean time, it becomes possible to reduce the spontaneous signal (*depletion*), and further increasing the intensity, to completely turn off all the excited molecules (depletion *saturation*).

STED technique consists in using far-field microscopy to directly confine or isolate a molecular state (in particular a fluorescent state) by transiently switching neighboring molecules into the dark state with depletion[23-26]. Summarizing, there are two main steps in this STED procedure (Figure 7):

- a first beam excites the molecule in the geometrical focus of the microscope;
- a depletion beam with a donut-shaped intensity distribution brings back to the ground state the molecules around the focus axis.

The donut shaped depletion spot is obtained using a suitable optical phase plate (spiral phase plate [27]) which generates the central dark core by means of diffraction.

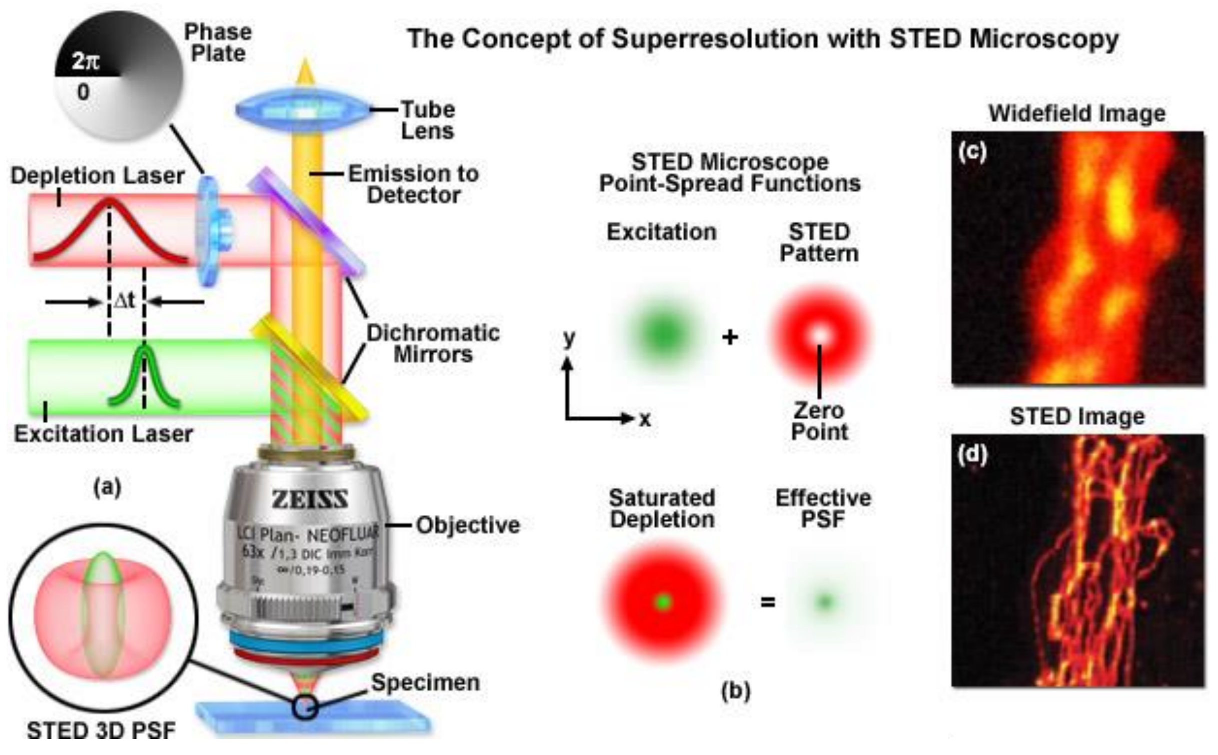


Figure 7 - Scheme of STED principle

The excitation pulse needs to ensure temporally defined excitation of the fluorophore, so its duration is shorter than the mean relaxation time of the vibrational levels of the excited states (τ_{vib}). The depletion pulse is stretched to durations much longer than τ_{vib}, since the high intensities involved in the STED process force the molecules back and forth between the ground and excited states, and a long time is required for the relaxation process to subtract the molecules from this cycle.

It's also possible to obtain a raw estimation of the increase in resolution generated by STED. In the focal plane, the two irradiance functions can be approximated as:

$$h(r) \cong \cos^2(ar)$$

$$h_{sted}(r) \cong \sin^2(ar)$$

where $r = \sqrt{(x^2 + y^2)}$ is the radial coordinate in the focal plane and $a = \frac{\pi}{2\delta r_{\min}}$, with $\delta r_{\min}$ the value of the first minimum of the probability density h(r), namely the diffraction limited resolution. After the incidence of the depletion pulse, the effective excitation PSF is a product between the excitation PSF *h* and the population of molecules still in the excited state n_1^{sted}:

$$\boldsymbol{h}_{eff}(v,u) = h(v,u)\, n_1^{sted}(v,u)$$

To evaluate the depletion efficiency we can refer to the depletion ratio *d*; for a Gaussian depletion pulse of duration τ_{sted}, it is defined as the ratio between the excited state population after a time $t = 4\tau_{sted}$ (with t=0 the starting point of the pulse) and the same population at the same time in case of no depletion:

$$d = \frac{n_1^{sted}(t = 4\tau_{sted})}{n_1(t = 4\tau_{sted})}$$

For a given $\tau_{sted} \ll \tau$ (τ spontaneous fluorescence lifetime), we can assume $n_1(t = 4\tau_{sted}) \approx 1$, hence

$$d \cong n_1^{sted}(t = 4\tau_{sted}) \cong \exp\left(-\frac{I_{sted}}{I_{sat}}\right)$$

I_{sat} is known as STED saturation irradiance and it is defined as the irradiance at which the depletion ratio is reduced by a factor 1/e. Typical values for I_{sat} are comprised between 1 and 100 MW/cm^2. After the depletion pulse is terminated, we can express the fluorescence probability of a molecule as

$$n_1^{sted}(v,u) \cong exp\left[-\frac{h_{sted}(v,u)I_{max}}{I_{sat}}\right]$$

with I_{max} is the peak irradiance of the depletion pulse, that is fixed by the source and the microscope setup, usually ~ 1 GW/cm^2 $\gg I_{sat}$. Substituting the approximated forms for excitation and depletion point spread functions in the previous expression and making use of the asymptotic series for sine, cosine and exponential up to the second order, we obtain

$$\boldsymbol{h}_{eff}(r) \approx 1 - \left(1 + \frac{I_{max}}{I_{sat}}\right)a^2 r^2.$$

Since resolution can be defined through the FWHM of the effective irradiance, namely twice the radial distance for which $h_{eff}(r) = 1/2$, we can invert the formula and solve for r:

$$r \approx \frac{1}{\sqrt{2}a\sqrt{1 + \frac{I_{max}}{I_{sat}}}}.$$

In conclusion, using $a = \pi NA/\lambda_0$ (NA numerical aperture), the new inferior limit for optical microscopy resolution is

$$\delta r_{sted} = 2r \approx \frac{\sqrt{2}}{\pi}\frac{\lambda_0}{NA}\frac{1}{\sqrt{1 + \frac{I_{max}}{I_{sat}}}}$$

The crucial change with respect to diffraction limited resolution is the presence of the factor $\frac{1}{\sqrt{1+\frac{I_{max}}{I_{sat}}}}$: the higher the ratio I_{max}/I_{sat}, the smaller the inferior distance that can be distinguished with the microscope. This theoretically infinite resolution is obviously limited, in particular for increasing I_{max}, by practical constraints such as photodamage of samples and the residual intensity in the central zero of the depletion donut, which causes the effective PSF to reduce in intensity, lowering the signal-to-noise ratio.

Higher harmonic generation (SHG, THG)

The phenomena of harmonic generation are optical nonlinear processes, due to higher order terms in the expression of the polarization P(t) of a medium in presence of external electrical field E(t). In fact, in nonlinear optics the optical response is described as a power series in the external field:

$$P(t) = \epsilon_0[\chi^{(1)}E(t) + \chi^{(2)}E^2(t) + \chi^{(3)}E^3(t) + \dots] = P^{(1)}(t) + P^{(2)}(t) + P^{(3)}(t) + \cdots$$

where ϵ_0 is the vacuum permittivity, $\chi^{(1)}$ the linear susceptibility, $\chi^{(2)}$ and $\chi^{(3)}$ the second and third order nonlinear susceptibilities

Second harmonic generation (SHG) is a nonlinear optical process in which photons with equal wavelength interact with a nonlinear material combining themselves in a new photon with twice the frequency and the energy. The principle is analogous for third harmonic generation (THG) too: it results from a tripling of the frequency of the excitation wavelength, because the combined energy of three photons is converted into one emitted photons with one third of the excitation wavelength. Physically, given a laser beam with electric field E(t) and frequency ω:

$E(t) = E_0 \cos(\omega t)$ and using the identity: $(cos\omega t)^3 = \frac{1}{4}\cos 3\omega t + \frac{3}{4}cos\,\omega t$

it results a third order polarization dependent from susceptibility χ:

$$P^{(3)}(t) = \epsilon_0\,\chi^{(3)}E^3(t) = \frac{1}{4}\epsilon_0\chi^{(3)}E_0^3 cos3\omega t + \frac{3}{4}\epsilon_0\,\chi^{(3)}E_0^3 cos\,\omega t$$

in which we can recognize the destruction of three photons of frequency ω and a construction of one with frequency 3ω .

If SHG occurs at ordered structures found mainly in metals and crystals or repetitive units in coiled structures and polymeric proteins (but not in media that present inversion symmetry, called "centrosymmetric"), on the other side THG can be used like a label-free scatter process dictated predominantly by structural interfaces, that's because it occurs when there is a local transitions of the refractive index or of third-order nonlinear susceptibility $\chi^{(3)}$ (e.g. water-lipid and water-protein interfaces, intra- and extracellular membranes, extracellular matrix structures…); in this way it can provide a versatile contrast technique complementary to fluorescence and SHG microscopy itself [28].

According to the momentum conservation law, the generated photons are mostly emitted in the same direction of the initial ones, that is in a forward direction away from the objective; this is why in transparent media it is better to detect with forward detectors directed towards the exiting laser beam. On the other side, with non-transparent samples, backward detection of photons is possible thanks to tissue backscattering; the weakness of this signal requires high-sensitivity detectors such as gallium arsenite phosphide (GaAsP) photomultipliers. Another advantage is that THG (and SHG too, like 2PE) is only generated in the focal volume: the signals are in this way intrinsically confocal.

The intensity of THG signal depends on power and polarization of the incident light, and strongly on the size of the structure and the numerical aperture of the objective (the resolution of the system); an optimal signal is generated by an orientation of the interface that is parallel to the excitation beam. So, what is to remember is that THG intensity depends strongly on local imaging conditions; another important characteristic is the excitation wavelength to use: the optimal one is determined by a trade-off between tissue scattering and water absorption of the laser light (longer wavelengths reduce scattering but are strongly absorbed by water, limiting tissue penetration). The detection of THG can be improved by using adaptive optics in order to achieve a sharp focus in light-scattering samples.

An interesting aspect is that THG (like SHG) is an energy-conserving process, because the emitted photon contains the total energy of the three exciting ones: in this way no photobleaching can be generated, being absorbed no energy by the structures (even if tissue damage can still happen due to the high illumination power required) [28-30].

Techniques used in this work

In this work all the previous techniques are used in a combined way, in order to obtain the best results and a multicolor image deriving from different channel each one highlighting different structures with different techniques (for example combined 2PE and THG).

4. 3D rendering

Confocal, multiphoton, and digital deconvolution microscopy have provided scientists with the powerful capability to collect high-resolution image volumes, an approach that is critical to characterizing the complex organization of cells and tissues. However, the proliferation of 3D microscopy has been slowed by the lack of effective and affordable systems for inspecting and evaluating image volumes.

To perform a 3D-rendering of the images taken at the microscope is another task of this work.

The importance of neuronal morphology has been recognized from the early days of neuroscience. Elucidating the functional roles of axonal and dendritic arbors in synaptic integration, signal transmission, network connectivity, and circuit dynamics requires quantitative analyses of digital three-dimensional reconstructions. Digital reconstructions can be acquired from any animal species, brain regions, neuron types, and a variety of experimental protocols. Morphological quantification is central in some studies, such as to investigate chemical, genetic, or behavioral effects on branch complexity. In other cases, morphology is traced as an aid, e.g., to document the identity of intracellularly recorded neurons. Once digitally reconstructed, however, morphological data can be re-used in secondary applications beyond the scope of the original project, including computational simulations, comparative analyses, and large-scale data mining across labs or techniques.

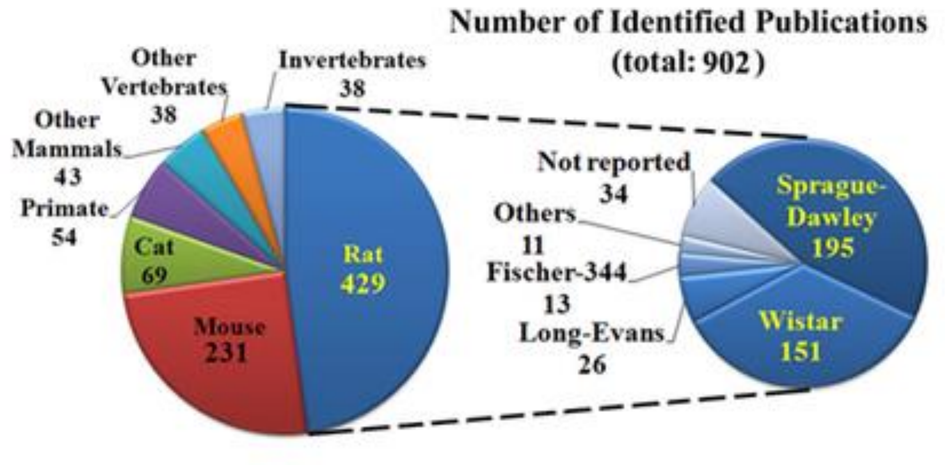


Figure 8 - Breakdown of the identified publications by animal species, with a further subdivision for rat strains 1997-2012

The temporal analysis of 902 identified publications describing digital reconstructions demonstrates a continuous growth starting from the mid 1990s; in the breakdown by animal species, rodents made up more than two-thirds of experimental subjects, with all invertebrates only accounting for just 4% of the reports (Figure 8) [31].

In Figure 9-10 it’s reported an example of 3D rendering of a mouse sciatic nerve derived from a stack [32, 33].

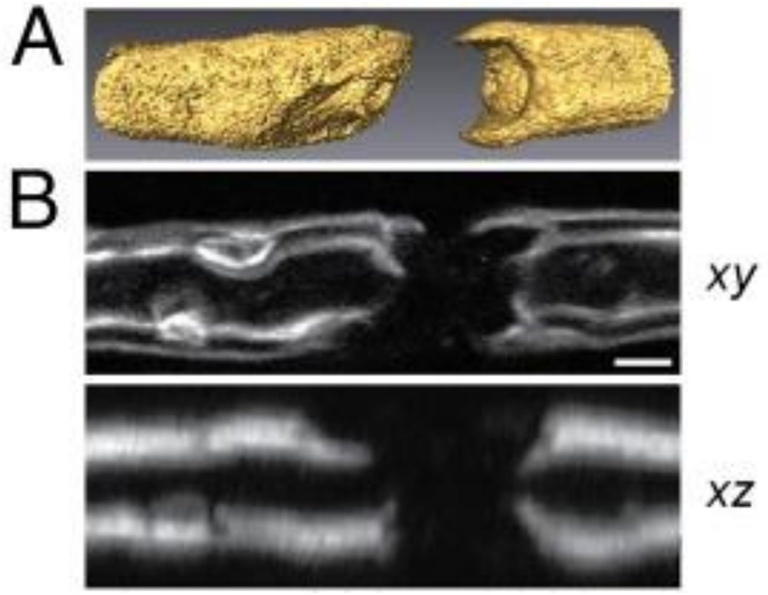


Figure 9 - Three-dimensional structure of domains in fresh teased adult rat sciatic nerve. (A) Volumetric rendering of nodal segment. (B) The lateral (xy) and longitudinal (xz) sections. (Scale bar, 3 µm.)

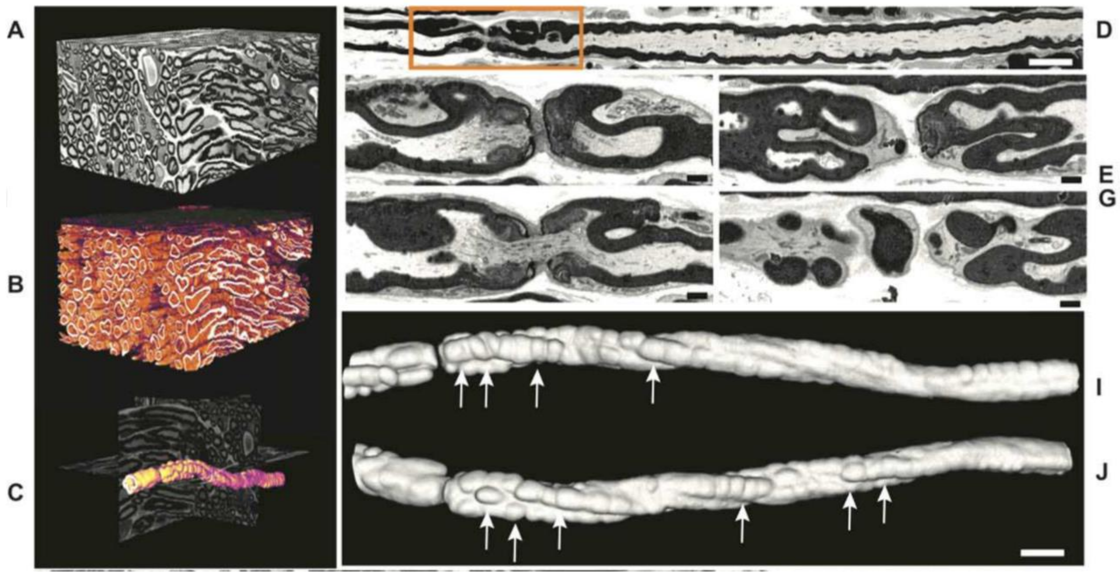


Figure 10 - Serial blockface scanning EM analysis of sciatic nerves shows aberrant myelination (arrows). (A–J) Three-dimensional reconstruction of image stack. Myelin foldings at nodes are elaborate as illustrated for a representative axon from this set (C, shown in orange on orthogonal slices). A single slice view (D, E–H from box in D); 3D view of the compact myelin (I, same orientation as D and J rotated through 180°). (Scale bars A–D: 5 µm; E–J: 1 µm).

5. Innovation and importance of the work

There are a lot of studies about connexins (Cx26/29/32/40…) in mice and rats, but no studies were performed in human samples: in particular, there are no data available about Cx32 localization in human Schwann cells. Alteration of Cx32 distribution can be related to the pathogenesis of CMT1X, as previously explained, given that the role of Cx32 in human neurolemma cannot be compensated by other connexins. Recent studies [34] suggest also that, together with GJs, a crucial role of Cx32 myelinating Schwann cells could be played by Cx32 hemichannels as they can release ATP controlling a purinergic signaling in the peripheral nervous system.

II. Materials and methods

1. Matlab rendering software

For a 3D rendering of the images taken with the microscope, I've written my own program in Matlab framework.

In the following the main StackTo3D '*s2tD*' program characteristics are reported; in appendix the whole code is documented.

2D modifications

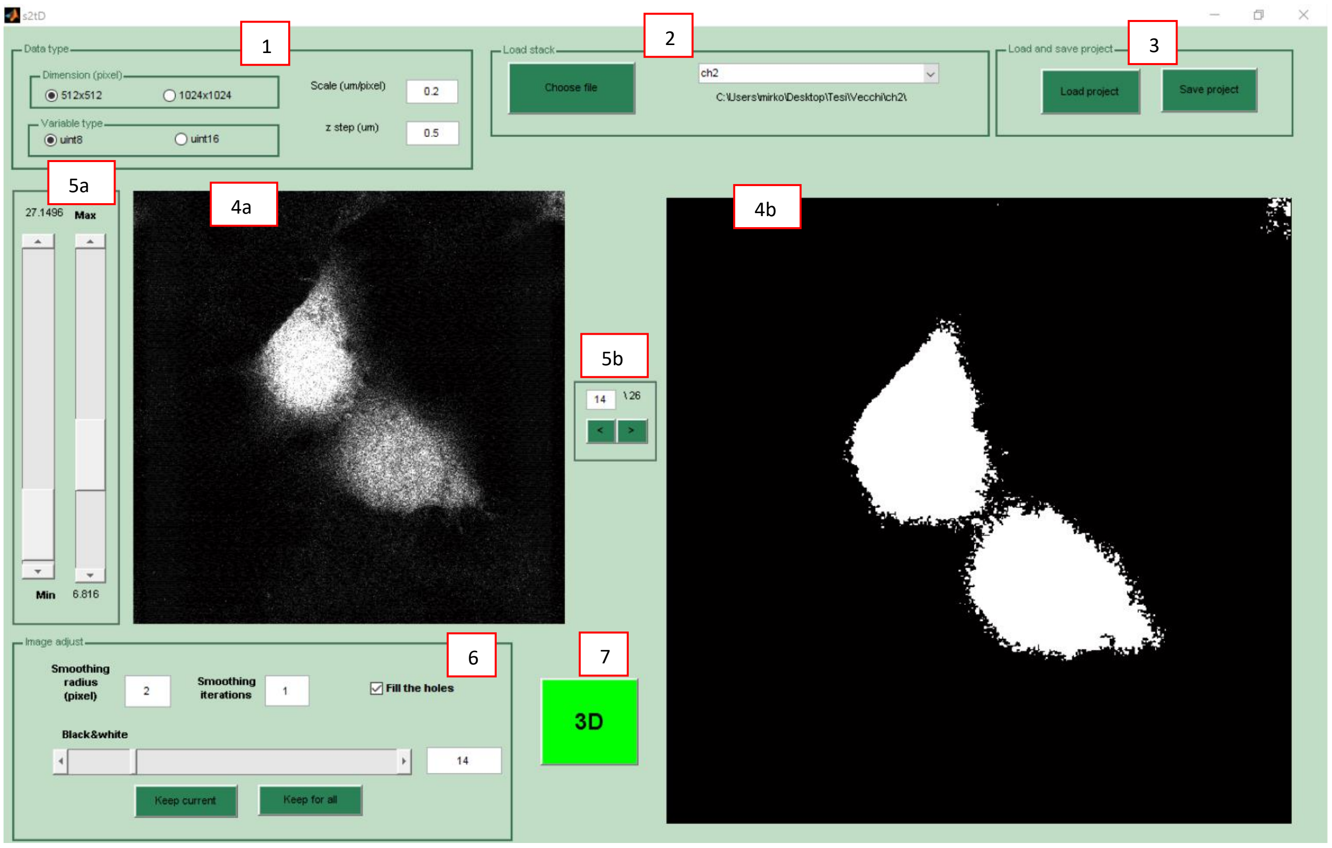

Dimension (pixel): [✓ 512x512] 1024x1024
Variable type: [✓ uint8] uint16
Scale (um/pixel): 0.2
z step (um): 0.5
ch2
C:\Users\mirko\Desktop\Tesi\Vecchi\ch2\
27.1496 Max
Min 6.816
14 \26
Smoothing radius (pixel): 2
Smoothing iterations: 1
[✓] Fill the holes
Black&white: 14

Figure 11 - Matlab 's2tD' program start window

1- In the first panel '*Data type*' the characteristics of the images that the user is going to render are selectable (pixel dimension and type of variable); furthermore, the scale (that is the length in micrometers of one pixel) and the measure (μm) of the steps are settable: with these fields the zthickness for the 3D-rendering is calculated, if they're left empty the rendering is performed in a scale 1:1:1 (that is, a z-step is considered as having 1 pixel length).

2- In the panel '*Load stack*' it is possible to choose the files of your z-stack: clicking on '*Choose file*' a window opens in which the files are selectable and savable in the work space giving a name at the channel; more than one channel (that is an identical series of the same z-stack investigated at different wavelength) can be loaded. The names of the loaded channels are selectable in the popup menu, under which the original path name is reported

3- The panel '*Load and save project*' on the right permits you to save the workspace for a future work and to load a matlab file you've previously saved.

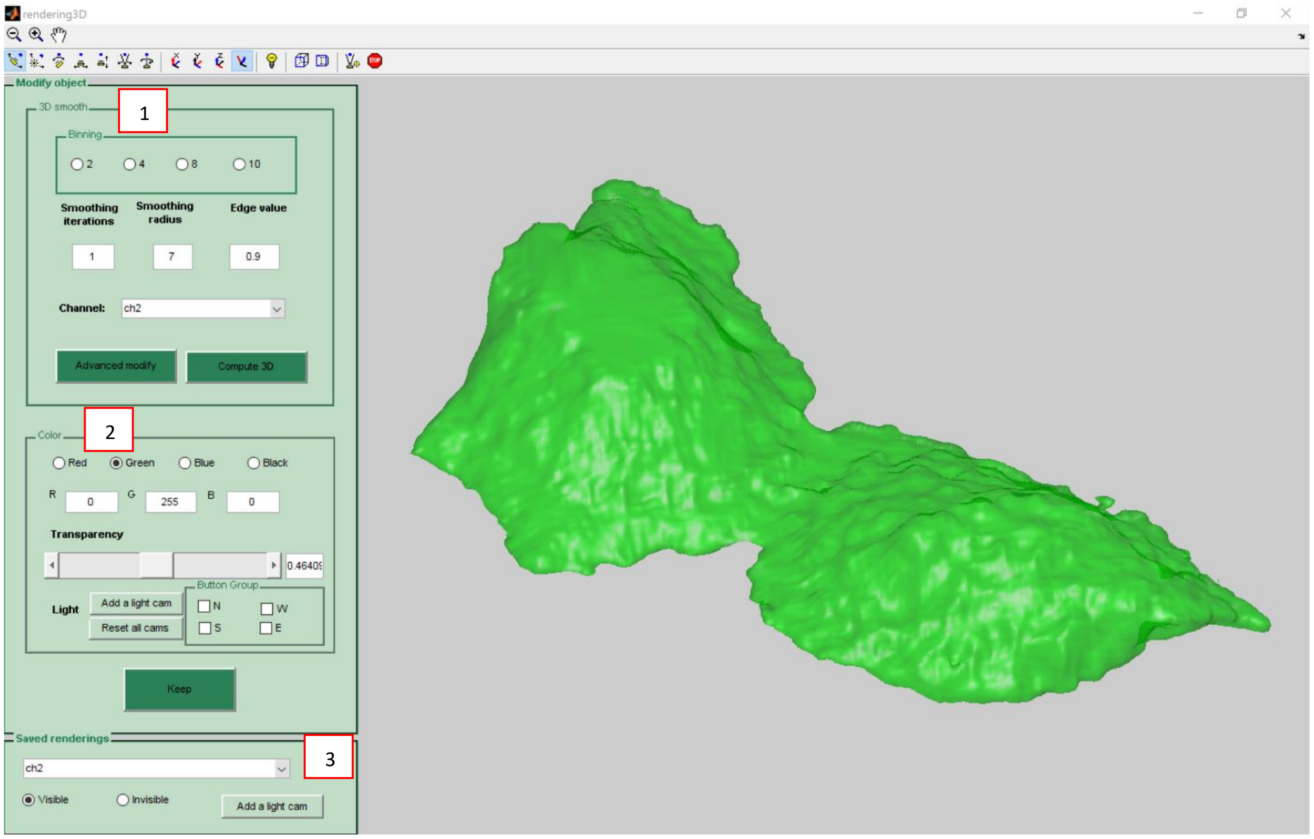
4- Loading a channel in the work space, it is possible to visualize the image of the first slice like raw figure on the left, and after a black-and-white conversion on the right.

5- With the bars on the left, the contrast in raw image visualization can be changed (regulating the maximum and minimum levels, that are visualized also as numbers representing the maximum value over which the points are considered white and the minimum under that all the points are black; the values are in a scale between 0-255 for uint8 precision, 0-65535 for unint16). In the middle there are the arrows (or the number of slice) to move through the z-slices of the selected channel.

6- The last panel '*Image adjust*' is the one on which modifications can be performed. First of all, a smoothing is selectable: in the first box one can choose the number of pixels taken into account to perform the mean smoothing, and in the second one the number of iterations of this smoothing (0 means no smoothing). There's the possibility to add the command *'Fill the holes*' that draw white pixel in small isolated black regions around white ones. Finally, with the black-and-white bar (or the corresponding value) it's possible to adjust the threshold of the conversion, that is visualized in the image panel on the right. To save these modifications on the current image (the visualized one) '*Keep current*' must be clicked, to keep the modifications also for all the other slices of the stack '*Keep for all*', otherwise no change is saved.

7- Once all the desired modifications are performed (on all the channels) one can open the new window for the 3D-rendering with the button '*3D*'.

3D rendering



Binning: ○ 2 ○ 4 ○ 8 ○ 10

Smoothing iterations: 1
Smoothing radius: 7
Edge value: 0.9
Channel: ch2

Color: ○ Red [✓ Green] ○ Blue ○ Black
R 0 G 255 B 0
Transparency 0.46409

Saved renderings: ch2
[✓ Visible] ○ Invisible

Figure 12 - Matlab 's2tD' program 3D window

1- In the first panel '*3D smooth*' all the parameters for the rendering are settable: first of all, a subpanel '*Binning*' permits to perform a reduction of the pixels (bringing a volume of nxnxn pixels and mediating it in a single pixel volume); in this way a smaller computational effort is needed to act on the image and rotate it. In the smoothing boxes the same operations like the ones described in 2D can be performed for a 3D smoothing (0 iterations means no 3D smoothing); '*Edge value*' indicates the iso-value, that means a surface linking all the points with that value is created to perform the 3D image. In the popup menu the channel to render is selectable; to perform the computation '*Compute 3D*' must be clicked and in the image box on the right the rendering is visualized. There's also the possibility to cut the 3D image or delete some elements that one doesn't want to visualize using the '*Advanced modify*' window, which provides a cutting tool.

2- In the second panel '*Color*' all the visual details are settable: the color can be chosen (selecting it or writing it in RGB notation) and the transparency can be fixed between 0 (invisible) and 1 (opaque). Finally, a '*light cam*' can be added, that is a point of light that render the image realistically, giving a real 3D effect; this cam is created in the point of view at which the image is oriented, at the same way one can select if create it in an upper position (Nord), lower (South), on the left (West) or on the right (East). All these cams can be deleted with '*Reset all cams'* button.

3- When the modifications are like desired the channel rendered can be kept in the final figure with the button '*Keep*': in this way the final figure is visualized, and the definitive channels are selectable in the last popup menu; for each channel one can select '*visible'* or '*invisible'*: if some channels are set to '*invisible'* they are not visualized in the final image. (It is possible to carry on other modifications to a channel in the final image, reselecting it on the first popup and performing a new rendering: while keeping it in the final image a message reporting the presence of a channel with the same name will be visualized, and it is possible to choose to overwrite the channel newly modified to the previous one).

Software output

Summarizing, my program is able to take a 2D z-stack (a series of images taken with steps along the optical axis), to make image filtering and to return a 3D image of the sample. The final purpose is to perform quantitative analyses of the distribution and size of Cx32 gap junction plaques, as well as to identify the location of Cx32 channels, still an open issue, how we've discussed.

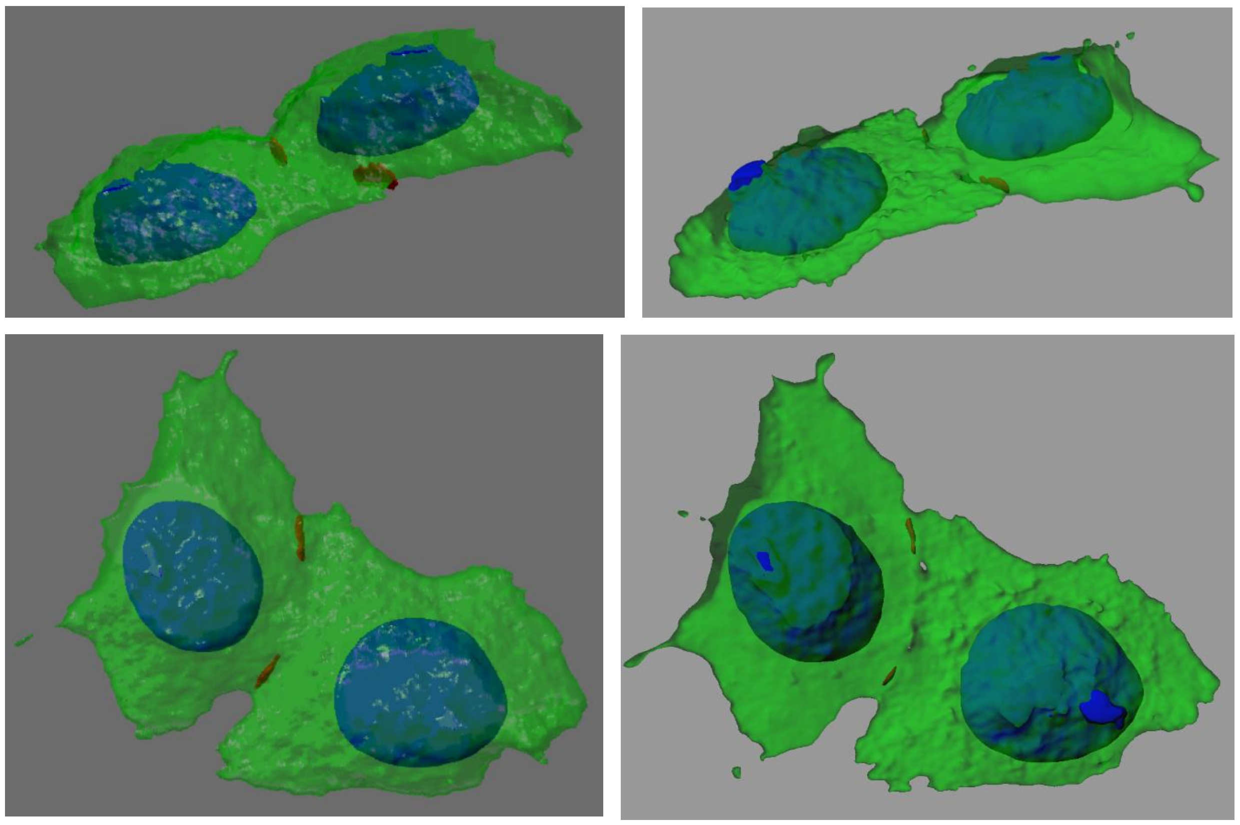


Figure 13 – HeLa cells rendered with Matlab (left) vs. Imaris (right)

In Figure 13 it's reported an explicative comparison between the results obtained with my Matlab program (on the left side) and the professional payment program Imaris (on the right): the images on which the rendering is performed are of HeLa cells; three channels are distinguishable, the nuclei (blue ones), the cytoplasm (in green) and the connexin plaques (in red). It can be noted that the final results are quite similar in terms of the rendered object.

2. Setup

In this chapter the setup used for the experiments is described, a home-made system based on a 2PE laser scanning microscope (Bergamo II System B232, Thorlabs), which is upgraded to perform two-beam multicolor multiphoton and STED microscopy [35].

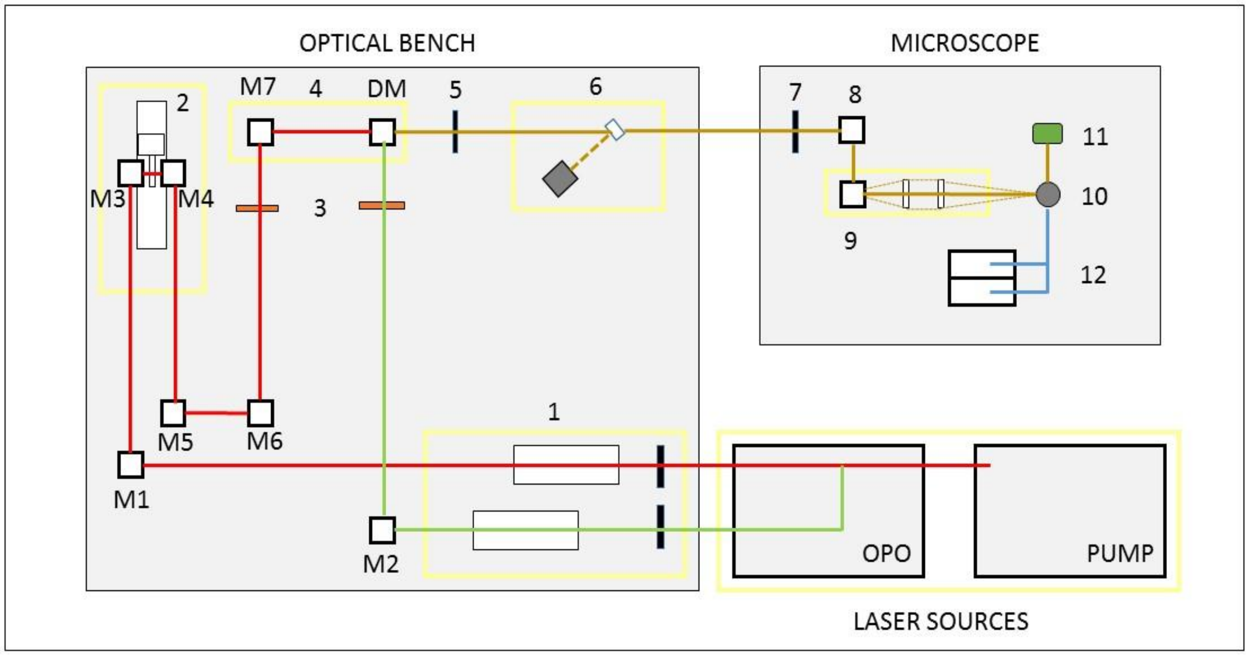


Figure 14 - Setup for 2PE configuration

OPTICAL BENCH

M=mirrors

1. Pockels cells: beams power modulation
2. Mobile delay line: temporal synchronization of the two pulses
3. Quarter-wave plates: circular polarization of light
4. Recombination mirrors (M7 and DM)
5. First diaphram: first reference for the alignment
6. Photodetection system: monitoring of the pulses at the oscilloscope
7. Second pinhole: second reference for the alignment

MICROSCOPE

8. Periscope (two mirrors)
9. Scan head (two galvo/resonance mirrors)
10. Objective
11. Sample site
12. Photomultipliers (PMT)

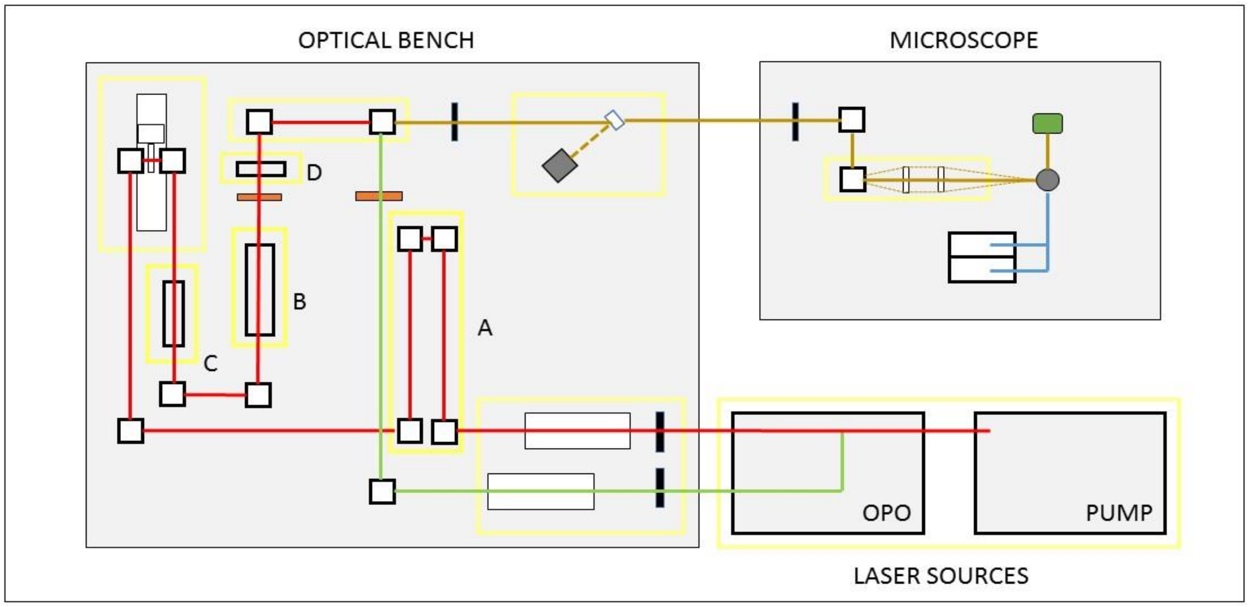


Figure 15 - Setup for STED configuration

ADJOINT STED ELEMENTS:

A. Fixed delay line: temporal synchronization of the two pulses
B. Optical fiber: temporal stretching of the pump pulse
C. Glass rod: protection for the single mode optical fiber
D. Spiral phase plate: modulation of the intensity distribution of the pump beam wavefront

To perform the presented tasks there's the necessity to work with two focused laser beams. A mode-locked Ti:Sapphire laser (Chameleon Ultra II, Coherent Inc.) with a wavelength that is tunable in the range between 680-1080nm and pulses of 140 fs at frequency of 80 MHz, can act like a pump for an Optical Parametric Oscillator (Chameleon Compact OPO, Coherent Inc.). This oscillator uses a fixed percentage of the main beam power to produce, (thanks to difference frequency generation), a second beam at a higher wavelength, that is tunable in the range 1000-1550 nm with pulse duration of 200 fs. The pulses start with about 6ns of relative delay and a diameter of about 1 mm, then they are forced to perform different paths in order to obtain the desired results (depending on the optical technique we want to perform) with different length, reaching the recombination mirrors with a temporal delay that can be modified by the mobile delay line on the OPO path.

With this basic setup, two possible configurations are available:

- *Femtosecond pulses*. The two beams are directly fed into the microscope, where they provide excitation light at half their own wavelengths. When the pulses are temporally superposed, and the foci of the beams on the sample are spatially superposed with sub-micrometric precision -both along the optical axis and on the focal plane- a third excitation channel opens up at an intermediate wavelength, so that it is possible to perform multiphoton multicolor microscopy.
- *STED nanoscopy*. The OPO beam is left unchanged and used for the sample excitation, the pump beam is used for the depletion of the excited molecules. A precise spatial superposition of the beams foci is still necessary, while the temporal delay between the pump and OPO pulses must be around 100 ps, in order to have the latter closely follow the former. To stretch the pump pulse width to hundreds of ps, the beam is focused on a single mode optical fiber (PM780-HP, Thorlabs, 100m length, NA 0.12, wavelength range 710-1100 nm). To avoid fiber damage, the beam travels along a 20cm-long glass rod (Hellma Optics, SF-7), which pre-stretches the pulse to about 1ps; at the end of the fiber the pulse is 10ps long. In order to apply STED depletion the intensity distribution of the pump beam is shaped as a doughnut by a spiral phase plate (VPP-1a, RPC Photonics, wavelength range 350-2000 nm), positioned immediately after the optical fiber output and before the recombination mirrors.

At the beginning of the path the PUMP beam has a power between 360-610 mW (depending on the wavelength), the OPO between 225-700 mW; to modulate the beams power, in order to work with lower values, there’s a system of two alimented Pockels cells that are placed at the exit of the laser source.

Pockels cell consists in a uniaxial crystal, with an ‘optic axis’ direction associated with an extraordinary refractive index, through which the phase velocity is the same for both planes of polarization. In this way a wave sees the crystal isotropic along the optic axis, but anisotropic in the orthogonal directions. Furthermore, there is an electro-optical effect that consists in a change of the refractive index caused by an external electric field: this can transform an isotropic material into an anisotropic one. If we expand the ordinary refractive index n with a Taylor series of the electrical field E:

$$n(E) = n + a_1 E + a_2 E^2 + \cdots$$

the linear contribution to n is known as Pockels effect, a variation of the indices values that is translated in a light phase velocity change: in this way, such a uniaxial crystal, can be used as a dynamic phase retarder (the effective Pockels cell).

A simple way to use these characteristics is to put a polarizer before the crystal to produce linearly polarized light, and another one to analyze the light at the exit of the crystal, that can have a polarization direction parallel or orthogonal to that of the first one (crossed polarizers configuration). In a longitudinal configuration (the external electric field is applied parallel to the optic z-axis along which the beam travels) without any voltage applied we obtain the maximum transmission for parallel polarizations, minimum for orthogonal ones; the transversal configuration (the field is applied in an orthogonal direction respect the optic axis) has an arbitrary operating point that can be compensated to minimum transmission in correspondence to zero AC voltage applying a DC voltage offset. At this point, when the AC voltage is increased across the crystal, there's a consequent increase in the transmitted light, until it reaches the maximum intensity, corresponding to a relative phase shift of 180° and a rotation of the polarization plane of 90°; in this way the linear polarizer in the exit produces a beam with an intensity that varies accordingly to the relative phase shift (δ): $I_{out} = I_{in} \sin^2 \pi\delta$.

The Pockels cells used in our apparatus are electro-optic modulators (350-80LA and 360-80LA, ConOptics Inc., respectively for pump and OPO beam) with a transversal design.

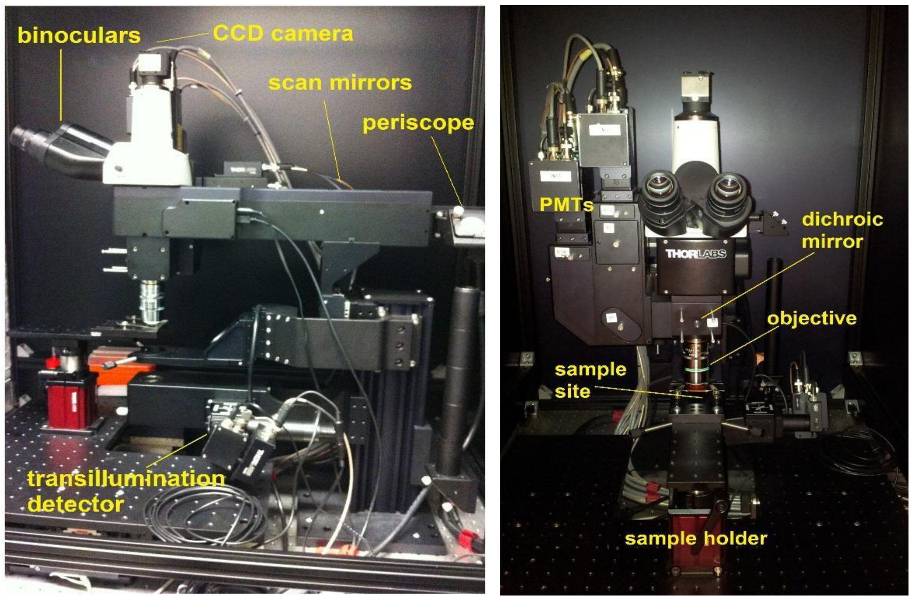


Figure 16 - Microscope (lateral and front view)

The first element in the path of the light inside the microscope setup is the periscope, a system of two mirrors that guides the light to the scan head. This last one consists in a galvo/resonance mirror set that rotate the beam on a pivot placed at the objective back focal plane at a 8 kHz rate. The result on the focal plane of the objective is a linear raster scan of the focal point. The microscope works in epifluorescence configuration, that is the objective is the condenser also: first it focuses the light onto the sample, then it collects the emitted light deriving from the sample itself; the excitation and detection paths are separated by a longpass 705 nm dichroic mirror, which transmits the illumination light onto the objective and reflects the emitted light towards the detection module. Detection is performed by means of four photomultipliers (4X GaAsP PMT, Hamamatsu H12310) coupled to four different band-pass filters:

- Channel A: (395 ± 25) nm [UV]
- Channel B: (460 ± 50) nm [Blue]
- Channel C: (525 ± 40) nm [Green]
- Channel D: (625 ± 90) nm [Red]

Then the emission light is directed to the correct detection channel by three shortpass dichroic mirrors reflecting light at 425 nm, 495 nm, 565 nm.

There's also the possibility to observe in transmission: the light is collected by the condenser below the sample and after that is sent to an INGaAs detector (DET08CL, wavelength 800-1800 nm).

3. STED improvement

A big part of the work consists in the effort of implementing the STED technique in such a way to be able to obtain super-resolute cell images. The problem is that the home-made apparatus is not ready to be used with this technique and it has to be improved: particularly the configuration appears to be good, but no STED results are obtained, that means improvements in resolution are not appreciable. The work consists in a physical study of the system to understand and discover the underlined problems and resolve them.

Spatial alignment

The procedure basically consists in the careful positioning of the mirrors required for beam stirring on the optical bench. The PUMP and OPO beams start their travel from two ports on the OPO, where they exit parallel to each other and at the same height over the optical bench. Thereafter, the beams encounter different optical elements before being recombined. The recombined beams must enter the microscope periscope at the same point and with the same angle so that the objective back aperture can be equally illuminated by both. In addition, under suitable achromaticity conditions, the two beams should be focused by the objective onto the same spot.

To perform this task there are two main steps: first of all, a precise alignment in the plane perpendicular to axial direction (x-y plane) is needed, and this is performed maximizing the power (calculated with a power meter) at the entrance of the periscope, regulating the last two mirrors in the optical path of both beams. The technique consists in alternating the adjustment of the two mirrors with the help of pinholes placed after these mirrors in order to maximize the laser power in a repetitive procedure: with the first mirror the x-y position is changeable, with the second one the angle is adjusted. Furthermore the user has the possibility to correct the alignment without modifying anything in the setup manually, but exploiting the piezoelectric systems available in the recombination mirrors holders. In fact, by applying separately in x, y or z axes an appropriate voltage (between 0 and 150 Volts) to piezo controllers via software (MDT693A Piezo Controller), the PUMP or the OPO beam respectively move. The correction can be applied observing quantum dots on a slide at maximum magnification (32x). The QDs size under the optical resolution limit makes them behave as point sources. A single QD is selected and imaged by both beams; acting on piezoelectric voltages, the perfect alignment is achieved when the two images overlap in (x, y) plane. The procedure is illustrated in Figure 17: the QD is imaged through the PUMP beam and centered in a ROI; the illumination is switched into the OPO beam, visualizing the same ROI. The OPO beam alignment is corrected by mean of piezoelectric control until the two QD images overlap.

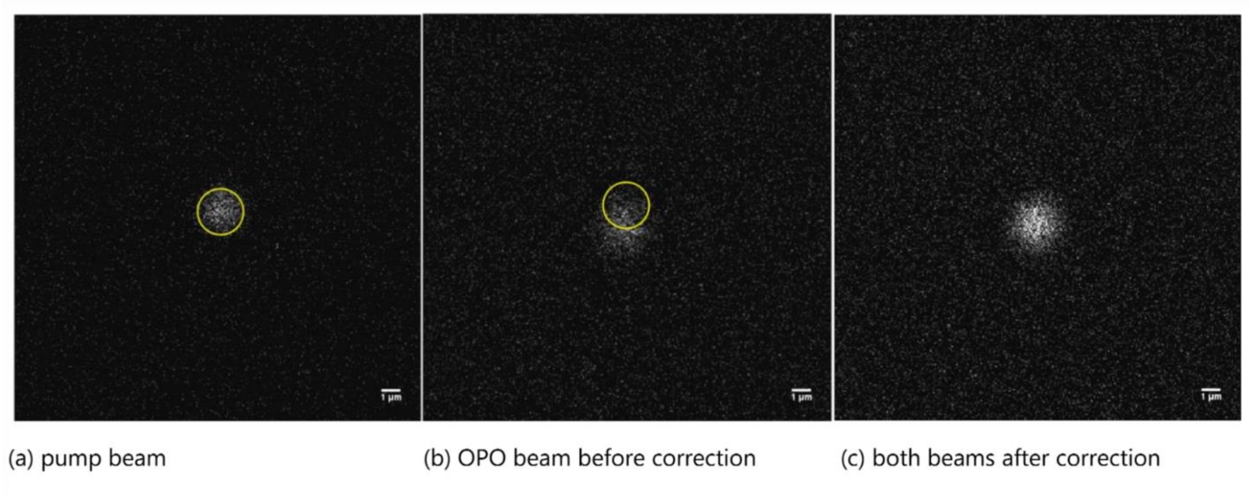

1 µm
1 µm
1 µm
(a) pump beam
(b) OPO beam before correction
(c) both beams after correction

Figure 17 - Alignment correction through piezoelectric control at 32x magnification observing QD PSF

The second step is the z-alignment (along the optical axes) and, to perform this, the introduction of telescopes is necessary: it was tested that introducing a telescope in the OPO beam path at the best a separation of 2,4µm is reachable.

Only with the introduction of a second telescope in the PUMP beam a $\Delta z \approx 0 \mu m$ is reachable. Obviously it is to consider that the telescopes influence the dimensions of the beams at the same time.

Stretch of the pulse and temporal delay

If the source of the PUMP beam is a laser with ultrashort pulses (~100 fs), as is the case for lasers usually used also for multi-photon microscopy, in order to achieve the desired pulse width conditions for an optimal fluorescence depletion is necessary to extend the duration of its pulses without reducing its pulse energy. The pulses width is increased by using laser pulse stretchers, which stretch the pulses temporally. An ideal stretcher increases the duration of the laser pulse without introducing losses, so the peak power is reduced but its average power remains constant. The peak power reduction can also be exploited to avoid damaging the optical fibers following the injection of high-energy lasers. According to the type of the laser and the requirements of the application, various types of laser pulse stretchers, both passive and active, are developed for temporal stretching of laser pulses. In the case of ultrashort pulses, the pulse stretching is achieved by dispersion, which is a suitable technique due to the very large spectral bandwidth that characterize this kind of pulses. The dispersive pulse stretcher is a configuration of dispersive optical elements that introduce wavelength dependent optical delays to induce a temporal arrangement of the frequency components of the laser pulse (frequency chirp). Common examples of dispersive stretchers are optical fibers and diffraction gratings.

Another important characteristic is the temporal delay: to perform STED, the depletion pulse has to arrive just after the stimulation OPO beam. In this setup the delay is adjustable thanks to a guide line that changes the length of OPO path; to understand the best position of this guide line to perform STED, another effect that occurs only with temporal coherency was tested: sum frequency generation. The signal given by OPO and PUMP SHG separately can sum to give rise to a new signal, that however can occur

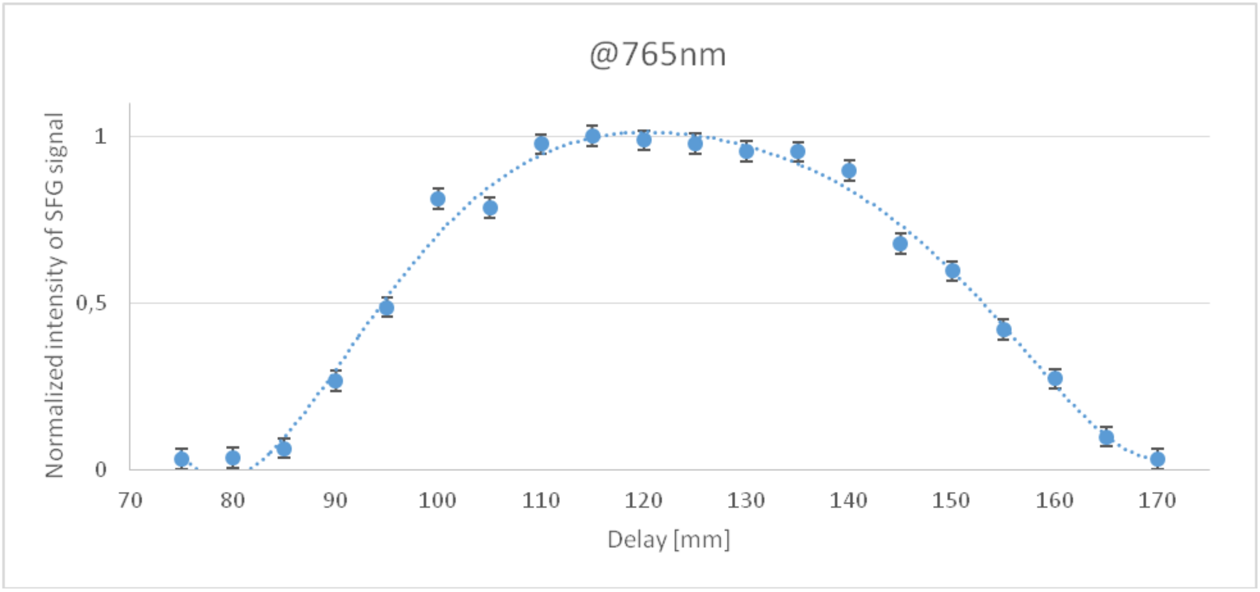


Figure 18 - SFG signal varying the delay line at 765nm of PUMP beam

only if the two beams are time aligned; more in detail, we are able to see a maximum signal of SFG only when the two beams are temporally aligned, then the signal intensity decays to zero. So, with reference to Figure 18, the plateau of maximum intensity is the region of time overlap, while in the ascendant region the OPO path is shorter, that means the OPO signal arrives before PUMP (and similarly in the descendent region it arrives after PUMP): it is clear that the region of interest for STED technique is the ascendant one, so the position of the guide line at half ascendant maximum was calculated for different wavelength; in this way one has the best position of the delay line for different wavelength.

From this experiment one can also calculate the temporal width of the PUMP pulse, looking at the FWHM. Considering that the light covers 30 centimeters every nanosecond, with these results one can estimate in a row way:

$$\frac{(11.6 \pm 0.1)cm}{\frac{(30 \pm 1)cm}{ns}} = (0.39 \pm 0.01)ns = (390 \pm 10)ps$$

OPO beam instead is seen like pointlike, because its width is under our precision possibility (in the order of hundred fs, that means tens of µm).

PSF FWHM

The profile obtained for the two beams are reported in Figure 19. How it is visible, with the reached conditions the FWHM of the Gaussian OPO beam equals the FWHM of the donut hole.

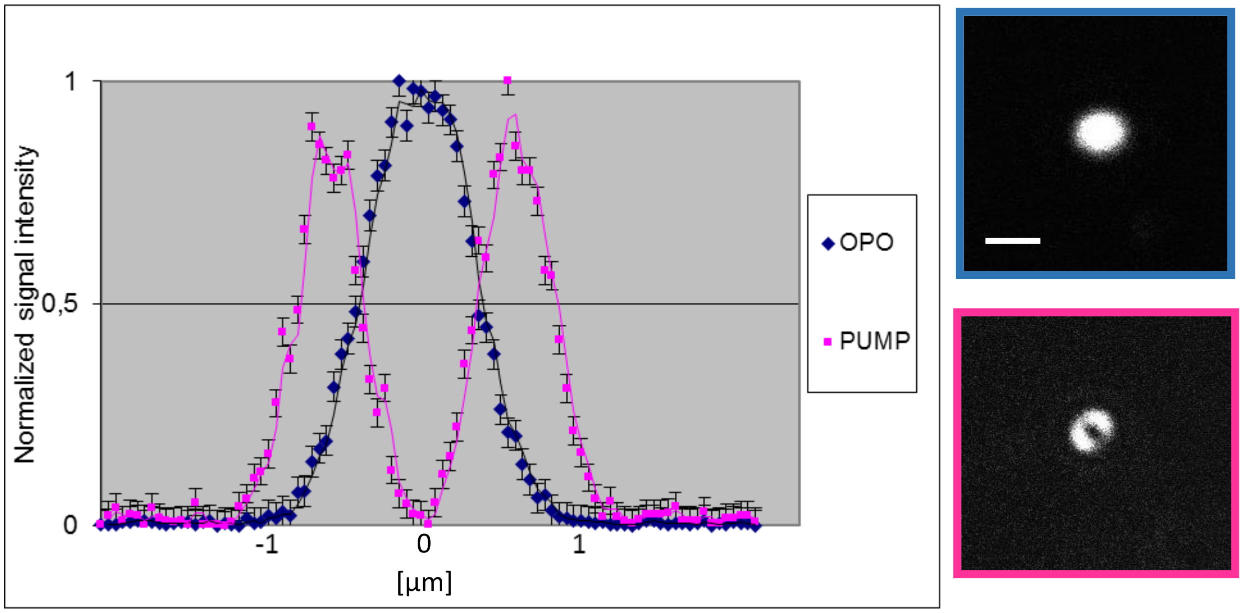


Figure 19 - OPO (blue) and STED (purple) beam profiles and PSFs on the focal plane (scale bar 2µm)

The important thing is that OPO beam tails lay inside the donut ones, in such a way the outer region of the OPO PSF is all depleted; another characteristic that can be convenient is the hole FWHM to be the smallest as possible.

Critical points

There are three points on which one should work to improve the STED technique with our apparatus:

- Reduce the FWHM of the hole PUMP PSF
- Raise the power of the PUMP laser beam at least around 180mW at the back focal plane
- Consider PUMP beam background influencing STED effect

Unfortunately, the first two points are no more improvable (unless big changes in the setup were performed) and the configuration is already the best possible for our components.

The background is on the other side a bigger problem, because it’s intrinsic and not eliminable at the wavelength we work with in this setup; furthermore, there’s the problem that PUMP pulse doesn’t excite only the 2PE tail, but it was also discovered that at certain wavelengths (that we use in STED technique with our apparatus and are around 780nm) it excites in a more significant way also at 1PE (exciting at an upper energy). This excitement creates a lot of background light that doesn’t permit to see STED effect. What is interesting is that, theoretically, this two events (1P background and STED) are statistically independent, that means one can subtract them: so, to obtain STED effect, it is of primary importance to perform an offline analysis, subtracting the PUMP beam background to the combined STED signal. A possible solution to reduce this intrinsic excitation is to work at upper wavelengths (more than 800nm) where the 1P effect is less probably; but all these solutions don’t give the secureness to obtain good results.

More in details, the point is that the recent introduction of highly photostable dyes in the red and far-red emission range, such as ATTO 647N, STAR 635P, silicon rhodamine (SiR), ATTO 594, and ATTO 590, offers new opportunities for 2PE microscopy; also, they are well suited for STED nanoscopy with its popular depletion wavelength of 775 nm. The problem is that these dyes have already been successfully used with 1PE-STED nanoscopy, however, because their 2PA characteristics have not yet been quantified, they have been of limited use in 2PE-STED nanoscopy (and 2PE microscopy in general), that’s our case[36].

4. Biological methods

In the follow the protocol used for the preparation of the sciatic nerve from a mouse is reported.

Nerve dissection

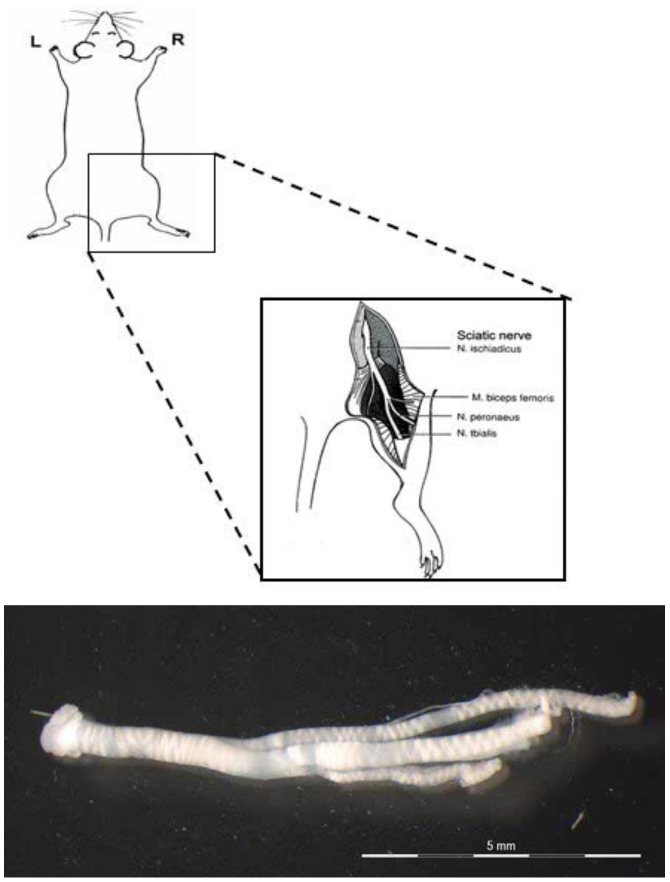


Figure 20 - Mouse sciatic nerve (in PBS on the bottom)

Fix the paws on the surface and wet the rear of the mouse to facilitate the cut; with the tongs, lift the rear skin and, with the scissors, after having cleaned it from the hair, uncover the lumbar muscles and that of the lower limbs. With thin scissors position yourself below the hip bone and hole the mouse, opening with attention the scissors: in this way the sciatic nerve is uncovered without damages. Free the area around the nerve from the muscle (paying attention not to cut the blood vessels), and make a sharp cut in the proximal part of the nerve (P in Figure 20). With the tweezers keep the nerve always only in the proximal part (not to damage it in other regions) and unstick it from the connective tissue very gently. At the end cut the nerve in the distal part (D in Figure 20) in a sharply way; finally it would be better to perform an oblique cut in the proximal part to distinguish it from the distal one. Put the nerve in an Eppendorf with Glutaraldehyde 2.5% or Paraformaldehyde (PFA) 4% for 10 minutes to fix it. After fixation, remove PFA and immerse nerves in fresh cold PBS; store sciatic nerves in PBS at 4ºC overnight (they may be stored in these conditions for up to a few days before beginning to tease them).

Nerve teasing

Transfer fixed nerves to a 35 or 60mm diameter Petri dish less than half full with PBS. Remove perineurial sheath by holding the proximal end of the nerve segment with forceps and gently stripping away the sheath with a fine needle or a second set of forceps; then subdivide the nerve segment into several fascicles with the fine needle (this requires lots of patience). Place a couple drops of PBS on a slide and transfer the bundle of nerves to the slide; still holding the proximal end of the nerve fibers with forceps, gently drag them across the slide. At the same time, use the long fine needle to 'brush' them out on the slide, reaching as much separation between the individual nerves fibers as possible (this requires lots of patience too). Allow the slides to air dry overnight and then store slides at –80ºC or stain them immediately as follows.

Nerve staining

Permeabilize slides with pre-chilled acetone for 5-10 minutes (or methanol for 10 minutes) at -20ºC; wash with fresh 1X PBS 2x times, 5 minutes/wash, at RT; wash for 10 minutes with BSA 1%. After that, use a blocking solution for 1 hour at RT: 2% BSA, 0.1% fish skin gelatin, 0.5% Triton X-100 (to permeabilize), 5% normal goat serum (to reduce unspecific stain), DPBS 1X. Dilute primary antibody in blocking solution prior to adding to slides. (For TSN, typically it requires only ~75 µL of primary antibody to cover the TSN).

Incubate in primary antibody overnight at 4ºC in a humidifying chamber. Wash 2x with PBS 5 minutes/wash (For TSN: 3x with PBS: 10min, 10min, 5min). Prepare secondary immediately before use and incubate slides in secondary antibody for 1 hour at RT, in the dark; wash 2x with PBS 5 minutes/wash (for TSN: 4X with PBS, 5 minutes/wash plus a wash with dH_20 for 5min).

At the end mount with Citifluor (+Hoescht) (for TSN use ~30 µL of Citifluor for each slide, placing a drop directly on the teased fibers), apply cover-slip without creating any air bubbles, and seal the edges with clear nail polish.

Connexin32 staining:

Postfixes slides with pre-chilled acetone at -20ºC for 10 minutes and wash with fresh 1X PBS 2x times, 5 minutes/wash, at RT; wash for 10 minutes with BSA 1%. For the blocking phase use a block solution for

1 hour at RT. Dilute primary antibody in blocking solution and incubate for 24-48h at 4ºC in a humidifying chamber. Wash 3x with PBS: 10min, 10min, 5min and incubate slides in secondary antibody for 1 hour at RT, in the dark.

Wash 4X with PBS, 5 minutes/wash and 1x with dH_20 for 5min. Dry slides thoroughly.

At the end, mount with Fluosave. Use ~30 µL of mounting medium for each slide, placing a drop directly on the teased fibers; apply cover-slip without creating any air bubbles, and seal the edges with clear nail polish and store at 4°C for 12-24 hour before use.

Choose of the dyes

The main experiments of this work consist in taking images with fluorescent 2PE microcopy to visualize different domains of a Schwann cell; the possibilities are to mark nuclei, myelin, cytoplasm, axon, neurofilament, Cx32. Clearly it's not possible to visualize all of them at the same time and a choose has to be done; that's because only few wavelengths are recordable (maximum four different, each for every channel of the apparatus) and so the right dyes must be chosen in order to obtain compatible emission spectra in the four admittance spectral region of the PMT (colored squared regions in Figure 21), considering at the same time that these dyes must present excitation spectra at half the wavelength we can use with our PUMP pulse (that is around 800nm).

Nuclei are one of the thing that we decide to mark, and in order to obtain this we used DAPI that links itself with them and emits in the blue channel.

Another fundamental element that can't miss in our work is Cx32, and to mark this the anti-Cx32 with, for example, Alexa647 (or the more stable Atto647) is used; this emits in the red channel.

The main structure of the Schwann cell can be visualized through the myelin; the interesting thing is that the surface change, passing from the outside to the myelin and from this to the axon can be observed with THG signal; that means that one can visualize the myelin sheath by THG, without the use of any dye. THG emits in the gray channel thanks to an excitation at 1175nm (with OPO pump).

Finally, the green channel remains available; one chose can be the use of P100 antibody, (conjugated with an Alexa or Qdot 525 in order to visualize it in the right channel), that permits to imaging the cytoplasm and thus the whole skeleton of the cell.

In

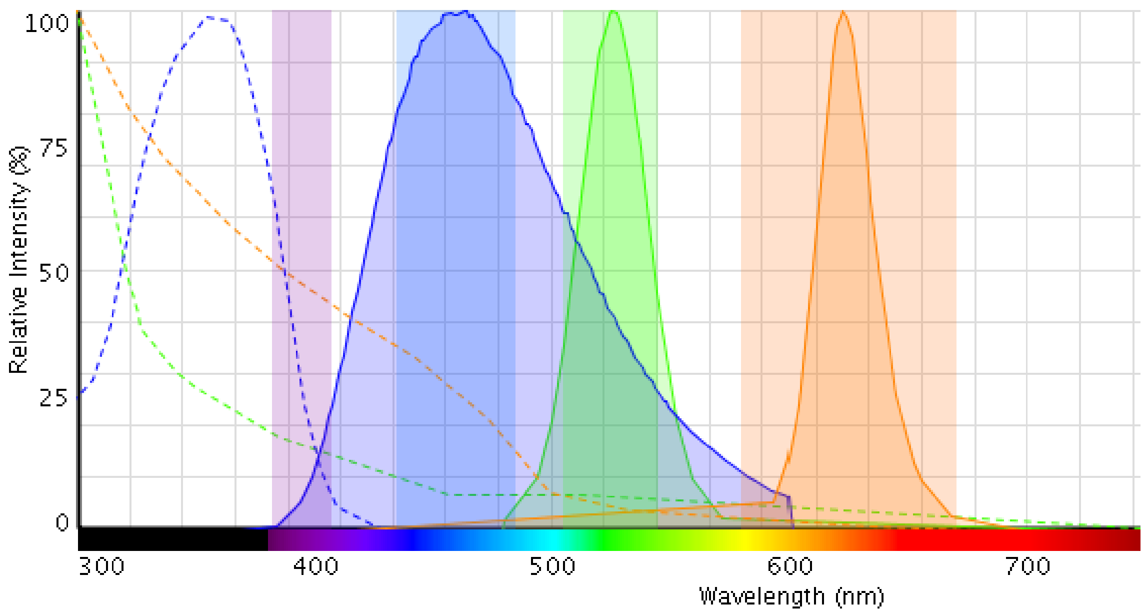


Figure 21 - Absorption (dotted line) and emission (continue line) spectra for DAPI, Qdot 525 and Qdot 625; acceptance interval of PMT (colored squared regions)

Figure 21 we can see an example of an ideal case in which the dyes are chosen in the optimal way to emit in the acceptance regions of our PMT (without overlapping to much) and are all excitable by 2PE with a laser wavelength of about 780nm (that means they're excited at about 390nm). The first region correspondent to channel A is used to collect THG signal, in the other three regions (channels B,C,D) the maxima of emission spectra respectively of DAPI, Quantum Dot 525 and Quantum Dot 625 occur; in this way we can imaging nuclei on channel B (blue) and two other cell domains in channels C and D conjugating the two dyes with appropriate antibodies. In the previous example the only problem that can occur is that one can obtain a consistent unwanted signal of DAPI fluorescence also in channel C (green) due to the big DAPI tail, imaging in this way two different cell domains in this channel.

III. Results

1. Rendering programs selection

For a 3D rendering of the images taken with the microscope, different software can be exploited: the problem is that much of these programs don't perform a real rendering but only a 3D view (putting all near together the slices), like for example ImageJ; other, like Vaa3D or Chimera, could be a good choice to perform surface generation but don't have a computational flexibility to perform a whole set of different measurements on the object created (for example because more oriented and specialized to molecular analysis only). Obviously, good software can be found on payment (an example is Imaris), even if they're quite expensive.

Program	***3D rendering***	***3D visualization speed***	***Customizability***	***Cost***
VolViewer	low	medium	low	high
Voxx2.1	low	medium	low	high
Vaa3d	medium	high	low	high
Chimera	high	high	low	high
Imaris	high	high	medium	low
Matlab	high	low	high	medium

Table 1 - Software comparison

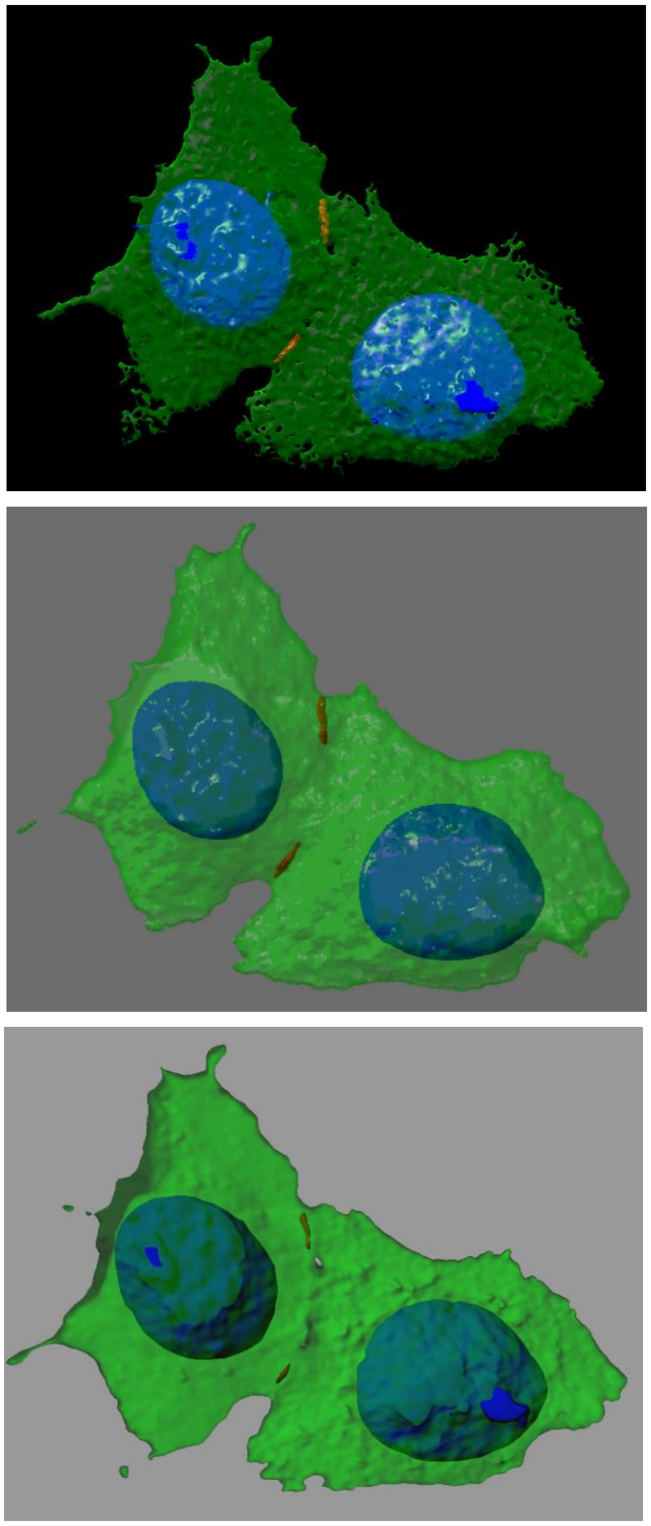


Figure 22 - Chimera (up) vs. Matlab (center) vs. Imaris (down) 3D HeLa rendering

In Table 1 a personal comparison of the main software analyzed is proposed: an evaluation (low, medium, high performance) is reported and the features that are enlightened are the general 3D-rendering of the images (regarding quality of the rendering, possibility of acting on the details of the rendering, easiness on performing it), the visualization speed (that is the speed in rotating the image), the customizability of the software (important feature to be able to improve better the program itself, for example in the rendering, but mainly to add capability for quantitative measures and personalized analysis), and the cost of the program (referring with 'high' performance for a low cost software).

How it is visible, some programs don't have such an optimal 3D-rendering capability for our desired standard, because it's not so easy and direct to create the surfaces you desire and also because the quality is not much improvable; furthermore, VolViewer and Voxx are not particularly customizable, and only a few restricted analysis capabilities are possible. Vaa3d and Chimera are quite good (in particular the second one has a very good rendering result) and a fast rotation, but the customizability remains restricted; Imaris is an excellent and fast software, with the problem that it's a payment one, and the price is very expensive. Matlab needs instead a different treatise: it's a programming code, and with a Matlab license, a lot of different performances are possible; it has the capability to create an user interface of your program, and with some (quite long) work and implementation it is possible to construct an own rendering software. What I was able to obtain it’s a quite good 3d-image similar to that of other programs, the main differences with other software consisting, instead, in the capability of 3D-image manipulation (the slowness in visualization being a characteristic of Matlab[37]): for example Imaris is more fluid and faster, clearly because professional and optimized, and to reach such a lightness in the image rotation I’ve introduced a binning, that is the possibility to reduce the pixels of the image in order to perform a minor computational effort; this leads to a less detailed image (but clearly more smooth too), that can be restored in all its definition at the end of the work. Apart from that, with my own program I’ve the full control of the image, and I can perform modification but also personalized quantitative measures on the volumes rendered (adjustable and implementable in every moment). These are the reasons that lead me, after the comparison of these programs, to choose to use a self-made software in Matlab environment.

In Figure 22 the best renderings obtained with the three different software evaluated with ‘high’ rendering performance are reported.

2. Experimental results

Preliminary STED results

Depletion experiments

To test our apparatus for STED technique, depletion experiments were conducted, consisting in a measurement of the intensity degradation applying gaussian PUMP beam (not shaped) on the OPO one. Obviously, at the best (if the PUMP beam covers all the OPO one), a depletion of 100% is expected, even if it is to consider that this depends on many factors like wavelength, sample tested, intensity of the beams.

The best result was obtained with a sample of Alexa 647 with PUMP at 795nm and OPO at 1200nm, both at maximum intensity, reaching a depletion of 65%. This is only a row calculation, and with a subtraction of the background (that is the background of the image itself and the background given by PUMP laser only) a depletion of 80% can be obtained.

STED experiments

In the following Figure 23 the best results of STED images that we were able to obtain are reported. The tests were conducted on Mitochondrial TOMM20 staining by Alexa 647 in HeLa cells.

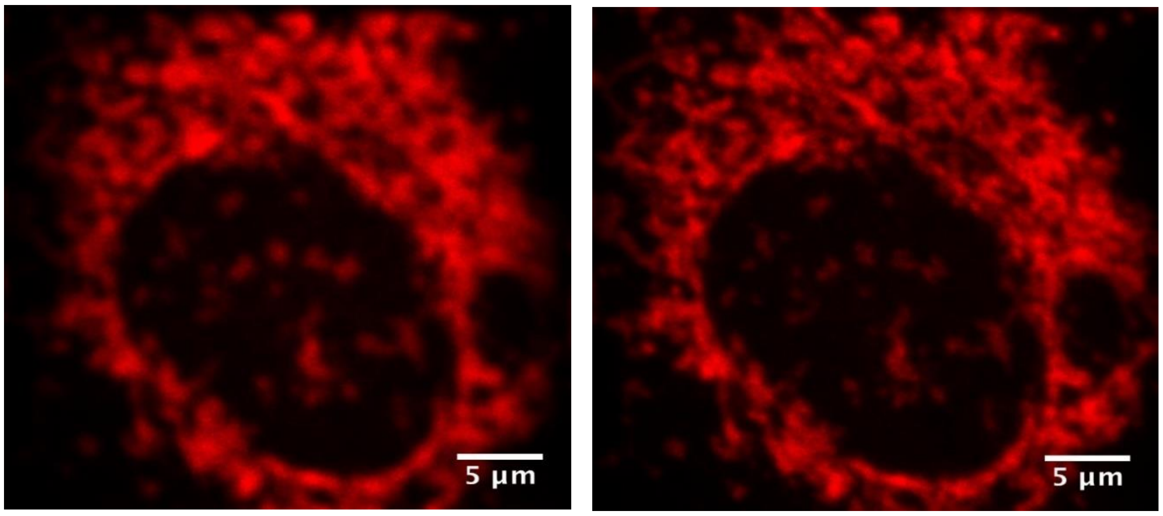


Figure 23 - Comparison between 2PE imaging (on the left) and STED imaging (on the right) on Mitochondrial TOMM20

How it's visible there's no such an improvement on resolution with respect to 2PE, accordingly to the low results in depletion that we have obtained previously.

Immunofluorescence and THG imaging results

In the following some of the most interesting images that have been obtained are reported.

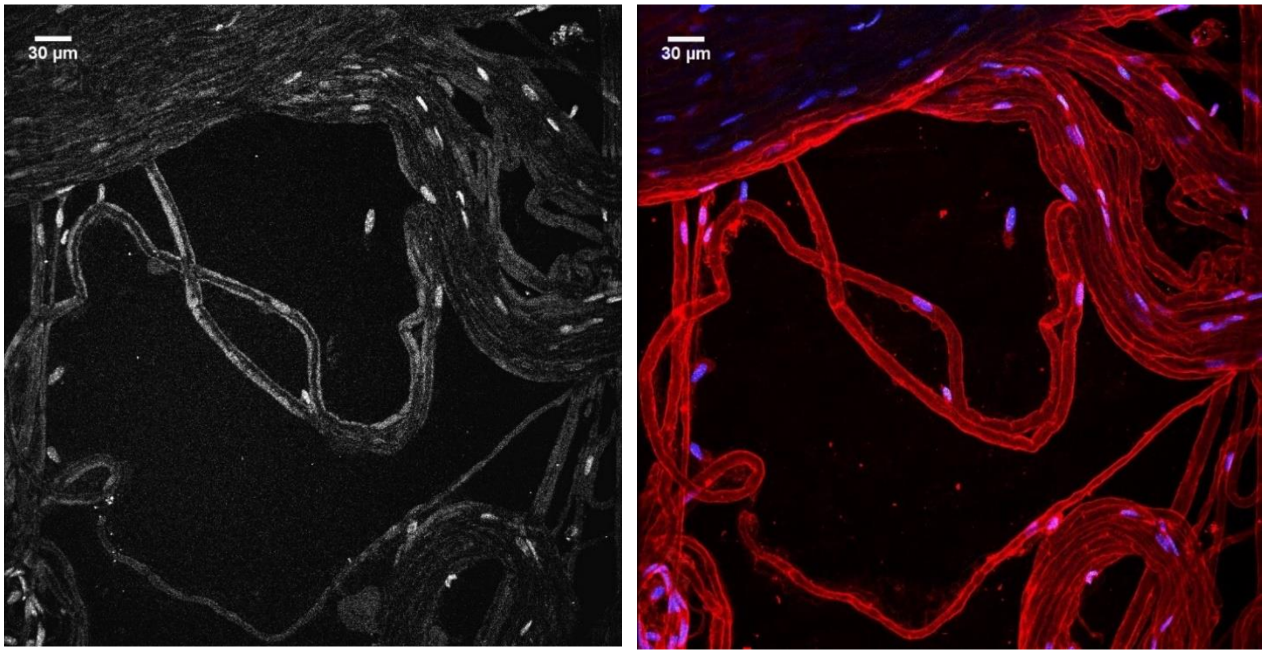


Figure 24 - Mouse sciatic nerve. THG (left), MBP+ DAPI (right).

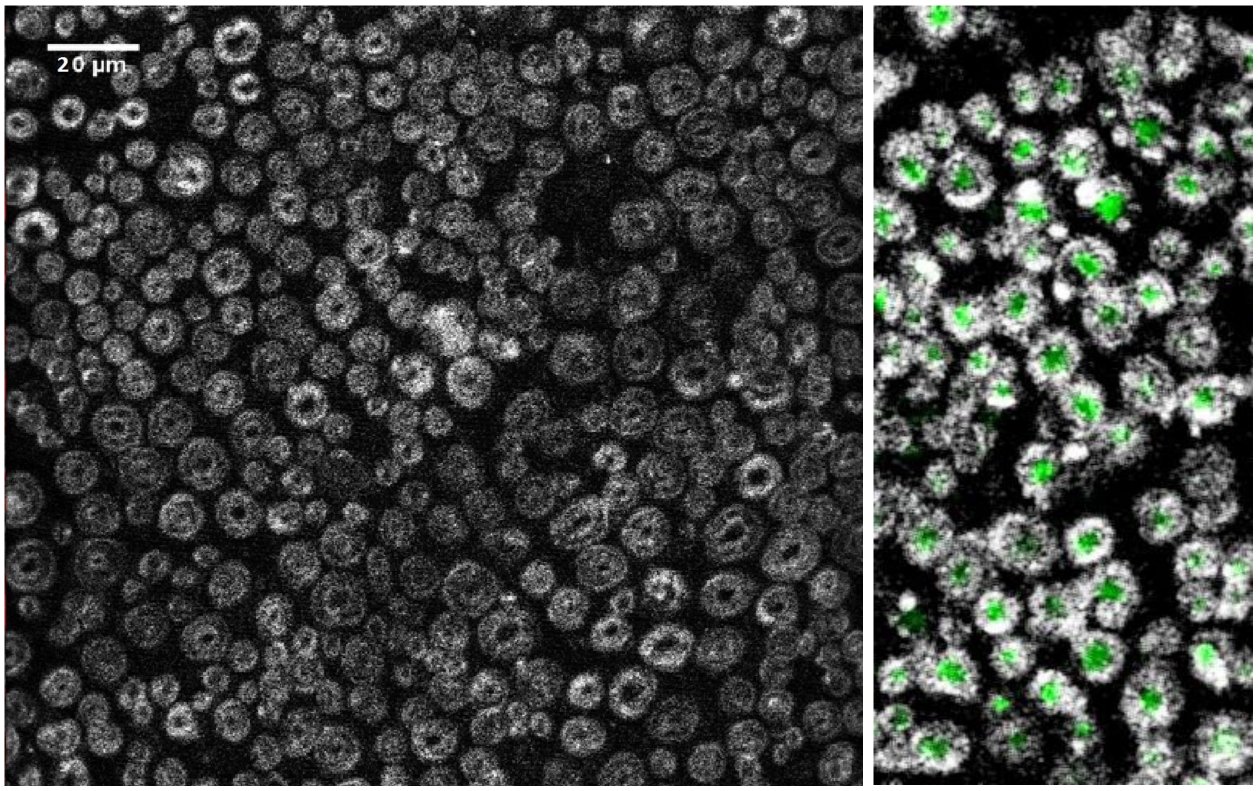


Figure 25 - Mouse sciatic nerve radial cut. THG signal (left) and merge with neurofilament (right).

In Figure 24 a sample of mouse sciatic nerve is shown. On the left image the THG signal is visible, on the right the myelin marked by MBP in red and the presence of nuclei (DAPI in blue) are enlightened by 2PE imaging. It is clear how the THG signal is colocalized with myelin MBP signal.

In Figure 25 a radial cut of the nerve is imaged by THG (on the left); this is another example of different preparation of our sample with a radial cut, in which the myelin that wraps around the axon is clearly visible. On the right there’s the green channel too, in which neurofilament signal is recorded; this shows the axonal region of Schwann cells.

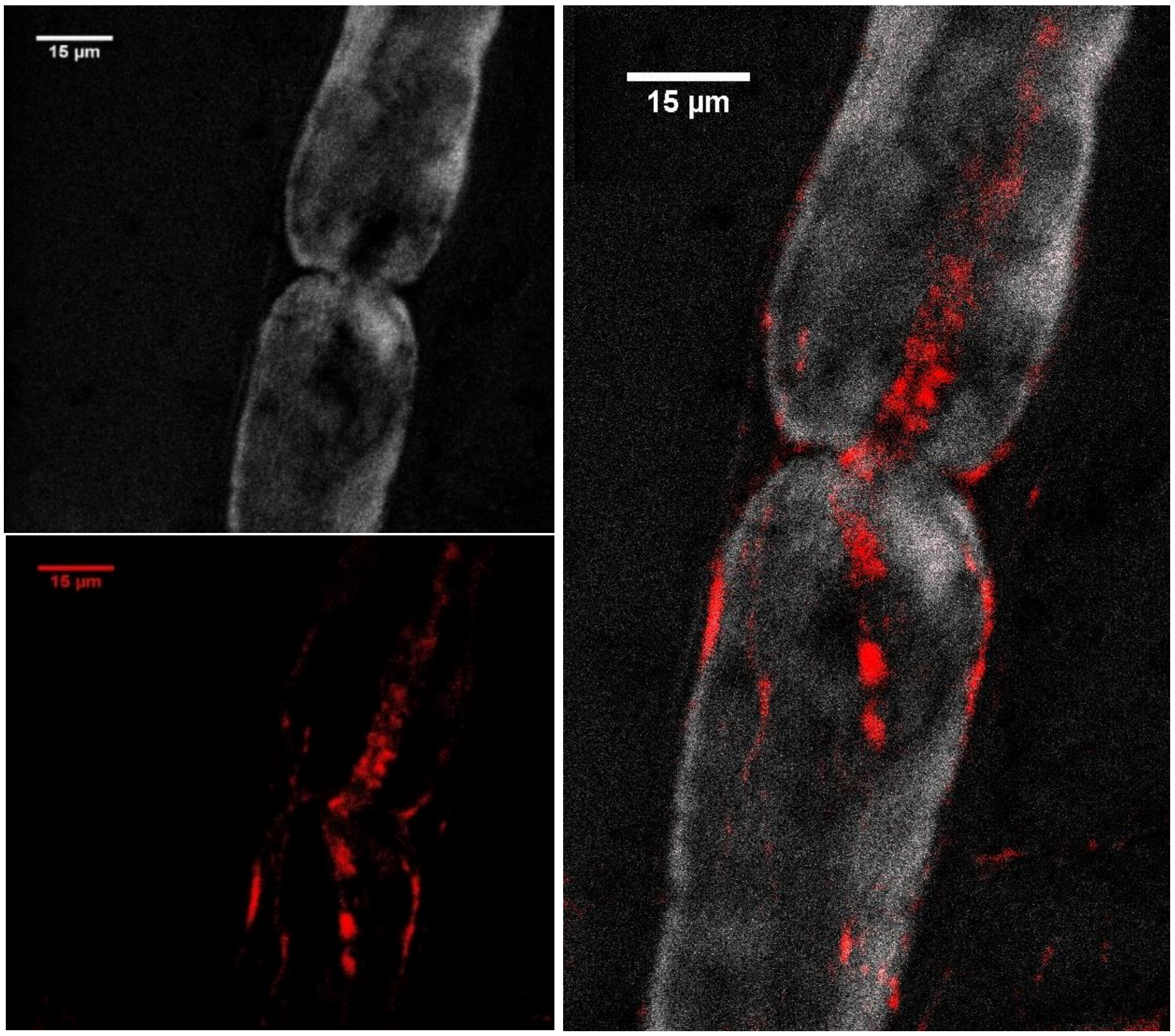


Figure 26 - Node of Ranvier. THG (top left), AntiCx32 (bottom left), merge (right)

In Figure 26 a particular of a Ranvier’s node is reported: the THG channel (gray), the anti-connexin32 (red) and the merge channel (on the right) are presented. It can be noted that a stronger red signal of connexin32 is found in correspondence with the node (even if aspecific signal is always presented too).

Another example of connexin plaque, enlightened in red with anti-Cx32, in correspondence of a Ranvier's node is shown in Figure 27. The red connexin indicator is highly present only in the region of the node; furthermore, the THG signal present more pronounced borders and less signal in the insider region, like it is expectable because of the myelin distribution around the central axon. This is visible very clearly also in the radial sample of Figure 24.

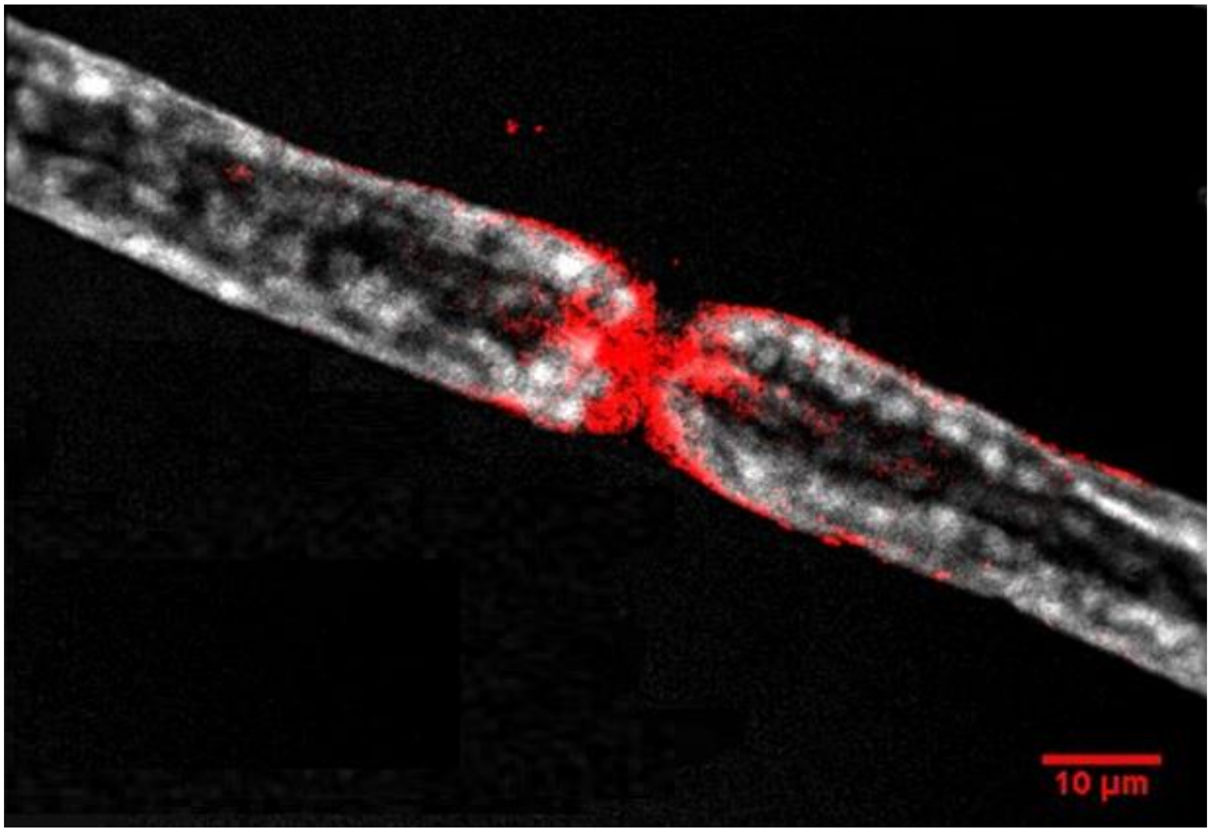


Figure 27 - Node of Ranvier in a mouse sciatic nerve (THG+anti-Cx32)

3D reconstruction of sciatic nerve

In the following the 3D reconstructions of the Schwann cells are reported.

In Figure 29 and Figure 29 3D-viewes of mouse sciatic nerve are visible: in the first one an entire myelin configuration is reconstructed, in the second a thin radial cut is reported: in this case the myelin is visible in gray around a central green core labelled by neurofilament.

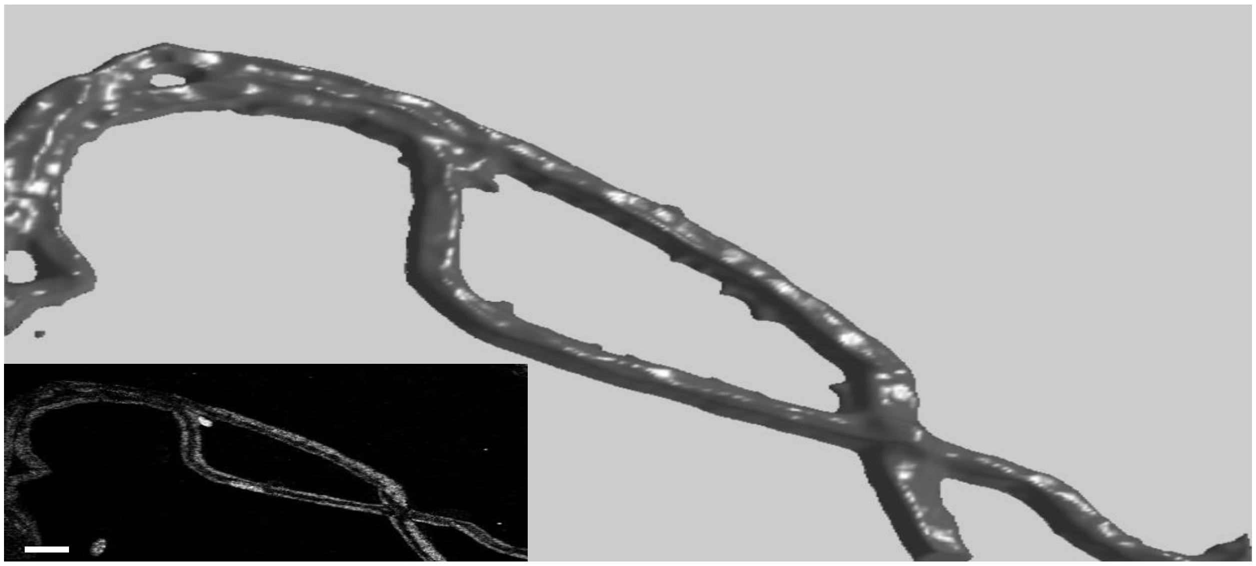


Figure 28 – Mouse sciatic nerves myelin reconstruction (scale bar 20µm)

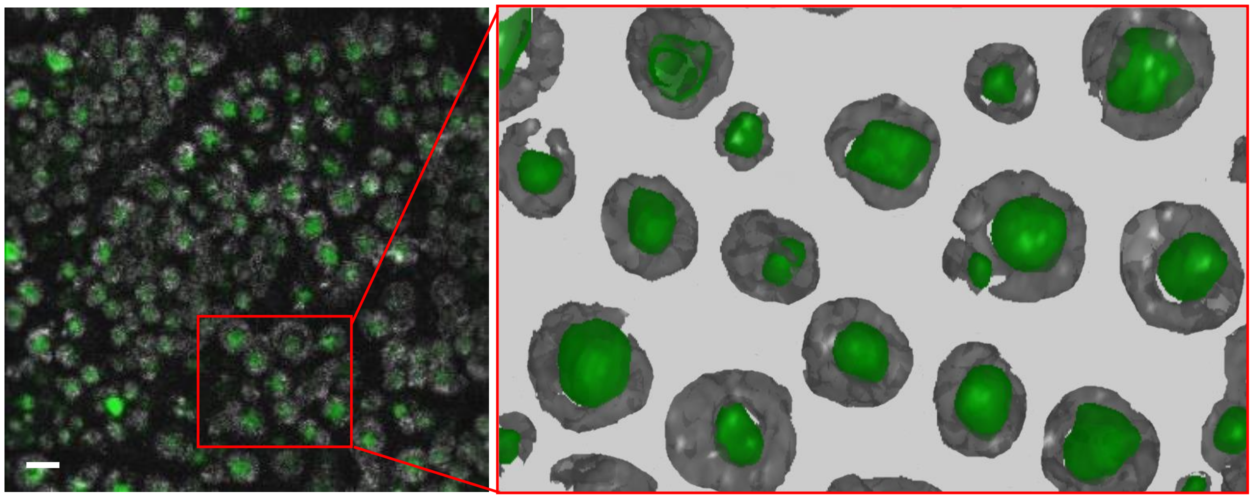


Figure 29 - Radial cut of a mouse sciatic nerve myelin and axons (scale bar 20µm)

In Figure 31 and Figure 31 two different reconstructions of a Schwann cell are reported, enlightening a node of Ranvier; the channel of Cx32 is visualized too.

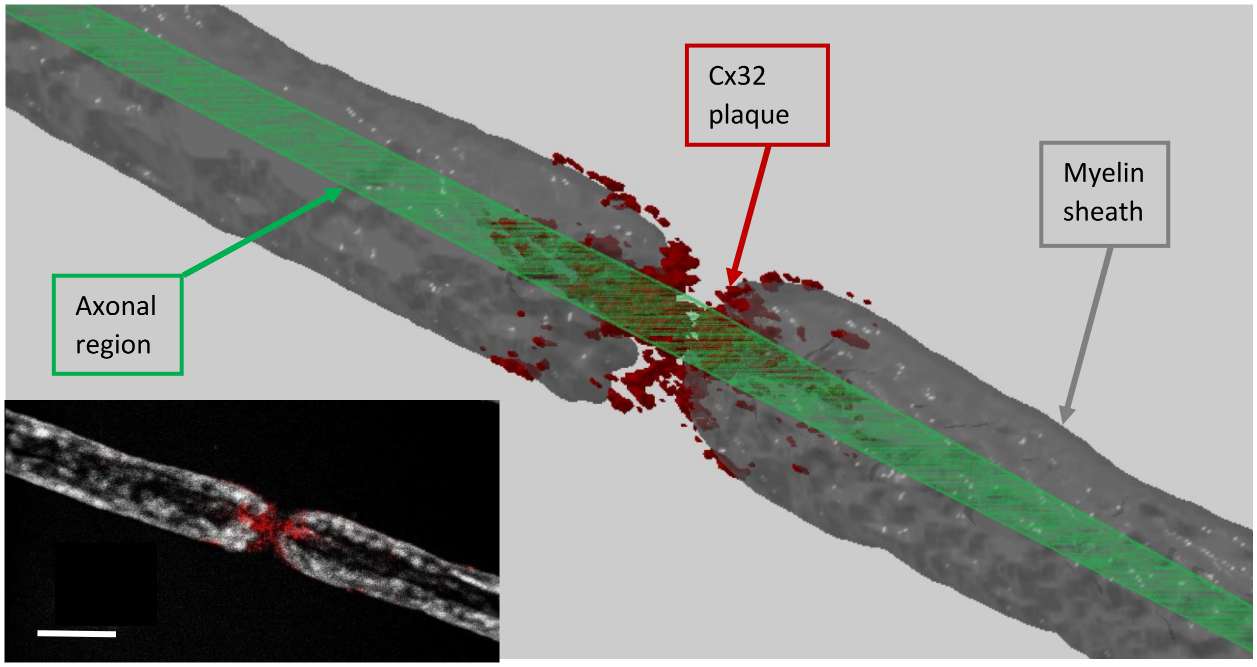


Figure 30 - Mouse Schwann cell reconstruction (scale bar 20µm)

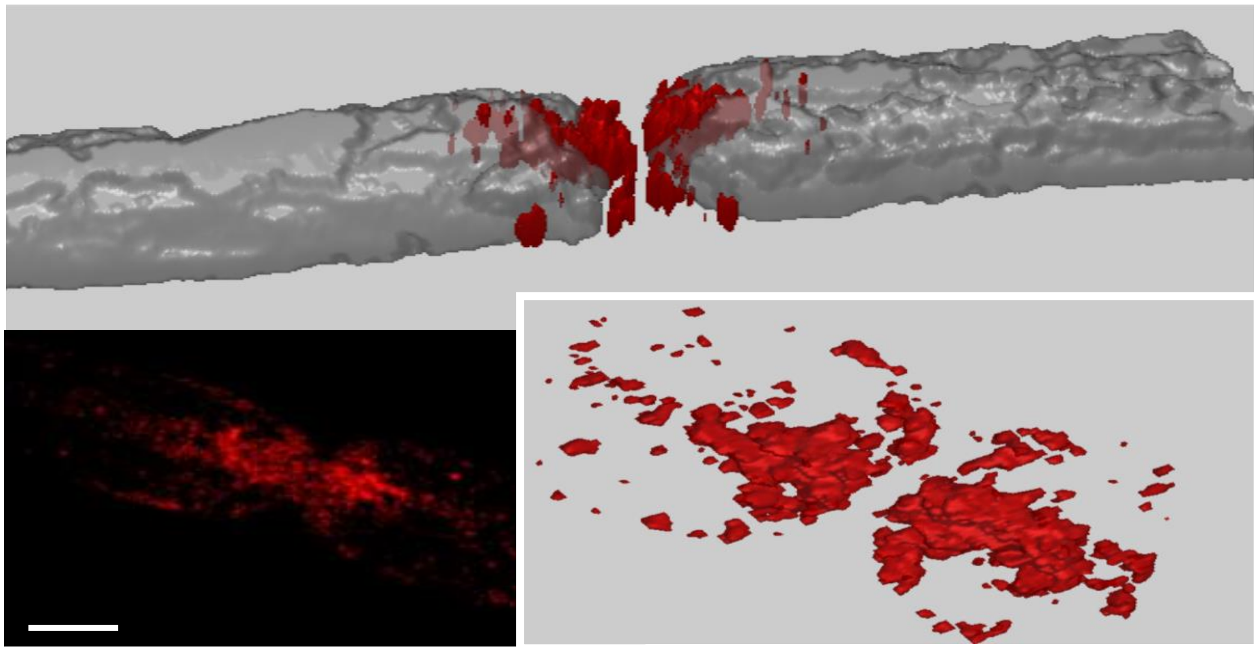


Figure 31 - Mouse sciatic nerve connexin reconstruction (scale bar 10µm)

In the first figure a Schwann reconstruction is reported, in which the myelin sheath is visible in gray thanks to THG signal, the region where the axon stays is enlightened in green, and the plaque of connexin is reconstructed in red using its antibody signal. From this image it's possible to understand the classical configuration of these kind of cells.

Particular interest is given in Figure 31 to connexin plaque: the single red channel is reported in the bottom figure giving an idea of its configuration around the Ranvier's node.

IV. Discussion

In this thesis work we setup the conditions to obtain high quality images of Schwann cells using 2PE and THG techniques. In particular, we obtained images of the myelin sheath, the nuclei and Cx32 plaques (Figure 24, Figure 25 and Figure 26).

A basic problem found in this work was to find the proper combination of dyes and antibodies in order to simultaneously stain different domains of the cell, considering the fact that we worked with 2PE. The fluorophores require significative excitation at half the wavelength of our laser, and at the same time they must have emission spectra in the acceptance regions of our PMTs, possibly not overlapping in these regions.

Particularly interesting were the experiments with antibody for connexin-32, that permits us to reveal a more intense signal in proximity of the nodes of Ranvier, in agreement with the literature (Figure 26 and Figure 27); the quality of these results can be improved, by reducing the aspecific signal: in the ideal case we would obtained a strong signal only in correspondence of connexins and no in other regions. This is one of the motivation that leads us to try to perform better immunofluorescence and to find most appropriate dies for our samples.

Secondly, THG permitted us to obtain very interesting results: we were able to demonstrate colocalization of this signal at myelin interfaces, as expected from the literature (Figure 24). The technique is very useful to enlighten the presence of the myelin sheath without the use of any fluorescent label.

We can conclude that the apparatus is well suited to perform in vitro study of the peripheral nervous system.

Another technique that we setup in our apparatus was STED; the setup can be adjusted to perform this kind of nanoscopy, even if, at present, no super-resolute results can be obtained. We have tried to optimize the path, the alignment and the power of the beams in order to achieve the state of art of STED characteristics, always limited by our physical configuration possibilities: unfortunately, the power of our beams is not so high in the back focal plane (in the order of one hundred mW) and the path cannot be modified in a substantial way, leading to results that are quite unsatisfactory with respect to the best literature results, like for example a depletion of only 65%. Furthermore, another problem can consist in the fact that our setup performs 2PE-STED, and recent studies have marked the fact that the most common dyes for STED imaging (like for example ATTO647, that we have used in our depletion experiments) are tested to work with 1PE-STED but no detailed study were performed with 2PE, on the contrary being in this case some unexpected effects that can disturb the STED signal (like for example 1PE signals that covers the 2PE one), increasing the background and not bringing to the excitation PSF reduction desired. A possibility is to find new dyes with spectra more efficient for 2PE-STED technique; another thing to think about is that, in a second time, one must be able to use this dyes with appropriate antibodies in order to link them to the different cell regions one wants to image.

In any case, at the present, our STED setup gives no image improvement with respect to 2PE (Figure 23), and further work has to be done in order to implement in a better way this technique in the apparatus.

The second part of this thesis work was related to image rendering; in particular a 3D-reconstruction starting from a z-stack must be obtained. Different software were compared deciding at the end to write an own program in Matlab framework: this permits us to reach a big customizability and the possibility to reach results comparable with professional programs (Figure 13). The main contra of working with Matlab were the slowness in rendering images of big dimensions and the effort to handle volumes: for example, object rotation slowness was overcome introducing a binning, that is a resizing of the pixels.

The preliminary results are very good and satisfactory, how it is visible from the reconstruction of our test cells, that are HeLa cells presenting three channels, the cytoplasm, the nuclei and the connexins plaques (Figure 13): all the cell domains were reconstructed with a good rendering and the merge was possible thanks to the transparency of some domains. The presence of more or less details depends on the smoothing value used for the reconstruction, so a good choice of the rendering parameters has to be done in order to obtain the desired precision in the reconstruction. In a second step, the sciatic nerve was reconstructed: the myelin was clearly rendered thanks to THG signal and also a preliminary reconstruction of Cx32 plaques were performed (Figure 30 and Figure 31). The results are not the best possible, sometimes being difficult to obtain good redenrings in this case, because of the quality of immunofluorescence images: for example in the case of Cx32 the rendering results are more difficultly obtained because of the presence of much aspecific signal.

In conclusion, there’s the possibility to obtain good 3D-reconstructions, and the customizability of the software permits to improve the results better such as to implement personalized analysis. But the domains of Schwann cell and in particular Cx32 plaques have to be investigated further, in order to obtain more detailed distribution maps.

V. Conclusions

This biophysical work was conceived as preliminary to obtain super-resolution images of Cx32 in human Schwann cells. The microscope apparatus was developed and optimized to obtain this result. Taking advantage of an optical parametric oscillator we obtained THG signals of the myelin in mouse samples; these results were combined with 2PE microscopy using three different channels which permitted us to visualize nuclei, the axon and Cx32 plaques at Ranvier nodes.

Cx32 precise localization was the main objective of this study as it plays a fundamental role in human Schwann cells. As recent works suggests that, not only GJs, but also Cx32 hemichannels have an important physiological function in neurolemmocytes, it is urgent to clarify where these hemichannels are located. This is why the super resolution STED technique is conceived in our study.

The 3D rendering software we have implemented in Matlab will be a platform analysis for the study of Cx32 that is expected to become a useful tool for scientists interested in using and developing it.

In collaboration with medical doctors of the Padua’s hospital, we will extend soon this work to the study of human samples to improve our understanding of the CMT1X neuropathy.

Bibliography

1. Kleopas A. Kleopa, I.S., *Connexins, gap junctions and peripheral neuropathy.* Neuroscience Letters, 2015. **596**: p. 27-32.
2. Hashem A.Dbouk, R.M.M., Marwan E.El-Sabban, Rabih S.Talhouk, *Connexins: a myriad of functions extending beyond assembly of gap junction channels.* Cell Communication and Signaling, 2009. **7:4**.
3. M.E.Shy, C.S., E.R.Swan, K.M.Krajewski,T.Doherty, D.R.Fuerst, P.J.Ainsworth, R.A.Lewis, S.S.Scherer, A.F.Hahn, *CMT1X phenotypes represent loss of GJB1 gene function.* Neurology, 2007. **68**: p. 849-855.
4. D.M.Menichella, D.A.G., E.Sirkowski,S.S.Scherer, D.I.Paul, *Connexins are critical for normal myelination in the CNS.* Journal Neuroscience, 2003. **23**: p. 5963-5973.
5. B.Odermatt, K.W., A.wallraff, G.Seifert, J.Degen, C.Euwens, B.Fuss, H.Bussow, K.Schilling, C.Steinhauser, K.Willecke, *Connexin 47 deficient mice with enhanced green fluorescent protein reporter gene reveal predominant oligodendrocytic expression of Cx47 and displasy vacuolized myelin in the CNS.* Journal Neuroscience, 2003. **23**: p. 4549-4559.
6. B. Uhlenberg, M.S., F. Ruschendorf, N. Ruf, A.M. Kaindl, M. Henneke, H. Thiele, G. Stoltenburg-Didinger, F. Aksu, H. Topaloglu, P. Nurnberg, C. Hubner, B. Weschke, J. Gartner, *Mutations in the gene encoding gap junction protein alpha12(Connexin46.6)causePelizaeus–Merzbacher-likedisease.* The American Journal of Human Genetics, 2004. **75**: p. 251-260.
7. Amanda Mierzwa, S.S., Jack Rosenbluth, *Permeability of the Paranodal Junction of Myelinated Nerve Fibers.* TheJournalofNeuroscience, 2010. **30(47)**: p. 15962-15968.
8. Jortner, B.S., *Preparation and analysis of the peripheral nervous system.* Toxicological pathology, 2011. **39**: p. 66-72.
9. Asao Hirano, H.M.D., *A structural analysis of the myelin sheath in the central nervous system.* The journal of cell biology, 1967. **34**: p. 555-568.
10. John W. Griffin, W.J.T., *Biology and Pathology of Nonmyelinating Schwann Cells.* Glia, 2008. **56**: p. 1518-1531.
11. Felipe A. Court, L.W., M. Laura Feltri, *Basal lamina: Schwann cells wrap to the rhythm of space-time.* Current Opinion in Neurobiology, 2006: p. 16:501-507.
12. Felipe A. Court, D.L.S., Thomas Pratt, Emer M. Garry, Richard R. Ribchester, David F. Cottrell, Susan M. Fleetwood-Walker, Peter J. Brophy, *Restricted growth of Schwann cells lacking Cajal bands slows conduction in myelinated nerves.* Nature, 2004: p. 431:191-195.
13. M.A.Chernousov, W.-M.Y., Z.-L. Chen, D.J.Caray, S.Strickland, *Regulation of Schwann Cell Function by the Extracellular Matrix.* Glia, 2008. **54**: p. 1498-1507.

14. Shumin Zhao, A.F., David C. Spray, *Characteristics of gap junction channels in Schwann cells from wild-type and connexin-null mice.* Annals New York academy of sciences.

15. Meejin Ahn, J.L., Andreas Gustafsson, Alan Enriquez,Eric Lancaster,Jai-Yoon Sul, Philip G. Haydon, David L. Paul, Yan Huang, Charles K. Abrams, Steven S. Scherer, *Cx29 and Cx32, Two Connexins Expressed by Myelinating Glia, Do Not Interact and Are Functionally Distinct.* Journal of Neuroscience Research 2008. **86**: p. 992-1006.

16. N.Kamasawa, A.S., M.Morita, T.Yasumura,K.G.V.Davidson,J.I.Nagy,J.E.Rash, *Connexin-47 and connexin-32 in gap junctions of oligodendrocyte somata, myelin sheaths, paranodal loops and Schmidt-Lanterman incisures: implications for ionic homeostasis and potassium siphoning.* Neuroscience, 2005. **136**: p. 65-86.

17. Donahue, G.J., *Facts About Charcot-Marie- Tooth Disease & Related Diseases*, in *Muscular Dystrophy Association*2009.

18. Steven S. Scherer, L.W., *Molecular Mechanisms of Inherited Demyelinating Neuropathies.* Glia, 2008. **56**: p. 1578-1589.

19. Irene Sargiannidou, G.-H., StylianaKyriakoudi, Baik-Lin Eun, Kleopas A. Kleopa, *A start codon CMT1X mutation associated with transient encephalomyelitis causes complete loss of Cx32.* Neurogenetics, 2015. **16**: p. 193-200.

20. Linda Jo Bone, S.M.D., Rita J. Balice-Gordon, Kenneth H. Fischbeck, Steven S. Scherer, *Connexin32 and X-linked Charcot–Marie–Tooth Diseas.* Neurobiology odf Disease, 1997. **4**: p. 221-230.

21. Pierre Mahou, N.O., Guillaume Labroille, Louise Duloquin, Jean-Marc Sintes, Nadine Peyrieras, Renaud Legouis, Delphine D´ebarre, Emmanuel Beaurepaire, *Combined third-harmonic generation and four-wave mixing microscopy of tissues and embryos.* Biomedical optics express, 2011. **2841**.

22. Pierre Mahou, M.Z., Karine Loulier, Katherine S Matho, Guillaume Labroille, Xavier Morin, Willy Supatto, Jean Livet, Delphine Débarre, Emmanuel Beaurepaire, *Multicolor two-photon tissue imaging by wavelength mixing* Nature methods, 2012. **9-8**: p. 815-822.

23. Benjamin Harke, J.K., Chaitanya K. Ullal, Volker Westphal, Andreas Schönle and Stefan W. Hell, *Resolution scaling in STED microscopy* Optics express, 2008. **4156**.

24. Janina Hanne, H.J.F., Frederik Goerlitz, Patrick Hoyer, Johann Engelhardt, Steffen J. Sahl, Stefan W. Hell, *STED nanoscopy with fluorescent quantum dots.* Nature communications, 2015. **8127**.

25. B.Hein, K.I.W., C.A.Wurm,V.Westphal,S.Jakobs,S.W.Hell, *Stimulated Emission Depletion Nanoscopy of Living Cells Using SNAP-Tag Fusion Proteins.* Biophysical Journal, 2010. **98**: p. 158-163.

26. S.Jacobs, C.A.W., *Super-resolution microscopy of mitochondria.* Current Opinion in Chemical Biology, 2014. **20**: p. 9-15.

27. Takeshi Watanabe, M.F., Yoshi Watanabe, Nobuhito Toyama and Yoshinori Iketaki *Generation of a doughnut-shaped beam using a spiral phase plate.* Review of scientific instruments, 2004.
28. Bettina Weigelin, G.-J.B., Peter Friedl, *Third harmonic generation microscopy of cells and tissue organization.* The Company of Biologists, 2016: p. 129:245-255.
29. Markus Rehberg, F.K., Ulrich Pohl, Steffen Dietzel, *Label-Free 3D Visualization of Cellular and Tissue Structures in Intact Muscle with Second and Third Harmonic Generation Microscopy.* Label-Free 3D Microscopy, 2011. **6**(11).
30. Omid Masihzadeh, T.C.L., Scott R. Domingue, Malik Y. Kahook, Randy A. Bartels, David A. Ammar, *Third harmonic generation microscopy of a mouse retina.* Molecular Vision 2015. **21**: p. 538-547.
31. Maryam Halavi, K.A.H., Ruchi Parekh and Giorgio A. Ascoli, *Digital Reconstructions of Neuronal Morphology: Three Decades of Research Trends.* Frontiers in Neuroscience, 2012. **6**: p. 49:1-11.
32. Hyungsik Lim, D.S., Imran Kassim, Yanqing Zhang, James L. Salzer, Carmen V. Melendez-Vasquez, *Label-free imaging of Schwann cell myelination by third harmonic generation microscopy.* PNAS, 2014. **111**: p. 50:18025-18030.
33. Jennifer K.Ness, A.A.S., Eng-Hui Yap, Eduardo J.Fajardo, Andras Fiser, Nikos Tapinos, *Nuc-ErbB3 Regulates H3K27me3 Levels and HMT Activity to Establish Epigenetic Repression during Peripheral Myelination.* Glia, 2016. **64**: p. 977-992.
34. Nualart-Marti, A., et al., *Role of connexin 32 hemichannels in the release of ATP from peripheral nerves.* Glia, 2013. **61**(12): p. 1976-89.
35. Pedro Felipe Gardeazábal Rodríguez, Y.W., Harpreet Singh, Hui Zhao, Ligia Toro, Enrico Stefani, *Building a fast scanning stimulated emission depletion microscope: a step by step guide* Current Microscopy Contributions to Advances in Science and Technology, 2012: p. 791-801.
36. M.G.M.Velasco, E.S.A., P.Yuan,J.Grutzendler, J.Bewersdorf, *Absolute two-photon excitation spectra of red and far-red fluorescent probes* Optics Letters, 2015. **40**: p. 4915-4919.
37. Hanchuan Peng, A.B., Zhi Zhou, Giulio Iannello, Fuhui Long, *Extensible visualization and analysis for multidimensional images using Vaa3D.* Nature Protocols, 2014. **9:1**: p. 193-208.

Appendix

i. Matlab code

S2tD

```
function varargout = s2tD(varargin)
% S2TD M-file for s2tD.fig
%      S2TD, by itself, creates a new S2TD or raises the existing
%      singleton*.
%
%      H = S2TD returns the handle to a new S2TD or the handle to
%      the existing singleton*.
%
%      S2TD('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in S2TD.M with the given input arguments.
%
%      S2TD('Property','Value',...) creates a new S2TD or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before s2tD_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to s2tD_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help s2tD

% Last Modified by GUIDE v2.5 21-Jun-2016 11:38:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @s2tD_OpeningFcn, ...
                   'gui_OutputFcn',  @s2tD_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before s2tD is made visible.
function s2tD_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to s2tD (see VARARGIN)

% Choose default command line output for s2tD
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes s2tD wait for user response (see UIRESUME)
% uiwait(handles.base);


% --- Outputs from this function are returned to the command line.
function varargout = s2tD_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%


global l; %counter to refresh the currentStack (the stack that it's loaded in the workspace)
l=0;
global numPixel;
numPixel=512;  % adjust here the number of pixel in the images
global precision;
precision=255;  % adjust in relation to variable type: uint8 255, uint16 65535
%set z-thickness parameters (scale)
global parameters;
if get(handles.scale, 'String')=='-' | get(handles.zStep, 'String')=='-'
    parameters.scale=NaN;
    parameters.zStep=NaN;
else
   parameters.scale=str2double(get(handles.scale,'String'));
   parameters.zStep=str2double(get(handles.zStep,'String'));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% Load the workspace
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in chooseFile - choose the images from your folder
function chooseFile_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global l;
global n; %counter for the current slice (the visualized one)
global numPixel;
global popupNames;
n=1;
l=l+1;
currentStack=l;
dataStack(currentStack).file.load=l;
% choose files saving: p=basic path, f{}=list of imported files
[f,p] = uigetfile({'*.*','All Files'},'Open sequence file','MultiSelect','on', 'C:\Users\mirko\Desktop\Tesi\Vecchi');
if p==0
   display('No file selected');
   return;
end
%call saveAs to give a name to the imported channel
waitfor(saveAs);
%refresh loaded stacks popup with the new channel
pop_up=get(handles.chooseSelectedFile, 'String');
num=numel(pop_up)+1;
skip=0; %counter for the popup menu
for j=1:num-1
    if strcmp(pop_up{j},getappdata(0, 'varName')) %getappdata=title from saveAs function
        choice = questdlg('A channel with the same name already exist!','Name overlap','Overwrite','Cancel','Cancel');
        switch choice
        case 'Cancel'
            skip=1;
            break;
        case 'Overwrite'
            skip=2;
            num=j;
            currentStack=j-1;
            break;
        end
    end
end
if skip==0 % no name overlapping
    dataStack(currentStack).file.title =getappdata(0, 'varName');
    pop_up{num}=dataStack(currentStack).file.title;
    set(handles.chooseSelectedFile, 'String', pop_up);
end
popupNames=pop_up;
%update dataStack
dataStack(currentStack).file.path=p; %path
dataStack(currentStack).file.names=f; %file names
```

```
set(handles.chooseSelectedFile, 'Value', num);
dataStack(currentStack).file.nSlices=numel(dataStack(currentStack).file.names); %number of slices
%refresh path
set(handles.pathName,'String',p);
%3d matrix creation
dataStack(currentStack).modifiedMatrix = zeros(numPixel,numPixel,dataStack(currentStack).file.nSlices); %zero
matrix
for i=1:dataStack(currentStack).file.nSlices
    filename=[dataStack(currentStack).file.path,dataStack(currentStack).file.names{i}];
    dataStack(currentStack).rawMatrix(:,:,i)=imread(filename); % load the raw image
    dataStack(currentStack).modifiedMatrix(:,:,i)=im2bw(dataStack(currentStack).rawMatrix(:,:,i), 0.1); %bw
convertion (saved in modifiedMatrix)
end
%visulization
imshow(dataStack(currentStack).rawMatrix(:,:,n),'Parent',handles.rawFig);
imshow(dataStack(currentStack).modifiedMatrix(:,:,n),'Parent',handles.modifiedFig);
%setting min max values of grayscale
dataStack(currentStack).min=min(min(min(dataStack(currentStack).rawMatrix)));
dataStack(currentStack).max=max(max(max(dataStack(currentStack).rawMatrix)));
set(handles.minGray,'String',num2str( dataStack(currentStack).min));
set(handles.maxGray,'String',num2str(dataStack(currentStack).max));
set(handles.maxSlider, 'Max', dataStack(currentStack).max , 'Min', dataStack(currentStack).min, 'Value',
dataStack(currentStack).max);
set(handles.minSlider, 'Max',dataStack(currentStack).max , 'Min', dataStack(currentStack).min, 'Value',
dataStack(currentStack).min);
%set number of slices
set(handles.currentImage, 'String', num2str(n));
set(handles.totSlices, 'String', ['\ ' num2str(dataStack(currentStack).file.nSlices)]);



% --- Executes on selection change in chooseSelectedFile - choose a channel from the popup menu
function chooseSelectedFile_Callback(hObject, eventdata, handles)
global currentStack;
global dataStack;
global n;
n=1;
currentStack=get(handles.chooseSelectedFile, 'Value');
%visualization
imshow(dataStack(currentStack).rawMatrix(:,:,n),'Parent',handles.rawFig);
imshow(dataStack(currentStack).modifiedMatrix(:,:,n),'Parent',handles.modifiedFig);
% setting min max values of grayscale
dataStack(currentStack).min=double(min(min(min(dataStack(currentStack).rawMatrix))));
dataStack(currentStack).max=double(max(max(max(dataStack(currentStack).rawMatrix))));
set(handles.minGray,'String',num2str(dataStack(currentStack).min));
set(handles.maxGray,'String',num2str(dataStack(currentStack).max));
set(handles.maxSlider, 'Max', dataStack(currentStack).max , 'Min', dataStack(currentStack).min, 'Value',
dataStack(currentStack).max);
set(handles.minSlider, 'Max',dataStack(currentStack).max , 'Min', dataStack(currentStack).min, 'Value',
dataStack(currentStack).min);
%set number of slices
set(handles.currentImage, 'String', num2str(n));
set(handles.totSlices, 'String', ['\ ' num2str(dataStack(currentStack).file.nSlices)]);
%set z-thickness parameters (scale)
if isnan(dataStack(currentStack).file.scale) && isnan(dataStack(currentStack).file.zStep)
    set(handles.scale, 'String','-');
    set(handles.zStep, 'String','-');
else
    set(handles.scale, 'String', num2str(dataStack(currentStack).file.scale));
    set(handles.zStep, 'String', num2str(dataStack(currentStack).file.zStep));
end

function chooseSelectedFile_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pathName_CreateFcn(hObject, eventdata, handles)


%set the scale value (um/pixel)
function scale_Callback(hObject, eventdata, handles)
global parameters;
parameters.scale=str2double(get(handles.scale,'String'));

function scale_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%set the z step (um/step)
function zStep_Callback(hObject, eventdata, handles)
global parameters;
parameters.zStep=str2double(get(handles.zStep,'String'));

function zStep_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%Move through the slices
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in previous image.
function previous_Callback(hObject, eventdata, handles)
global n;
if n>1
    n=n-1;
    %adjust grayscale levels
    adjustGrayScale(hObject, eventdata, handles);
end
set(handles.currentImage,'String',sprintf('%0.0f',n));

% --- Executes on button press in next image.
```

```
function next_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global n;
if n<dataStack(currentStack).file.nSlices
    n=n+1;
    %adjust grayscale levels
    adjustGrayScale(hObject, eventdata, handles);
end
set(handles.currentImage,'String',sprintf('%0.0f',n));


function currentImage_Callback(hObject, eventdata, handles)
global n;
n=str2double(get(handles.currentImage,'String'));
%adjust grayscale levels
adjustGrayScale(hObject, eventdata, handles);

function currentImage_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Image adjustment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% --- Executes on maximun gray limit slider movement.
function maxSlider_Callback(hObject, eventdata, handles)
adjustGrayScale(hObject, eventdata, handles)

function maxSlider_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on minimun gray limit slider movement.
function minSlider_Callback(hObject, eventdata, handles)
adjustGrayScale(hObject, eventdata, handles)

function minSlider_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function adjustGrayScale(hObject, eventdata, handles)
global dataStack;
global currentStack;
global n;
global numPixel;
global precision;
matrix=dataStack(currentStack).rawMatrix(:,:,n);
%stretch grayscale vaues for slice n
maxsl=get(handles.maxSlider, 'Value');
minsl=get(handles.minSlider, 'Value');
if minsl>maxsl
    for i=1:numPixel
        for j=1:numPixel
            matrix(i,j)=precision-double(precision)*double(matrix(i,j)-maxsl)/double(minsl-maxsl);
        end
    end
else
    for i=1:numPixel
        for j=1:numPixel
            matrix(i,j)=double(precision)*double(matrix(i,j)-minsl)/double(maxsl-minsl);
        end
    end
end
set(handles.minGray,'String',num2str(minsl));
set(handles.maxGray,'String',num2str(maxsl));
%visualization
imshow(matrix,'Parent',handles.rawFig);
imshow(dataStack(currentStack).modifiedMatrix(:,:,n),'Parent',handles.modifiedFig);


% --- Executes on black&white slider movement.
function blackWhiteSlider_Callback(hObject, eventdata, handles)
val=get(handles.blackWhiteSlider, 'Value');
set(handles.blackWhiteThreashold, 'String', num2str(val));
convertbw(hObject, eventdata, handles);

function blackWhiteSlider_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on black&white threashold setting
function blackWhiteThreashold_Callback(hObject, eventdata, handles)
val=get(handles.blackWhiteThreashold, 'String');
set(handles.blackWhiteSlider, 'Value', str2num(val));
convertbw(hObject, eventdata, handles);

function blackWhiteThreashold_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on smoothing radius setting
function smoothingRadius_Callback(hObject, eventdata, handles)
convertbw(hObject, eventdata, handles);

function smoothingRadius_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


function smoothingIterations_Callback(hObject, eventdata, handles)
convertbw(hObject, eventdata, handles);

function smoothingIterations_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press 'fill the holes'
function fill_Callback(hObject, eventdata, handles)
convertbw(hObject, eventdata, handles);


function convertbw(hObject, eventdata, handles)
global dataStack;
global currentStack;
global n;
global matrix;
global precision;
matrix=dataStack(currentStack).rawMatrix(:,:,n);
thr=get(handles.blackWhiteSlider, 'Value')/precision; %bw threashold
rad=str2double(get(handles.smoothingRadius,'String')); %smoothing radius
iter=str2double(get(handles.smoothingIterations,'String')); %smoothing iterations
if rad==0||iter==0; %no smoothing;
else   %smooth
    for i=1:iter
        FA=fspecial('average', rad);
        matrix=imfilter(matrix,FA);
    end
end
if get(handles.fill,'Value')==0; %no filling the holes
else   %fill
    matrix=imfill(matrix, 'holes');
end
%bw conversion
matrix=im2bw(matrix,thr);
%visulaization
imshow(matrix,'Parent',handles.modifiedFig);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
%Keep modifications
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in Save for current image.
function keepImg_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global n;
global matrix;
global precision;
dataStack(currentStack).modifiedMatrix(:,:,n)=matrix;
%save parameters
dataStack(currentStack).parameters.bw=get(handles.blackWhiteSlider, 'Value')/precision;
dataStack(currentStack).parameters.rad=str2double(get(handles.smoothingRadius,'String'));
dataStack(currentStack).parameters.smooth=str2double(get(handles.smoothingIterations,'String'));
dataStack(currentStack).parameters.fill=get(handles.fill,'Value');
%display
imshow(dataStack(currentStack).modifiedMatrix(:,:,n),'Parent',handles.modifiedFig);


% --- Executes on button press in Save for all images.
function keepAll_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global precision;
thr=get(handles.blackWhiteSlider, 'Value')/precision;
for i=1:dataStack(currentStack).file.nSlices
    matrix=dataStack(currentStack).rawMatrix(:,:,i);
    matrix=im2bw(matrix,thr);
    rad=str2double(get(handles.smoothingRadius,'String'));
    iter=str2double(get(handles.smoothingIterations,'String'));
    if rad==0||iter==0; %no smoothing
    else   %smooth
        for j=1:iter
            FA=fspecial('average', rad);
            matrix=imfilter(matrix,FA);
        end
    end
    if get(handles.fill, 'Value')==0; %no filling
    else   %fill
        matrix=imfill(matrix, 'holes');
    end
    dataStack(currentStack).modifiedMatrix(:,:,i)=matrix;
end;
%save parameters
dataStack(currentStack).parameters.bw=get(handles.blackWhiteSlider, 'Value')/precision;
dataStack(currentStack).parameters.rad=str2double(get(handles.smoothingRadius,'String'));
dataStack(currentStack).parameters.smooth=str2double(get(handles.smoothingIterations,'String'));
dataStack(currentStack).parameters.fill=get(handles.fill,'Value');


% --- Executes on button press in rendering3D.
function rendering3D_Callback(hObject, eventdata, handles)
% call the new function to perform 3d
rendering3D;




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load and save project
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in saveProj.
function saveProj_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global l;
global numPixel;
global precision;
global popupNames;
global endFigure;
global num;
global pop_up;
global parameters;
uisave({'dataStack','currentStack','l','numPixel','precision','popupNames','endFigure','num','pop_up','parameters'},
'zStack_1');


% --- Executes on button press in loadProj.
function loadProj_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global n;
global num;
global endFigure;
global pop_up;
global parameters;
global popupNames;
n=1;
uiload;
%refresh 2D window
set(handles.chooseSelectedFile,'String', dataStack(currentStack).file.title);
set(handles.chooseSelectedFile, 'Value', currentStack);
set(handles.pathName,'String', dataStack(currentStack).file.path);
set(handles.chooseSelectedFile, 'String', popupNames);
set(handles.currentImage, 'String', num2str(n));
set(handles.totSlices, 'String', ['\ ' num2str(dataStack(currentStack).file.nSlices)]);
%parameters 2D
set(handles.blackWhiteThreashold,'String', dataStack(currentStack).parameters.bw);
set(handles.blackWhiteSlider,'Value', str2double(dataStack(currentStack).parameters.bw));
set(handles.smoothingRadius,'String', dataStack(currentStack).parameters.rad);
set(handles.smoothingIterations,'String', dataStack(currentStack).parameters.smooth);
set(handles.fill,'Value', dataStack(currentStack).parameters.fill);
%display
imshow(dataStack(currentStack).rawMatrix(:,:,n),'Parent',handles.rawFig);
imshow(dataStack(currentStack).modifiedMatrix(:,:,n),'Parent',handles.modifiedFig);
%3D
rendering3D
%refresh 3D window
handles=guihandles;
set(handles.popupmenu2,'String', pop_up);
set(handles.popupmenu2, 'Value', num);
```

Rendering3D

```
function varargout = rendering3D(varargin)
% RENDERING3D M-file for rendering3D.fig
%      RENDERING3D, by itself, creates a new RENDERING3D or raises the existing
%      singleton*.
%
%      H = RENDERING3D returns the handle to a new RENDERING3D or the handle to
%      the existing singleton*.
%
%      RENDERING3D('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RENDERING3D.M with the given input arguments.
%
%      RENDERING3D('Property','Value',...) creates a new RENDERING3D or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before rendering3D_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to rendering3D_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help rendering3D

% Last Modified by GUIDE v2.5 22-Jul-2016 11:15:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @rendering3D_OpeningFcn, ...
                   'gui_OutputFcn',  @rendering3D_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before rendering3D is made visible.
function rendering3D_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes rendering3D wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = rendering3D_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


global popupNames;
global dataStack;
global currentStack;
global parameters;
set(handles.channelsPopup,'String', popupNames);
set(handles.channelsPopup,'Value',currentStack);
dataStack(currentStack).parameters.binning=0;
if isnan(parameters.scale) || isnan(parameters.zStep);
    zThickness=1;
else
    zThickness=(parameters.scale/parameters.zStep);
end
set(handles.zThick,'String',num2str(zThickness));

% --- Executes on selection change in channelsPopup.
function channelsPopup_Callback(hObject, eventdata, handles)
global currentStack;
global dataStack;
global lightcam;
    lightcam.val=0;
    currentStack=get(handles.channelsPopup, 'Value');
    dataStack(currentStack).parameters.binning=0;
    dataStack(currentStack).parameters.color='red';
    set(handles.red, 'Value',1);
    set(handles.transpSlider, 'Value', 1);
    set(handles.transparency, 'String', num2str(1));

function channelsPopup_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function defineBinning(hObject, eventdata, handles)
global dataStack;
global currentStack;
rad_on=get(get(handles.uipanel6,'selectedobject'),'Tag');
if isempty(rad_on)
  dataStack(currentStack).parameters.binning=0;
else
   switch rad_on
      case 'radiobutton11'
      dataStack(currentStack).parameters.binning=2;
      case 'radiobutton12'
      dataStack(currentStack).parameters.binning=4;
      case 'radiobutton13'
      dataStack(currentStack).parameters.binning=8;
      case 'radiobutton14'
      dataStack(currentStack).parameters.binning=10;
   end
end

% --- Executes on button press in radiobutton11.
function radiobutton11_Callback(hObject, eventdata, handles)
defineBinning(hObject, eventdata, handles);


% --- Executes on button press in radiobutton12.
function radiobutton12_Callback(hObject, eventdata, handles)
defineBinning(hObject, eventdata, handles);


% --- Executes on button press in radiobutton13.
function radiobutton13_Callback(hObject, eventdata, handles)
defineBinning(hObject, eventdata, handles);


% --- Executes on button press in radiobutton14.
function radiobutton14_Callback(hObject, eventdata, handles)
defineBinning(hObject, eventdata, handles);

function smoothValue_Callback(hObject, eventdata, handles)
val=str2num(get(handles.smoothValue, 'String'));
if val==0
elseif mod(val,2)==0
   warndlg('Insert an integer odd number!','warn');
   set(handles.smoothValue, 'String',num2str(val)+1)
end

function smoothValue_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function smoothIterations_Callback(hObject, eventdata, handles)

function smoothIterations_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function isoValue_Callback(hObject, eventdata, handles)

function isoValue_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function RGBred_Callback(hObject, eventdata, handles)
defineColorAdvance(hObject, eventdata, handles);

function RGBred_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function RGBgreen_Callback(hObject, eventdata, handles)
defineColorAdvance(hObject, eventdata, handles);

function RGBgreen_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function RGBblue_Callback(hObject, eventdata, handles)
defineColorAdvance(hObject, eventdata, handles);

function RGBblue_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in red.
function red_Callback(hObject, eventdata, handles)
if get(handles.red, 'Value')==0
    set(handles.red, 'Value',1);
else
    defineColor(hObject, eventdata, handles);
end

% --- Executes on button press in green.
function green_Callback(hObject, eventdata, handles)
if get(handles.green, 'Value')==0
    set(handles.green, 'Value',1);
```

```
else
    defineColor(hObject, eventdata, handles);
end


% --- Executes on button press in blue.
function blue_Callback(hObject, eventdata, handles)
if get(handles.blue, 'Value')==0
    set(handles.blue, 'Value',1);
else
    defineColor(hObject, eventdata, handles);
end

% --- Executes on button press in black.
function black_Callback(hObject, eventdata, handles)
if get(handles.black, 'Value')==0
    set(handles.black, 'Value',1);
else
    defineColor(hObject, eventdata, handles);
end


function defineColor(hObject, eventdata, handles)
global dataStack;
global currentStack;
rad_on=get(get(handles.uipanel2,'selectedobject'),'Tag');
switch rad_on
 case 'red'
 dataStack(currentStack).parameters.color3d='red';
 set(handles.RGBred, 'String', num2str(255));
 set(handles.RGBgreen, 'String', num2str(0));
 set(handles.RGBblue, 'String', num2str(0));
 case 'green'
 dataStack(currentStack).parameters.color3d='green';
 set(handles.RGBred, 'String', num2str(0));
 set(handles.RGBgreen, 'String', num2str(255));
 set(handles.RGBblue, 'String', num2str(0));
 case 'blue'
 dataStack(currentStack).parameters.color3d='blue';
 set(handles.RGBred, 'String', num2str(0));
 set(handles.RGBgreen, 'String', num2str(0));
 set(handles.RGBblue, 'String', num2str(255));
 case 'black'
 dataStack(currentStack).parameters.color3d='black';
 set(handles.RGBred, 'String', num2str(0));
 set(handles.RGBgreen, 'String', num2str(0));
 set(handles.RGBblue, 'String', num2str(0));
end
if isnan(dataStack(currentStack).parameters.p);
else
    adjustParam(hObject, eventdata, handles);
end


function defineColorAdvance(hObject, eventdata, handles)
global dataStack;
global currentStack;
r=str2double(get(handles.RGBred, 'String'));
g=str2double(get(handles.RGBgreen, 'String'));
b=str2double(get(handles.RGBblue, 'String'));
if r<0||r>255||g<0||g>255||b<0||b>255
    msgbox('RGB colors are defined by numbers between 0 and 255!', 'Warning','warn');
    if r<0
       set(handles.RGBred, 'String', num2str(0));
       r=0;
    elseif r>255
       set(handles.RGBred, 'String', num2str(255));
       r=255;
    elseif g<0
       g=0;
       set(handles.RGBgreen, 'String', num2str(0));
    elseif g>255
       g=255;
       set(handles.RGBgreen, 'String', num2str(255));
    elseif b<0
       b=0;
       set(handles.RGBblue, 'String', num2str(0));
    elseif b>255
       b=255;
       set(handles.RGBblue, 'String', num2str(255));
    end
end
dataStack(currentStack).parameters.color3d=[r/255, g/255, b/255];
set(handles.red, 'Value', 0);
set(handles.green, 'Value', 0);
set(handles.blue, 'Value', 0);
set(handles.black, 'Value', 0);


if isnan(dataStack(currentStack).parameters.p);
else
   adjustParam(hObject, eventdata, handles);
end


% --- Executes on slider movement.
function transpSlider_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
val=get(handles.transpSlider, 'Value');
set(handles.transparency, 'String', num2str(val));
if isnan(dataStack(currentStack).parameters.p);
else
    adjustParam(hObject, eventdata, handles);
end

function transpSlider_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function transparency_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
val=str2num(get(handles.transparency, 'String'));
if val<0||val>1
    msgbox('Give a number normalized between 0 and 1!', 'Warning','warn');
    if val<0
        val=0;
        set(handles.transparency, 'String', num2str(val));
    elseif val>1
        val=1;
        set(handles.transparency, 'String', num2str(val));
    end
end
set(handles.transpSlider, 'Value', val);
if isnan(dataStack(currentStack).parameters.p);
else
    adjustParam(hObject, eventdata, handles);
end

function transparency_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in light.
function light_Callback(hObject, eventdata, handles)
global lightcam;
lightcam.val=1;
adjustParam(hObject, eventdata, handles);



% --- Executes on button press in resetLight.
function resetLight_Callback(hObject, eventdata, handles)
global lightcam;
lightcam.val=0;
adjustParam(hObject, eventdata, handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in compute3D.
function compute3D_Callback(hObject, eventdata, handles)
rad_on=get(get(handles.uipanel6,'selectedobject'),'Tag');
if isempty(rad_on)
  waitfor(msgbox('You can select a binning for a faster rotation!', 'Tip'));
end
axes(handles.axes2);
[az,el] = view;
start3D(hObject, eventdata, handles);
perform3D(hObject, eventdata, handles);
view(az, el);
```

```
function start3D(hObject, eventdata, handles)
global dataStack;
global currentStack;
dataStack(currentStack).parameters.color3d='red';
dataStack(currentStack).parameters.iso=str2double(get(handles.isoValue, 'String'));
dataStack(currentStack).parameters.smoothIterations=str2double(get(handles.smoothIterations, 'String'));
dataStack(currentStack).parameters.smoothValue=str2double(get(handles.smoothValue, 'String'));
barra = waitbar(0,'Please wait...');
set(handles.figure1,'Pointer','watch');
defM=dataStack(currentStack).modifiedMatrix;

waitbar(1/4);
if str2num(get(handles.smoothIterations, 'String'))==0
else
    dataStack(currentStack).V = smooth3(defM,'box',dataStack(currentStack).parameters.smoothValue);
end
waitbar(2/4);
for i=2:dataStack(currentStack).parameters.smoothIterations
    dataStack(currentStack).V =
smooth3(dataStack(currentStack).V,'box',dataStack(currentStack).parameters.smoothValue);
end

[x,y,z]=size(dataStack(currentStack).V);
if dataStack(currentStack).parameters.binning==0
    dataStack(currentStack).Vr=dataStack(currentStack).V;
else
    dataStack(currentStack).Vr=resize(dataStack(currentStack).V,[x/dataStack(currentStack).parameters.binning,
y/dataStack(currentStack).parameters.binning, z/dataStack(currentStack).parameters.binning]);
end
[x,y,z]=size(dataStack(currentStack).Vr);
[dataStack(currentStack).parameters.X,dataStack(currentStack).parameters.Y,dataStack(currentStack).parameters.Z]
= meshgrid(1:x,1:y,1:z);
waitbar(3/4);
dataStack(currentStack).parameters.FV =
isosurface(dataStack(currentStack).parameters.X,dataStack(currentStack).parameters.Y,dataStack(currentStack).par
ameters.Z,dataStack(currentStack).Vr,dataStack(currentStack).parameters.iso);
dataStack(currentStack).visible=1;
waitbar(4/4);
close(barra);

function perform3D(hObject, eventdata, handles)
global dataStack;
global currentStack;
axes(handles.axes2);
cla reset;
dataStack(currentStack).parameters.p = patch(dataStack(currentStack).parameters.FV);
isonormals(dataStack(currentStack).parameters.X,dataStack(currentStack).parameters.Y,dataStack(currentStack).par
ameters.Z,dataStack(currentStack).Vr, dataStack(currentStack).parameters.p);
adjustParam(hObject, eventdata, handles);
set(handles.figure1,'Pointer','arrow');


function adjustParam(hObject, eventdata, handles)
global dataStack;
global currentStack;
global lightcam;
[az,el] = view;
if ishandle(dataStack(currentStack).parameters.p)
    dataStack(currentStack).parameters.zThick=str2double(get(handles.zThick, 'String'));
    dataStack(currentStack).parameters.transp=get(handles.transpSlider, 'Value');

set(dataStack(currentStack).parameters.p,'FaceColor',dataStack(currentStack).parameters.color3d,'EdgeColor','none',
'EdgeLighting','flat');
    daspect([1 1 dataStack(currentStack).parameters.zThick])
    lighting gouraud;
    cameratoolbar('Show');
    axis off;
    if lightcam.val==1
       h = camlight('headlight','infinite');
    else
       lighting none;
    end
    set(dataStack(currentStack).parameters.p,'FaceAlpha',dataStack(currentStack).parameters.transp);
    view(az,el);
    set(gca,'Color',[0.55 0.55 0.55]); %gray background
    ax=gca;
    guidata(hObject,handles);
else
    warndlg('Re-perform 3D rendering first!','warn');
end
drawnow;


% --- Executes on button press in keep3d.
function keep3d_Callback(hObject, eventdata, handles)
global dataStack;
global currentStack;
global endFigure;
global num;
global pop_up;
skip=0;
if isnan(dataStack(currentStack).parameters.p)
    warndlg('Perform 3D rendering first!','warn');
else
    pop_up=get(handles.finalRenderings, 'String');
    for i=1:numel(pop_up)
        if strcmp(dataStack(currentStack).file.title, pop_up{i})
            choice = questdlg('A channel with the same name already exist!','Name
overlap','Overwrite','Cancel','Cancel');
            switch choice
            case 'Cancel'
            skip=2;
            break;
            case 'Overwrite'
            skip=1;
            num=i;
            break;
            end
        end
    end
    if skip==0||skip==1
        if skip==0
            num=numel(pop_up)+1;
        end
        pop_up{num}=dataStack(currentStack).file.title;
        set(handles.finalRenderings, 'String', pop_up);
        set(handles.finalRenderings, 'Value', num);
        %num=num-1;
        endFigure(num).X=dataStack(currentStack).parameters.X;
        endFigure(num).Y=dataStack(currentStack).parameters.Y;
        endFigure(num).Z=dataStack(currentStack).parameters.Z;
        endFigure(num).V=dataStack(currentStack).Vr;
        endFigure(num).FV=dataStack(currentStack).parameters.FV;
        endFigure(num).p=dataStack(currentStack).parameters.p;
        endFigure(num).zThick=dataStack(currentStack).parameters.zThick;
        endFigure(num).transp=dataStack(currentStack).parameters.transp;
        endFigure(num).color3d=dataStack(currentStack).parameters.color3d;
        endFigure(num).vis=1;
        plotFinal(hObject, eventdata, handles);
    end
end

% --- Executes on selection change in finalRenderings.
function finalRenderings_Callback(hObject, eventdata, handles)
global num;
global endFigure;
num=get(handles.finalRenderings, 'Value');
    if endFigure(num).vis==1;
        set(handles.visible, 'Value',1);
    else
         set(handles.invisible, 'Value',1);
    end
    plotFinal(hObject, eventdata, handles);


function finalRenderings_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function plotFinal(hObject, eventdata, handles)
global endFigure;
global lightcam;
global pop_up;
axes(handles.axes2);
cla(handles.axes2);

pop_up=get(handles.finalRenderings, 'String');
numr=numel(pop_up);
    for i=1:numr
        if endFigure(i).vis==1
            hold on;
            endFigure(i).p = patch(endFigure(i).FV);
            isonormals(endFigure(i).X,endFigure(i).Y,endFigure(i).Z,endFigure(i).V, endFigure(i).p);
            set(endFigure(i).p,'FaceColor',endFigure(i).color3d,'EdgeColor','none');
            daspect([1 1 endFigure(i).zThick]);
            axis off;
            view(3);
            cameratoolbar('ResetSceneLight');
    light('style','local');
    light('visible','off');
    material metal;
    axis off;
            if lightcam.val==1
               camlight(-45,45);
               %set(gcf,'Renderer','zbuffer');
               lighting flat;
            else
               lighting none;
            end

                set(endFigure(i).p,'FaceAlpha',endFigure(i).transp);

        elseif endFigure(i).vis==0
            continue;
        end
    end


function defineVisibility(hObject, eventdata, handles)
global endFigure;
global num;
if isstruct(endFigure)
    rad_on=get(get(handles.uipanel3,'selectedobject'),'Tag');
    switch rad_on
    case 'visible'
    endFigure(num).vis=1;
    case 'invisible'
    endFigure(num).vis=0;
    end
    plotFinal(hObject, eventdata, handles)
else
    warndlg('Perform 3D rendering first!','warn')
end


% --- Executes on button press in visible.
function visible_Callback(hObject, eventdata, handles)
if get(handles.visible, 'Value')==0
    set(handles.visible, 'Value',1);
else
    defineVisibility(hObject, eventdata, handles);
end


% --- Executes on button press in invisible.
function invisible_Callback(hObject, eventdata, handles)
if get(handles.invisible, 'Value')==0
    set(handles.invisible, 'Value',1);
else
    defineVisibility(hObject, eventdata, handles);
end
```