Aus dem Deutschen Rheuma-Forschungszentrum
Ein Institut der Leibniz-Gemeinschaft
eingereicht über den Fachbereich Veterinärmedizin
der Freien Universität Berlin

Bioimage analysis linking information at protein and transcriptional level in tissues

Inaugural-Dissertation
zur Erlangung des Grades eines
Doctor of Philosophy (PhD)
in Biomedical Sciences
an der
Freien Universität Berlin

vorgelegt von
Ralf Köhler
aus Schkeuditz

Berlin 2023
Journal-Nr.: 4418

mbvberlin | mensch und buch verlag

Aus dem Deutschen Rheuma-Forschungszentrum
Ein Institut der Leibniz-Gemeinschaft
eingereicht über den Fachbereich Veterinärmedizin
der Freien Universität Berlin

Bioimage analysis linking information at protein and transcriptional level in tissues

Inaugural-Dissertation
zur Erlangung des Grades eines
Doctor of Philosophy (PhD)
in Biomedical Sciences
an der
Freien Universität Berlin

vorgelegt von
Ralf Köhler
aus Schkeuditz

Berlin 2023

Journal-Nr.: 4418

Gedruckt mit Genehmigung des Fachbereichs Veterinärmedizin

der Freien Universität Berlin

Dekan:	Univ.-Prof. Dr. Uwe Rösler
Erste Gutachterin:	Univ.-Prof. Dr. Raluca A. Niesner
Zweite Gutachterin:	Prof. Dr. Anja Erika Hauser
Dritter Gutachter:	Prof. Dr. Sigmar Stricker

Deskriptoren (nach CAB-Thesaurus):
tissues, transcription, gene expression, tissue protein, fluorescence, histology, biology, medical records

Tag der Promotion: 28.11.2023

Bibliografische Information der *Deutschen Nationalbibliothek*
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <https://dnb.de> abrufbar.

ISBN: 978-3-96729-236-7
Zugl.: Berlin, Freie Univ., Diss., 2023
Dissertation, Freie Universität Berlin
D188

 Choriner Str. 85 - 10119 Berlin
verlag@menschundbuch.de – www.menschundbuch.de

Contents

Figures

Tables

Abbreviations

2D	two-dimensional
3D	three-dimensional
CCD	charge coupled device
cDNA	complementary deoxyribonucleic acid
DFT	discrete Fourier transformation
DFT^{-1}, iDFT	inverse discrete Fourier transformation
DNA	deoxyribonucleic acid
ES	enrichment score
FACS	fluorescence-activated cell sorting
FFT	fast Fourier transform
FOV	field of view
FT	Fourier transform
FWHM	full width at half maximum
GSEA	gene set enrichment analysis
IF	Immunofluorescence
ILC	innate lymphoid cell
LSFM	lightsheet fluorescence microscope/microscopy
LSM	laser scanning microscope/microscopy
MELC	multi-epitope ligand cartography
mRNA	messenger ribonucleic acids
MSE	mean squared error
n	refractive index
NA	numerical aperture
PBS	phosphate-buffered saline
PC	principal component
PCA	principle component analysis
PLA	polyactide
PMT	photo multiplier tube
PSF	point spread function
PSNR	peak signal to noise ratio
RNA	ribonucleic acids
RNA-seq	ribonucleic acids sequencing
SARS-CoV-2	severe acute respiratory syndrome coronavirus 2
SBR	signal to background ratio
sCMOS	scientific complementary metal-oxide-semiconductor

SNE	Stochastic Neighborhood Embedding
SNN	shared nearest neighborhood
SNR	signal to noise ratio
ssGSEA	Single sample gene set enrichment analysis
ST	spatial transcriptomics
TIC	Toponome Image Cycler
t-SNE	t-distributed Stochastic Neighborhood Embedding
UMAP	Uniform Manifold Approximation and Projection
UMI	unique molecular identifier
λ	wavelength
σ	standard deviation
σ^2	Variance
τ	fluorescence lifetime

1. Introduction

Biological tissue samples contain unique and invaluable information that reflects an individual's entire history of lifestyle, development, or disease progression. Insight into the complex organization and structure of different cell types within these tissues can reveal the underlying mechanisms of functional biological processes and diseases. Various techniques, such as microscopy, flow cytometry or sequencing, provide access to this information, with each method specializing in certain parameters and aspects of cellular analysis.

In recent years, the fields of multiplex fluorescence microscopy and histocytometry have emerged with the aim of linking spatial structural information and multiparameter feature extraction with subsequent quantitative data analysis (Gerner et al., 2012; Gerstner et al., 2004; Giesen et al., 2014; Schapiro et al., 2017, 2018). These technologies enable a deeper understanding of biological and pathophysiological processes by considering that cells do not exist in isolation, but interact within cell groups or tissue microenvironments.

A powerful technique in this area is "multi-epitope ligand cartography" (MELC, Schubert et al., 2006, 2012), which allows the co-mapping of multiple proteins within the same tissue section. MELC facilitates phenotypic classification, the study of local and global cell communication, and the identification of disease-related changes in morphology while preserving sample integrity. However, to perform quantitative image analysis, it is critical to accurately detect and analyze individual cells within microscope images. This process is affected by system-specific and physical limitations, including resolution, signal strength, and image artifacts, which must be carefully considered and corrected for in the analysis pipelines, especially by the microscopy technique used.

To address these challenges, appropriate correction and identification algorithms must be developed that are tailored to the specific technology and imaging system. While fluorescence microscopy focuses on protein-level analysis, emerging methods such as spatial transcriptomics (Moses & Pachter, 2022; Ståhl et al., 2016) enable image-based analysis at the transcriptional level. Despite the differences in molecular targets, the conceptual steps of the analysis pipelines remain similar and include signal localization and classification.

By integrating advanced imaging techniques with robust quantitative image analysis methods, we can gain a deeper understanding of the spatial organization, communication networks, and molecular signatures within biological tissues. The development and application of automated quantitative image analysis pipelines is necessary to ensure standardized, comparable, and reproducible results and ultimately to advance our knowledge of complex biological processes.

1.1. Image based system biology

The basic components of living organisms are cells. They were discovered, introduced and called as such by Robert Hooke back in 1665, by looking through a candle illuminated compound microscope, observing plant structures. This was only possible by the previous work of Z. Janssen and his son H. Janssen, who combined several glasses to magnify structures and invented in this way the idea of microscopes and telescopes at the same time around the 17th century. With one of the modified microscope versions Hooke used, first drawings of the structures could be shared (Masters, 2008).

Over the past decades various techniques have been continuously developed and supported the functional investigation of cell organisms. Since the 20th century, sample information such as tissue structures could be inspected, displayed or resolved by cell-specific fluorescence staining or light stimulation. With the invention of performant computers, the door was open for systematic and reliable image processing.

1.1.1. Label free imaging – phase contrast microscopy

Image magnification in early days of microscopy allowed scientists to observe the microbiological components of tissue samples, blood donations or plant cells. The visibility of structures is limited to the fact that thin transparent samples have small differences in intensity and the resolution of human eye is limited as well. Based on sample architecture properties and related density differences, F. Zernike discovered and proposed principles of phase contrast microscopy in early 1930s. Here phase changes and scattered light differences caused by sample thickness and refractive indices are modulated into intensity variations, resulting in edge enhanced microscopic images (Zernike, 1942, 1955).

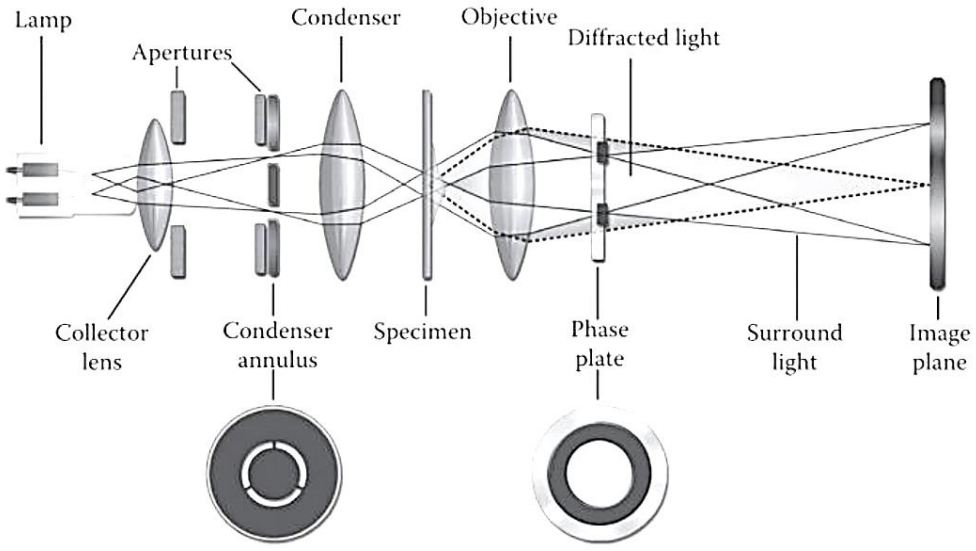

Figure 1 Overview of a phase contrast microscope setup and exemplary light paths. White regions in the condenser annulus and phase plate indicate transparent areas, dark regions absorbs light (Boas et al., 2016).

This change in phase is due to the fact that different components of the light wave (such as the electric field vector) experience different phase shifts as they pass through the anisotropic material without interaction with the molecules. Compared to a conventional microscope setup, a condenser annulus is placed between collector and condenser, the front focal plane (aperture) of the condenser, and a phase plate (quarter wavelength plate) is placed between the objective and imaging plane, the back focal plane of the imaging objective lens (Figure 1). In principle, condenser annulus exiting light illuminates the sample. Undeviated light is reduced in intensity by the absorbing material (grey ring) within the phase plate, in comparison to the diffracted light, which appears everywhere in the back focal plane. This is necessary to match both light amplitudes. Phase plate introduced light wave filtering in the back focal plane (Fourier plane) of the objective lens leads to interference at the image plane. Light passing through regions of the specimen that are out of phase with the background will interfere destructively, leading to reduced intensity and darker regions in the image. Conversely, regions of the specimen that are in phase with the background will interfere constructively, leading to increased intensity and brighter regions in the image. In this way the change in refractive index or samples thickness lead to image intensity differences and allows to enhance structural borders, edges respectively (Boas et al., 2016; Goodman, 2005).

Phase contrast microscopy as a non-invasive method of thin, transparent and translucent samples allows studying cell morphology or living cell dynamics over a long time period without special or complex preparation, which makes it fast and cheap.

1.1.2. Fluorescence-based cell identification

Fluorescence as a subcategory of Luminescence describes the process of spontaneous emission of photons longer wavelengths from atoms or molecules, in higher excited electronic states via electromagnetic waves of wavelengths in the range of 250 to 700 nm. This happens in average time of around 10 ns (vacuum), also called fluorescence lifetime (τ), described as the time between excitation and relaxation to ground state. The distance between excitation and emission wavelength maximum amplitude is described as Stokes' shift (Lakowicz, 2013). The process of excitation and emission of fluorescent light is limited in the amount of repetitive cycles and is fading in measurable energy amplitude over time, which is caused by the decreasing number of excitable molecules. It is called photo-bleaching and depends on the used fluorescent molecules (Im et al., 2019; Lichtman & Conchello, 2005).

Besides naturally occurring fluorophores (fluorescent substances), unlabeled tissue structures or cells can be made to fluoresce as well, as shown in the late 19th century by Paul Ehrlich (1854-1915) or the developer of Köhler illumination microscopy August Köhler (1866-1948) (Masters, 2008). This method is called immunofluorescence (IF), which at the time was only described as a different absorption of white light. This can be accomplished through a process

known as "fluorescent staining". Here, specific antibodies that bind to the proteins (epitopes) of the structure or cell of interest are labeled with a fluorochrome. If the used primary antibody cannot be tagged with fluorophore, an additional (secondary) fluorophore coupled antibody can be used. This process is called indirect IF in comparison to direct IF and is applied within fluorescence microscopy, flow cytometry or fluorescence-activated cell sorting (FACS) to amplify the signal. By visualizing different fluorescent antibodies or fluorescent reporter proteins that detect different epitopes in parallel, cell types can be identified.

1.1.3. Gene expression based cell identification

Covalently linked deoxyribonucleotide units create double-stranded deoxyribonucleic acid (DNA) and store the complete genetic information within each cell from generation to generation. DNA sequencing, introduced by Frederick Sanger (1918 - 2013) in 1977, enables the determination of the nucleotide sequence of DNA molecules, facilitating the study of genomic features and genetic variations. In contrast, RNA sequencing (RNA-seq) provides insights into gene expression by analyzing the actively transcribed regions of the DNA, which are transcribed into messenger ribonucleic acids (mRNA). The amount of mRNA is a measure of gene expression. RNA-seq has emerged as a cost-effective and time-efficient method for examining gene expression patterns, allowing for comprehensive analysis of the transcriptome.

Next generation sequencing, as one of the newer high throughput RNA-sequencing technologies, uses a collection of complementary DNA (cDNA) fragments created library from initial RNA conversion via an enzyme (reverse transcriptase), which are tagged by adaptors and amplified on a special slide, the flow cell. Iteratively fluorescent-labeled nucleotides are coupled and read within the sequencing machine and therefore give information on gene expression. (Alberts et al., 2014; Reinartz et al., 2002; Sanger et al., 1977).

The investigation of differentially expressed genes allows interpretation of biological processes during disease, treatment or other tissue sample related pre-handlings. Compared to bulk RNA-sequencing, where the differentially gene expression of all contributing cells of a whole tissue sample are averaged, single cell RNA-sequencing (single cell RNA-seq) allows cell type identification and has revolutionized the biology field (Tang et al., 2009). A common feature of both methods is the enzymatic dissociation or homogenization of the tissue sample into liquid-based cell suspension. Thereby, the spatial information is detached from the natural environment surrounding the cells and the extra cellular matrix (Olsen & Baryawno, 2018). This changed by development of spatial transcriptomics. Here, unique spatially barcoded oligo-nucleotides covering a glass slide capture mRNA of the tissue sample placed on top, enabling microscopic imaging before sequencing. Based on location identifiers, the gene expression can be back tracked to the tissue samples origin at a resolution of 55 µm. Specifically, 5000

circular areas with a diameter of 55 µm are homogeneously distributed over the field of view with a center-to-center distance of 100 µm (Moses & Pachter, 2022; Ståhl et al., 2016).

1.1.4. Digital image formation

The relation from objects viewed through microscopes is directly connected to the key component of these optical systems, the objective lens. Regardless of the microscope architecture, the main task of the objective lens is to magnify the observed structure by transfer of emitted light of every object point to the observer or a sensor.

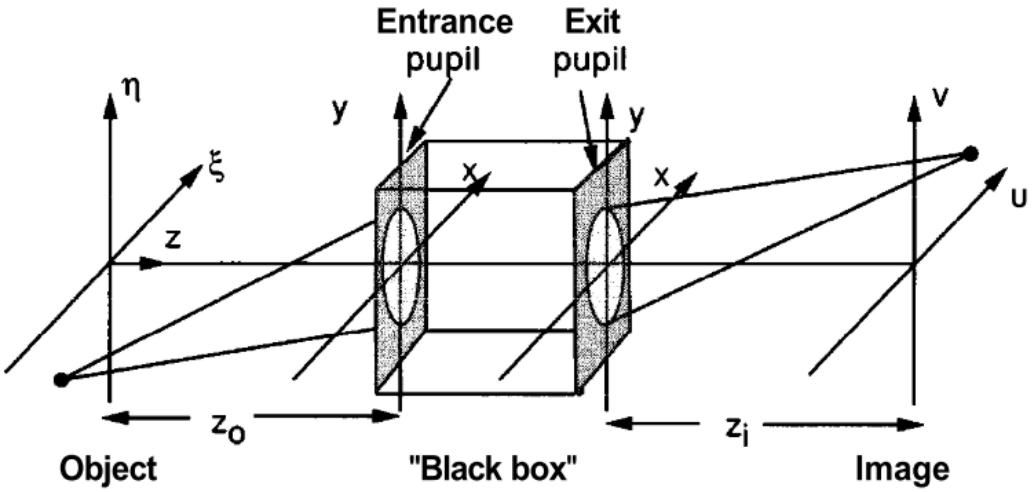

Figure 2 Overview of generalized optical system (Goodman, 2005)

Objective lenses typically consist of various single lenses focusing, defocusing and correcting light wave dependent properties such as chromatic and wave-front distortions, but for illustration of the light path transfer, the whole lens collection is taken as one system illustrated as “black box” (Figure 2), where following lenses for image creation in the eye of observer or camera are included. Finally, the recording system (image plane) only recognize the result intensity distribution of the exit pupil. Due to the finite size of the pupils, only a range of light rays can reach and release the optical system, this is why it is also called diffraction limited. As an effect higher frequencies (f) in space (analogues to wavelengths (λ)) are cut and lead to spatial resolution limit (S), where

$$S_{lateral} = \frac{1{,}22\lambda}{NA_{Objective} + NA_{Condenser}} \quad and\ S_{axial} = \frac{2\lambda}{NA^2} \quad with\ NA = n \cdot sin(\alpha), \qquad (1)$$

investigated by Abbe in 1873 (Abbe, 1873) or Lord Rayleigh in 1896 (S., 1896). Numerical aperture (NA) thereby defines the maximum angular range of entering light emitted of the object through refractive index (n) dependent medium. The Rayleigh criteria represented in equation (1) describes the minimum distance of two neighboring objects at which they can be

separable seen in case of fluorescence microscope. Separable at this point means the overlap of two airy disks, where the maximum intensity amplitude of the first object lays within the first minimum intensity amplitude of the second object. Additional to this formulation, the resolution limit can be interpreted as the full width at half maximum (FWHM) distance of a single objects intensity distribution in relation to its known size. Ideally, the real object (U_{real}) viewed through the optical system would be only a magnification (M) scaled version of the original object (U_O), so that

$$U_{real} = \frac{1}{|M|} U_O. \tag{2}$$

Due to diffraction limited system, light transfer by pupil area and optics introduce a convolution of real object image and transfer function

$$U_{image} = h * U_{real}. \tag{3}$$

Analogues to convolution, the operation above can be described as the product of two two-dimensional Fourier Transforms (FT), representing the FT property of optical systems, modifying shape of the input function, which can be formulated as

$$U_{image}(u,v) = \iint_{-\infty}^{\infty} h\left(u-\tilde{\xi}, v-\tilde{\eta}\right) U_{real}\left(\tilde{\xi},\tilde{\eta}\right) d\tilde{\xi} d\tilde{\eta} \;\; with\, \tilde{\xi} = M\xi;\; \tilde{\eta} = M\eta, \tag{4}$$

where the transfer function h is dependent of the signal amplitude (A), exit pupil (P)

$$h(u,v) = \frac{A}{\lambda z_i} \iint_{-\infty}^{\infty} P(x,y) e^{-j\frac{2\pi}{\lambda z_i}(ux+vy)} dxdy \tag{5}$$

and further represents the point spread function (PSF) of the optical system. Image aberrations or distortions introduced by the object itself or the optical system would result in a changed phase (additional exponent) of the transfer function (Goodman, 2005; HANSER et al., 2004). Finally, the objects emitted and transmitted light energy needs to be converted in the image plane, e.g. via a charge coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS) sensor. The approximate number of electrons

$$N = \delta A \delta t \int b(\lambda) q(\lambda)\, d\lambda \tag{6}$$

liberated in such a sensor is, next to the wavelength (λ), dependent of the pixel area (δA), exposure time (δt), incident photon flux (b) and quantum efficiency (q), where q defines the effective ratio of incoming photons and generated electrons. The finite size of the pixel array within the sensor lead to signal sampling and quantization of the continuous object function to discrete partitioned values (Gonzalez & Woods, 2002; Umbaugh, 2017).

1.2. Bioimage data analysis

Image acquisition of biological phenomena via microscopes, or expression measurements using sequencing alone describe only the starting point in hypothesis testing and require further expert-based interpretation of the data. In case of complex data (also termed “big data”), conclusion drawing is a time-demanding and subjective non-standardized process. In response to these challenges, computer driven digital image processing or related data analysis in general are used to support desired tasks, for example image correction, object identification or feature extraction. Thereby bioimage data analysis combines pre-existing algorithms and newly developed methods as integrated software tools or standalone computer programs. Their application involves extracting relevant information from the underlying image data and related biological samples. Besides the general idea of data analysis pipelines, each individual workflow created is unique to its components, tools and biological question or used devices, caused by raw data formation. In this way, results are standardized, comparable and ultimately contribute to the reliability of the interpretation (Miura & Sladoje, 2019). Therefore, the following section introduces the most important signal and data processing concepts.

1.2.1. Image signal processing and evaluation

1.2.1.1. Discrete Fourier transform

One of the most important tools in image processing is the Fourier transform (FT). It is used in many image analysis applications, such as image filtering, reconstruction, or compression, and has its origins in signal processing as telecommunications and electrical engineering developed.

A two dimensional digital image (*g(u,v)*) can be represented as a collection of one dimensional signal functions, sampled by pixel’s dimensions along the row or column arrays. The two dimensional discrete Fourier transformation (2D DFT), which is defined as

$$G(m,n) = \frac{1}{\sqrt{MN}} \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} g(u,v) \cdot e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}, \tag{7}$$

converts the finite sized image (MxN) as a set of periodic functions to frequency space image and therefore gives access to the image’s phase information with spectral coordinates *m=0,...,M-1* and *n=0,...,N-1*. Similar to the so called forward transformation, the back transformation (DFT^{-1}), or inverse DFT (iDFT)

$$g(u,v) = \frac{1}{\sqrt{MN}} \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G(m,n) \cdot e^{i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}, \tag{8}$$

recovers the main image signal in space domain with image coordinates *u=0,...,M-1* and *v=0,...,N-1*. These transformations mathematically describe analogously to 1.1.4, the spatial conversion from optical system's back focal plane (Fourier domain) to real image plane. The exponent within the sums represents a linear combination of single sinusoidal basis functions, where highest possible frequency is dependent on the sampling frequency, which is in this case the inverse image's pixel dimension. Any change in frequency, like in Fourier transformation image filtering, affects the signals reconstruction and leads to aliasing effects, caused by the pseudo periodic Fourier transformation of the finite sized image or if Nyquist-Shannon sampling theorem is not fulfilled. Additionally, a modification of frequencies in the Fourier domain adds artificial signals and therefore can cause artifacts after back transformation if the filter parameters are not adjusted accordingly. An optimized algorithm called "fast Fourier transform" (FFT) reduces the computation time of every single DFT from order n^2 to $n(log(n))$, where n is the number of data points (Burger & Burge, 2016).

Besides phase information access, DFT and FFT can be used within correlation analysis. By correlation of two signals, or in this case images (f(u,v),g(u,v)), the operation is called cross correlation (r) and is defined as

$$r_{fg}(u_0, v_0) = \sum_{u,v} f(u,v) g^*(u - u_0, v - v_0), \tag{9}$$

where (*) is equivalent to complex conjugate (inversion of the sign of the imaginary part). The same calculation can be performed by convolution, so that

$$r_{fg}(u_0, v_0) = \sum_{m,n} F(m,n) G^*(m,n) exp\left[i2\pi\left(\frac{mu_0}{M} + \frac{nv_0}{N}\right)\right]. \tag{10}$$

The uppercase letters indicate the Fourier transformation of the lower case letters. The method allows to find the displacement of two shifted images to each other. This requires inverse Fourier transforming the cross-correlation product, followed by locating the maximum peak, which becomes the centered offset position and is identical to the displacement. An introduced upscaling factor of the FFT images enable subpixel accuracy of the calculated shifts (Anuta, 1970; Guizar-Sicairos et al., 2008).

In addition to whole image transformation by DFT for image phase information access or image registration, another application is to describe closed curves by Fourier coefficients.

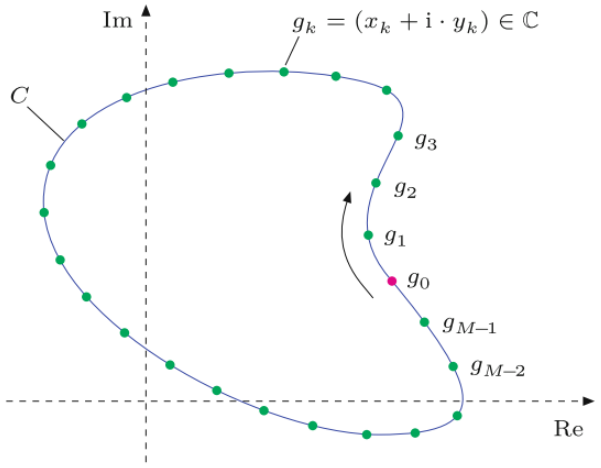

Figure 3 Object outline (C) sampled by equidistant points g_K in two dimensional complex plane (Burger & Burge, 2016)

These closed curves could be analogously represented by object outlines inside an image, as shown as single object graph in Figure 3. Here, the discrete valued curve is interpreted as a vector, which contain a collection of complex points in a two dimensional plane, where x_k represents the real part and y_k the imaginary part of every vector g_k. The DFT in this case is defined as

$$G_m = \frac{1}{M} \cdot \sum_{k=0}^{M-1} g_k \cdot e^{-i2\pi m \cdot \frac{k}{M}} = \frac{1}{M} \cdot \sum_{k=0}^{M-1} g_k \cdot e^{-i\omega_m \cdot \frac{k}{M}} \tag{11}$$

$$= \frac{1}{M} \cdot \sum_{k=0}^{M-1} [x_k + i \cdot y_k] \cdot \left[\cos(\omega_m \frac{k}{M} - i \cdot \sin(\omega_m \frac{k}{M}\right], \tag{12}$$

with angular frequency $\omega_m = 2\pi m$ and $0 \leq m \leq M$. The result out of the multiplication $G_m = A_m + iB_m$, creates the real (Re) and imaginary (Im) Fourier shape descriptors, or also called spectral coefficients

$$\mathrm{Re}(G_m) = \frac{1}{M} \sum_{k=0}^{M-1} \left[x_k \cdot \cos\left(\omega_m \frac{k}{M}\right) + y_k \cdot \sin\left(\omega_m \frac{k}{M}\right)\right], \tag{13}$$

$$\mathrm{Im}(G_m) = \frac{1}{M} \sum_{k=0}^{M-1} \left[y_k \cdot \cos\left(\omega_m \frac{k}{M}\right) - x_k \cdot \sin\left(\omega_m \frac{k}{M}\right)\right], \tag{14}$$

as long M is bigger than 1. The FFT can be used for calculation as well, but requires signal length $M = 2n$ for $n \in N$. Visually, each spectral coefficient defines a circle with decreasing radius dependent of the index frequency ω_m and thereby follows the outline's form. This mathematically tool allows to describe and recover complex shaped curves or image objects with a limited set of numbers (Burger & Burge, 2016; Gonzalez & Woods, 2002).

Altogether, DFT is an important algorithm used within many image analysis applications and developed FFT accelerates calculation speed by preservation of same results.

1.2.1.2. Linear regression analysis

The investigation of relationship between independent multivariate measurements or observations and their prediction of unknown dependent variables or data points is called regression. Its application differs from the independent input variables (covariate/regressor) and defines the output (response variables). There are three common use cases, which fall into linear regression. Data fitting involves finding the line of best fit that describes the global relationship between the variables. Interpolation involves estimating the values of the dependent variable for given values of the independent variable that falls within the range of the data points. Extrapolation involves estimating the values of the dependent variables for values of the independent variables that fall outside the range of the data points. This means that they differ in the mathematical function used, which is applied to a specific range of data to model the output.

In simple linear regression (data fitting) the relationship of only one set of response variables *Y* dependent of input variables *X* is inspected, where the linear model follows the definition

$$Y = \alpha + \beta X + e. \tag{15}$$

Assuming a straight (regression) trend line through the data points and existence of *n* observation pairs this can be written as

$$y_i = \alpha + \beta x_i + e_i, \tag{16}$$

where e_i represents an error and can be seen as difference between real value and its prediction. In addition α and β are called the regression coefficients, more precisely intercept term and slope parameter.

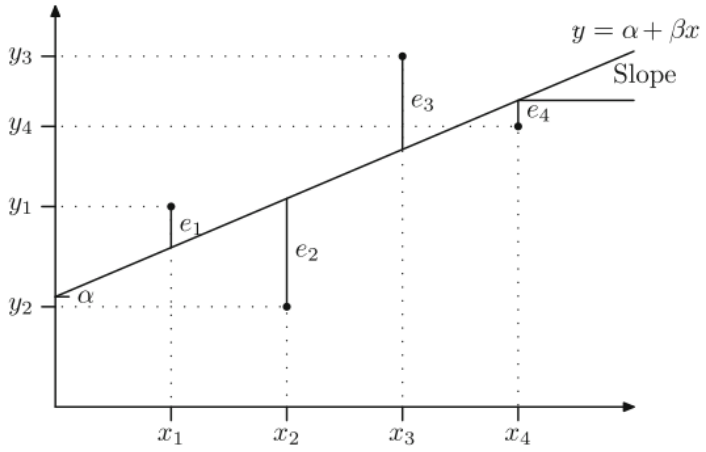

Figure 4 scatterplot of observation points with regression line, including error deviation (Heumann et al., 2016)

Based on the linear model, the minimization of all error terms leads to final fit of regression line, as illustrated in Figure 4 and further be formulated as

$$\min_{\alpha,\beta} \sum_{i=1}^{n} e_i^2 = \min_{\alpha,\beta} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2, \tag{17}$$

which analogously describe the method of least squares. The estimation of regression coefficients in case of two parameters can be calculated by

$$\hat{\beta} = \frac{S_{xy}}{Sxx} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - \bar{x}n^2} \;;\; \hat{\alpha} = \bar{y} - \hat{\beta}\bar{x} \tag{18}$$

and create the final prediction function for any value out of these data points

$$\widehat{y_i} = \hat{\alpha} + \hat{\beta} x_i. \tag{19}$$

Errors $\widehat{e_i}$ out of the prediction value in comparison to the measurement points are also called residuals. Throughout variance decomposition, the quality of found model is quantifiable using the R² criteria:

$$R^2 = \frac{SQ_{Regression}}{SQ_{Total}} = \frac{\sum_{i=1}^{n} (\widehat{y_i} - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} = 1 - \frac{SQ_{Error}}{SQ_{Total}} = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \tag{20}$$

in range $0 \leq R^2 \leq 1$. For values of R² towards one, the fitting model represents to higher degree the given values, predicted values as well and vice versa for R² closer to zero.

When working with multiple covariates, the linear regression model can be extended to their weighted linear combination:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + e. \tag{21}$$

In contrast to the equation (*15*, the intercept term is β_0. Based on p covariates and n data points, a set of n equations defines from now on the model:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} + e_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_p x_{2p} + e_2 \\ &\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_p x_{np} + e_n \end{aligned} \tag{22}$$

The introduction of matrix notation allows to compress all of the expressions to

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{e} \tag{23}$$

Where bold letters are equivalent matrices and vectors, such that

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}.$$

Within X the column of ones represents the intercept term at all observations. Via least squared method, minimization of the error terms, the estimation of β calculates to

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}. \quad (24)$$

Thereby $\widehat{\boldsymbol{\beta}}$ includes estimates of the all covariates. A continues update during minimization process of the error terms and related residuals results in a fitted model along the parameter space until the model converges or satisfy a certain limit.

Application of regression analysis is not limited to physically real variables. The introduction of dummy variables makes it possible to work with categorical parameters as well and is used in classification. Another field, nonlinear regression, covers data fitting related to known distribution functions. Typically in natural and life sciences, the parameters are distributed following Gaussian functions – therefore Gaussian fitting is key. Here, the initial step consists of linearization before calculating the parameter estimates. Linearization can be realized through mathematical transformations or Taylor series decomposition. However, not all nonlinear regression problems can be solved by support of linearization. Based on the specific nature of the nonlinear relationship, other techniques may be more appropriate (Fahrmeir et al., 2016; Heumann et al., 2016).

1.2.1.3. Decision trees and random forests

Evaluation of image signals also refers to pixel or object classification. Based on certain criteria or rules, numerical and categorical variables are assigned to certain group(s) or class(es), respectively. In the simplest case of a single limit (threshold), binarization of the input variables takes place and ends in two classes. In case of multiple available input parameters, a sequence of binary separations can be realized to refine the final classification, where every outcome of the previous binarization influences the following step.

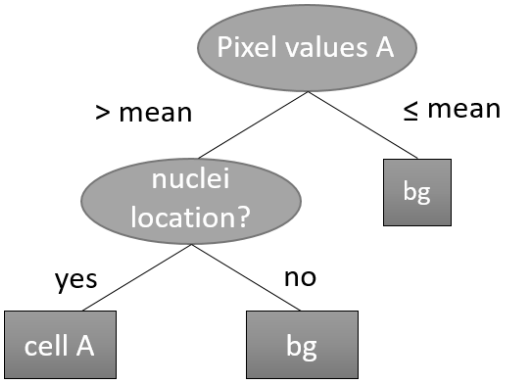

Figure 5 Decision tree construction example to classify image pixels of image A to be cell class A or background (bg), related to presence of nuclei signal at same location

Altogether the whole structure can visually represented as a flow chart (Figure 5), which is better known as decision tree. The first node on top is called the “root node”, where ending nodes are called “terminal nodes” or “leafs”, reflecting the final classes. Inner nodes are equivalent to feature attributes. Commonly, branches which do not fulfill a condition are placed on the right side.

Construction of decision trees follows a recursive process, finding the best split-point of the data at each feature attribute, where every nonterminal point separates the previous outcome data (whole data set at the root node) into two subgroups, those that meet the criterion and those that don't. One of the metrics for describing the information content of a data set, which is also known from thermodynamics, is the entropy

$$H(p) = H(p_1, \dots, p_n) = -\sum_{i=1}^{n} p_i \log_2(p_i), \tag{25}$$

where p denotes the n class probability distribution, equivalent to the amount of data points within each class. Taking together all classes' probabilities sums up to a value of one. For probabilities equal to zero, the entropy is defined to be zero. Out of the entropy the information content of a data set (D) is defined as

$$I(D) = 1 - H(p) = 1 - H(D). \tag{26}$$

Based on the global and individual information content, the information gain (G) of every single attribute (A) related to the whole data set can be calculated:

$$G(D, A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} I(D_i) - I(D). \tag{27}$$

The attribute at highest information gain is the one to be taken for the node, dividing the data set. In case of continuous attributes a similar principle is applied within the value set to get a final threshold

$$\theta_{D,A} = \mathrm{argmax}_v\{G(D, A > v)\}. \tag{28}$$

For every value (v) the information gain is calculated. Thereby, "argmax" returns the argument (input value) that maximizes a given function. The threshold value arising in the highest information gain is taken. Introduction of "mean squared residual" as splitting criteria of a given data set enables decision-tree based regression as well. In context of classification, other metrics, such as "misclassification error" or "Gini index" were also used for split-point calculation. The subsets, separated by category or threshold, now go through the same procedure until no further subdivision is possible, or to a defined end point (number of elements within one group). The created decision tree structure fits the underlying data. Minimizing the risk of overfitting and increasing the model prediction of unknown data points, the original data set is split into training and test data set, where test data set is unseen from the model during creation (training). The proportion of right classifications enables model quality check. A partial reduction of trees (pruning) can increase the global prediction accuracy (Ertel, 2011; Feigenbaum & Simon, 1961; Hastie et al., 2009; Quinlan, 1986).

In contrast to single decision-tree based classification or regression, Random Forest uses a whole set of de-correlated trees, which averaged results reduces single model variance. Thereby, each individual tree (h_j) is built out of a subset (Θ) of the original data set ($D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$). During construction only a few predictor variables (p) are used to calculate the split-point, typically this number m is the square root of p and at least one. Data sets subset (D_j) selection follows a random distribution and is called bootstraping. Via bagging (bootstrap aggregating) each of the tree than is fitted to its data set selection and related p. Out of all fitted trees ($\hat{h}$), the prediction function in case of regression results in

$$\hat{f}(x) = \frac{1}{J}\sum_{j=1}^{J} \hat{h}_j(x), \tag{29}$$

where J denotes the number of trees. In case of classification the prediction function belonging to a certain class y follows

$$\hat{f}(x) = \text{argmax}_y \sum_{j=1}^{J} I(\hat{h_j}(x) = y). \tag{30}$$

Due to bootstraping, some of the samples of the original data set are not used. This untouched subset is called “Out-Of-Bag Data” and is taken to calculate the model’s related error, which in case of regression is defined by out-of-bag means squared error (MSE_{oob})

$$MSE_{oob} = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{f}_{oob}(x_i))^2 \tag{31}$$

and in case of classification, the generalized out-of-bag error rate (E_{oob})

$$E_{oob} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq \hat{f}_{oob}(x_i)). \tag{32}$$

In comparison to averaged errors, in classification a separated calculation of the error rate at each wanted class, enables distinct observations of the final predictions and their accuracy. Within a “confusion matrix” all of the single error rates can be summarized via cross-tabulating the out-of-bag predictions and their known values. Adjusting the individual trees by their number or size is referred to tuning and can be used to get better predictions. A weighting of unbalanced class occurrences also can refine the global prediction performance (Breiman, 2001; Cutler et al., 2012; Hastie et al., 2009).

Application of single decision-trees or random forest follows a simple model creation process, but giving a powerful multipurpose tool. Especially in multiclass classification, the interpretability of the outcome predictions support understanding of underlying data structure and their relations.

1.2.2. Statistical analysis of objects

1.2.2.1. Dimensionality reduction

The increasing number of different variables inevitably leads to an increase in the complexity of the respective data set. Among data correlation or prediction, visualization of such relations is limited by the number of variables. Reduction of these multidimensional feature spaces, by preserving global or local dependencies is the goal of dimensionality reduction methods. Additionally, these representations supports the decrease of computationally expensive tasks, taking them as input variables.

Principal component analysis (PCA) is one of the oldest algorithms (F.R.S., 1901; Hotelling, 1933), which estimates the orientation of the individual feature attributes with the highest variance and projects the principal components (PCs) thus found into new subspaces created by each individual computed PC (Ertel, 2011). Each of the PCs are orthogonal to each other, which means that the dot product between two PCs is zero, confirming that they are independent of each other. Correlated feature attributes are represented in similar regions and in this way create a representation of the data with low dimensionality. Estimation of orientation follows the same concept as in linear regression analysis, where the sum of squared residuals of a line fitting the data is minimized. In contrast to the low-dimensional representation subspaces created in PCA, other methods such as “t-distributed Stochastic Neighborhood Embedding” (t-SNE) or “Uniform Manifold Approximation and Projection” (UMAP) produce only one final map of the associated connectivity.

The t-SNE algorithm is based on the previously developed Stochastic Neighborhood Embedding (SNE) and differs mainly in the distribution function used and the symmetrized cost function, but is conceptually the same. It converts similarities, calculated by Euclidean distances between all data points (x), to joint probabilities, then followed by minimization of the Kullback-Leibler (KL) divergence between the joint probabilities of the low-dimensional space (embedding, q(y)) and the original high-dimensional space (p(x)). Finally the cost function (C) can be described by

$$C = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$= \sum_i \sum_j \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right)} \log \left(\frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}{\sum_{k \neq l} \exp\left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right)} \Big/ \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \right). \quad (33)$$

The variance (σ) of the joint probability distribution function is related to the perplexity parameter k, the effective number of neighboring data points. Besides the manual choice of k, σ should equalize the distribution functions entropy to $\log_2(k)$. Minimization of C is performed via gradient descent

$$\Upsilon^{(t)} = \Upsilon^{(t-1)} + \eta \frac{\delta C}{d\Upsilon} + \alpha(t)\left(\Upsilon^{(t-1)} - \Upsilon^{(t-2)}\right), \tag{34}$$

where $\Upsilon^{(t)}$ denotes the result at iteration t, η denotes the learning rate and α(t) defines the momentum at iteration t. Here the previous solution of the loss function and momentum optimizes the minimization process. The resulting low-dimensionality embedding is upon random initialization of gradient descent and random variable selection different at each calculation. Therefore, variable relations stay the same, however representation of the final embedding looks different. In some cases gradient descent tends to stuck in local minimum rather finding the global minimum (Hinton & Roweis, 2003; van der Maaten & Hinton, 2008).
The alternative method for dimensionality reduction, which is also frequently used, is UMAP. The working principle follows the same two-stage scheme as for t-SNE. First, conversion from high-dimensional space into similarity measures, in this case realized by construction of weighted k-neighbor graph (fuzzy simplical complex), and second their embedding within a low-dimensional space via stochastic gradient descent. Thereby, hyper parameters such as number of local neighbors during construction of the graph and the scaled distance (spread) in the low-dimensional space embedding are taken by manual choice, giving more flexibility of visualization. The advantage of UMAP lies in the fast calculation time for large data sets by preserving balance of global data structure and local connectivity (McInnes et al., 2018).

1.2.2.2. Clustering

Partitioning of unlabeled data into similarity-based data point groups is considered to be cluster analysis. Similarity is a measure of distance metrics, similar to dimensionality reduction methods. In contrast to distances between neighboring features attributes, the distances of different partitioned groups (cluster) are bigger and thus define the required criterion. Additionally, each variable should have only one assigned cluster, allowing no overlap of the found clusters.
A wide spread and simple clustering algorithm is k-means (MacQueen, 1967), where k indicates the pre-defined number of groups to partition the data. The working principle in brief is described in the following. With k random selected centroid points the first clusters positions are initialized, followed by calculation of the distances of all the data points to the cluster midpoints. The minimal distance to a cluster midpoint then defines the cluster label. Based on all assigned data points, the centroid positions of clusters are recalculated. If the location of

the previous cluster centroid is different from the current, the position is shifted, followed by recalculation of data points distances and cluster assignment. The process of repositioning and relabeling is repeated until no further change of calculated cluster midpoint location is reached. Normalizing the feature attributes avoids distortion of the distances to the desired clusters. Due to random initialization, a pre-investigation of different features can enhance better probable cluster initialization position or even the number of clusters needed.
In hierarchical clustering (Narasimha Murty & Krishna, 1980) a pre-definition of the number of clusters is not required. Here, all data points are initially assigned to individual clusters and then sequentially combined based on a similarity metric until a termination point is reached or the last cluster contains all data points. Visually, this method resembles a binary tree. Application of a distance metric, in which the minimal distance between two points of two clusters is calculated, leads to “nearest neighbor algorithm”.
The "correct" number of clusters to use depends on the data itself and the underlying question. Introducing quality measures for each cluster (e.g. “silhouette width criterion”) only estimates the proportions of distances and does not meet user preference. By exploring the relationships between features (variables) and attributes (clusters), the decomposition of partitioned data allows the rediscovery of meaningful patterns and structures within the data. This approach enhances the interpretation and usability of clustering results, enabling deeper insights and a more refined understanding of the data (Ertel, 2011; Kubat, 2017).

1.3. Aims of this work

In recent years, there has been a growing demand for automated quantitative analysis of (image) data in various scientific fields. This need is driven by the desire to ensure standardized, comparable, and reproducible results and to take advantage of advances in hardware technology and software processing. Existing techniques have laid the foundation for such automated analysis pipelines, but further development and application is required to address specific challenges and improve overall system performance.
One area of particular interest is spatial analysis, which involves the exploration and interpretation of spatial patterns and relationships within biological samples. Spatial information plays a critical role in understanding complex biological systems by providing valuable insights into the organization, interactions, and dynamics of cells and molecules within tissues. By integrating spatial analysis into automated quantitative analysis pipelines, we can gain a deeper understanding of biological processes and reveal spatially distinct phenotypes.
In addition, a critical aspect to consider is the physical imaging process and the associated artifacts that can be affected. These artifacts can be introduced by various factors such as the imaging system itself, sample preparation, and the interaction of the light with the biological

tissue. Understanding and mitigating these artifacts is essential to obtaining accurate and reliable results in automated quantitative analysis.

The aims of this thesis, based on existing methods and techniques:

- Improvement of the existing MELC system, including
    - Image quality (signal to noise (SNR) enhancement, resolution)
    - Extension of the detection of fluorescently labeled antibodies with regard to the unused spectral range
- Hardware dependent artifact removal, by application of machine learning algorithms in relation to physically limited image formation
- Property-related object identification within microscopy images, independent of device architecture
- Multi-parameter correlation and phenotypic classification at protein and transcriptional level by visualization of complex data in a compressed manner

2. Material and Methods

Spatial quantitated image analysis consist of three major tasks: acquiring the image data, pre-process the data including feature extraction and followed functional correlation analysis to validate biological hypotheses. As part of these steps, devices, feature extraction methods and used software or data analysis environments relevant for the present thesis are explained.

2.1. Microscopes

Based on the tissue properties and related resolution required to observe distinct biological processes, the choice of microscope is essential. In this section, microscopy techniques used to obtain the image data analyzed within this work, is explained.

2.1.1. MELC

The core of the multi epitope ligand cartography (MELC) system is an inverted widefield (epi)fluorescence phase contrast microscope (Leica DM IRE2) combined with a pipetting robot, based on a reconfigured/modified Toponome Image Cycler MM3 (TIC) manufactured by MelTec GmbH & Co.KG Magdeburg, Germany (Schubert et al., 2006).

Figure 6 Overview of MELC system (Toponome Image Cycler MM3 (TIC)) and explanation of main parts

The basic working principle of system's (Figure 6) inverted widefield (epi)fluorescence phase contrast microscope in short. An inverted microscope with the objective lens and related optics under the sample enables access to the sample from top, which in case of the MELC system allows for pipetting and suction via the pipetting robot. Furthermore, widefield configuration means illumination of the entire sample via a light source. An (epi)fluorescence microscope setup means, that the light path for fluorescence excitation and the detection of the emitted light is the same. Phase contrast configuration allows for the observation of unstained samples under the microscope. The different phase components of (transmission) light passing through the sample are converted in amplitude differences or intensities. A varying phase is caused by

light-dependent transit time, which is dependent on samples' refractive index architecture. Condenser annulus aperture in front focal plane and phase plate in the back focal plane limiting the angle of the incoming light beam and shift the phase of the diffracted light. As a function of the shift, interference takes place and results in intensity translated differences, requiring Köhler illumination setup, which means a centered alignment of all optical components (apertures, field diaphragms) to create homogenous illumination over the entire sample.

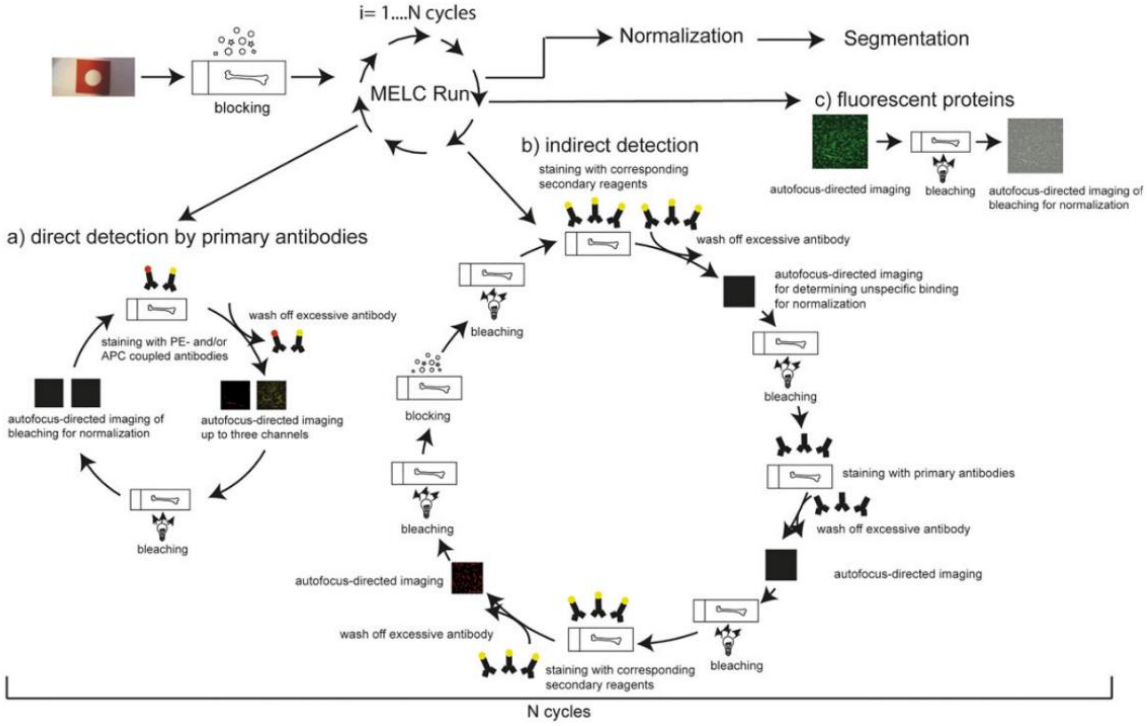

Figure 7 overview of cyclic MELC run workflow taken from Figure 1 in (Holzwarth, Köhler et. al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

Theoretically, during every MELC run or MELC experiment an unlimited number of fluorescently labeled antibodies can be acquired in up to four different field of views (FOVs). Thereby fully software (MelTec TIC-Control) automated driven imaging procedure is sequentially performed, consisting of four major cycled steps (Figure 7):

1) Loading of cooled fluorescently labelled antibodies (up to four at the same time) from a 96-Well plate via the pipetting robot, followed by the application of the antibodies on to the tissue sample, where incubation, subsequent washing and liquid level adjustment takes place
2) Autofocusing of tissue sample's preset FOV based on cross-correlation and following imaging. Here a phase contrast image acquired by user's selected FOV at the beginning is used as a reference image and compared with current images to find the same sample location. If necessary, the motorized microscope moves the sample until maximum correlation value has been reached. At first phase contrast images are

acquired along the axial axis, then fluorescence images of the desired fluorescence channels are acquired.

3) Signal removal by time controlled photo-bleaching is combined with washing to keep a constant liquid level, to avoid drying of the sample and to remove free radicals.
4) Another round of autofocusing and imaging takes place. Besides phase contrast images, photo-bleached fluorescence images are acquired as well.

All steps and general imaging related settings, such as incubation time, channel information, antibody names and their location within the 96-Well plate, etc. are included in the user's created MELC run initialization file (*.xml) which is read by the machine and is used for downstream analysis. In addition to phase contrast images, fluorescence signal can also be used for autofocusing. Actually, phase contrast images offer a staining independent signal, since they are created only on tissue sample's heterogeneous refractive index distribution, which remains stable over long experiments.

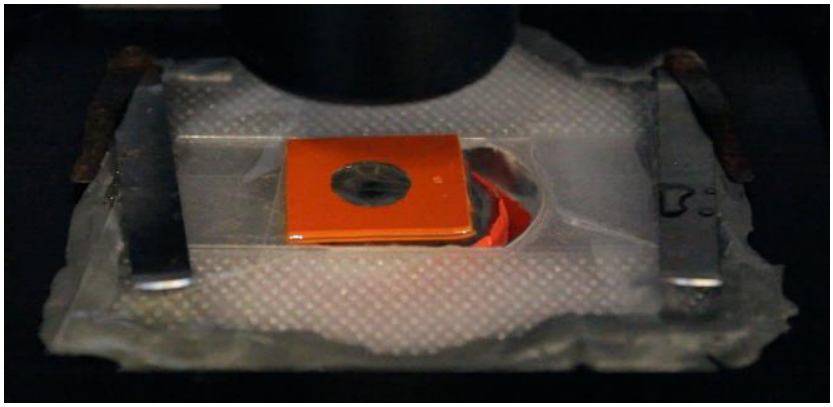

Figure 8 tissue sample under microscope of MELC system

In order to perform MELC experiments tissue samples had to be prepared in a special way. Five to ten micrometer thick fresh frozen cut tissue samples were placed on or glued on a cover slide (24 × 60 mm #1; Menzel-Gläser, Braunschweig, Germany). The sections were fixed and blocked, according to the sample's origin (Holzwarth et al., 2018; Mothes et al., 2023; Pascual-Reguant et al., 2021). Then a liquid reservoir made of one-millimeter-thick silicone sheets was created containing a volume of 100 µl (Figure 8), which was prefilled with phosphate-buffered saline (PBS). The silicon reservoir prevented the liquid from running off, and the section from drying. In addition, the pipetting robot could accurately pipette the required fluorescence-labeled antibodies for staining or PBS for washing onto the tissue section and remove the excess liquid. Excess liquid from the aspiration was discarded into the wash box positioned around the sample at the beginning of the automated run.

2.1.2. LSM

Compared to a widefield microscope, a laser scanning microscope (LSM) creates an image by measuring laser excited fluorescence emission light along a planar grid of measuring points in a stepwise manner. Based on the size of the focal volume (PSF), the step width (lateral and axial) is adjusted. These are both dependent of the magnification of used objective lens. Light is detected by photo multiplier tubes (PMTs), at the end every measured point the photon count is converted to intensity values along the image's pixel grid. Measuring at different depth (often called image stacking) allows one to acquire three-dimensional image information. The microscope used (Zeiss LSM 710 / LSM 880) is a confocal configuration, which means that only in focus light passes through a motorized size variable pinhole. Out of focus light is blocked and therefore the system results in higher resolved images, equivalent the PSF is smaller in the LSM compared to widefield microscopes. In case of the mouse model for vascular dementia, mice were perfused with 4% paraformaldehyde. The brains were dissected and embedded in Tissue Tek (Sakura), frozen in a methylbutane/dry ice mixture and cut into 10- or 30-µm sections using a cryostat. Sections were stained with goat anti-EGFP (Rockland, conjugated at the DRFZ to Alexa®488), mouse anti-GFAP-Alexa®488 (eBioscience, Germany), rabbit anti-Noxo-1 (Novus Biologicals, Germany), or goat anti-p47 (Abcam, Germany). Secondary antibodies used were donkey anti-rabbit Alexa®647 (Life Technologies, Germany) or donkey anti-goat-Alexa®647 (Radbruch et al., 2017).

2.1.3. LSFM

In contrast to the previous microscopy techniques, light-sheet fluorescence microscopy (LSFM) excites the samples from the side and detects the emitted light from the top, so illumination and detection axes are perpendicular to each other – a beam-path geometry well known from fluorescence spectrometers. The illumination is performed in a wide-field manner, due to the use of cylindrical lenses with low NA, in shape of a single plane – light sheet. One or several light sheets, from one or multiple laser beams, differently oriented to each other, form the final excitation light sheet. The goal is to achieve a uniform illumination throughout the sample. Because of the two crossing, orthogonal PSFs, i.e. excitation PSF and detection PSF, out of focus regions from the used objective lens in the detection part do not contribute to signal. Photo-toxicity and photo-bleaching of the sample is reduced, due to single plane illumination.

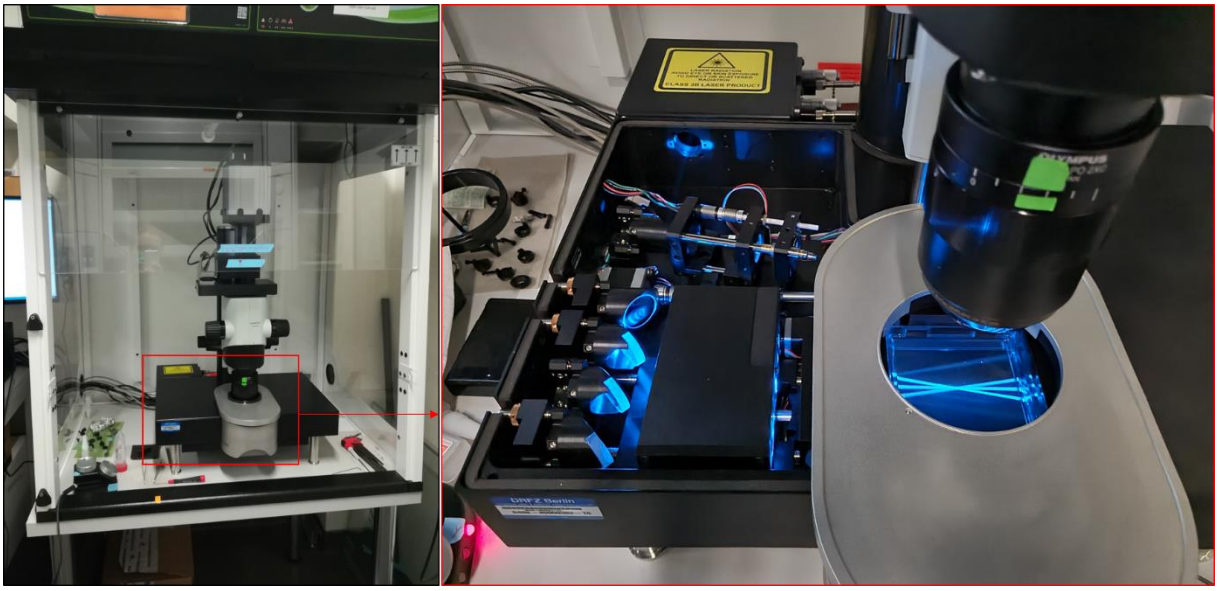

Figure 9 Overview of Ultra II light sheet microscope (LaVision Biotec, Germany), red boxed area shows illumination function creating the light sheet

Tissue clearing and sample preparation of murine femoral and tibial bones were processed with an adapted tissue clearing protocol, specifically optimized for rendering hard mineralized tissue optically transparent. In brief, the harvested organs were thoroughly fixed with a flexible polyepoxide scaffold, and in subsequent chemical steps, decalcified, delipidated and decolorized. Followed by a staining step with a fluorescent conjugated nanobody against the endogenous reporter fluorescence, the refractive index of the remaining peptide structure was matched with a fitting solution. This made the samples optically transparent while preserving endogenous reporter fluorescence. Combined with signal enhancement, this provided a sample suitable for deep bone marrow imaging.

During image acquisition, the transparent samples, mounted in small cuvettes, were surrounded by the refractive index matching solution. This cuvette was in turn immersed in the sample chamber of the Ultra II light sheet microscope (LaVision Biotec, Germany, Figure 9) filled with immersion oil matching the refractive indices of the contents of the sample cuvette. Images were acquired slide by slide, scanning through the entire sample depth, in axial steps of 3 to 10 µm with a horizontal laser light sheet illumination. The following excitation/emission wavelengths for the different fluorophores were used: 488/525(50) nm; 561/620(60) nm; 640/680(30) nm. The light sheet width was set to a minimum of 4 µm, and the NA was set to its highest configuration 0.135. The zoom body setup with an Olympus MVPLAPO 2x/NA 0.5 objective was used for stepwise magnification of 0.63x to 6.3x equipped with an optical abberation corrected dipping cap. For image acquisition a 5.5 Megapixel sCMOS camera was used.

2.2. ST

Spatial transcriptomics (ST) is a method enabling the visualization and localization of gene expression within histological tissue sections, by combining microscope image acquisition and sequencing. It is based on unique spatially barcoded oligo nucleotides, which capture mRNA of the tissue sample.

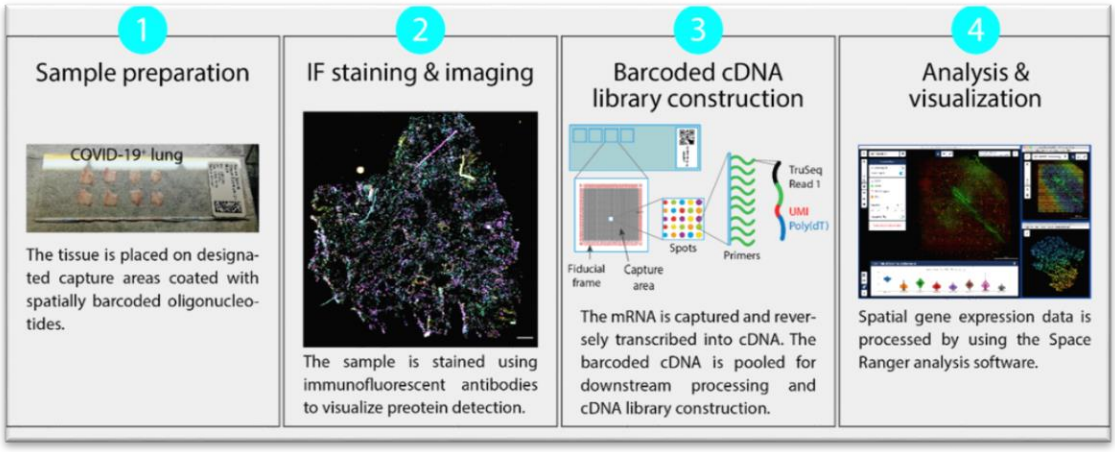

Figure 10 overview of ST sample preparation, image acquisition and gene expression extraction

Sample preparation was performed following the manufacturers protocol (10x Genomics, Figure 10). Briefly, samples were snap frozen in an isopentane bath on liquid nitrogen/dry ice. 10 µm frozen sections were cut into on a MH560 cryotome (ThermoFisher, Waltham, Massachusetts, USA), and placed on a pre-cooled (-20°C) 10X Visium slide. The tissue sections were fixed in pre-chilled methanol for 30 minutes, stained with CD45-AF647, CD31-AF594 and DAPI for 30 minutes, imaged using a LSM 880 confocal microscope (Zeiss). After imaging, a 10 minutes permeabilizing step prepares spatially tagged cDNA libraries construction using the 10x Genomics Visium Spatial Gene Expression 3' Library Construction V1 Kit. cDNA libraries were sequenced on an Illumina NextSeq 500/550 using 150 cycle high output kits with sequencing depth of ~5000 reads per spot (Mothes et al., 2023).

2.3. Image analysis software and algorithms

In this section, open-source software and algorithms referring to image segmentation or object identification are presented.

2.3.1. Watershed segmentation - Cell Profiler

A classical approach for image segmentation is to classify pixels based on their intensity. At a certain signal limit (threshold), pixels are set to foreground or background, resulting in a binarized image, in which foreground pixels define areas of objects in the image. Followed connected component analysis then tries to find the single foreground objects, giving each point cloud an individual identifier. Sometimes objects are densely packed, or objects with

similar intensities are very close to each other, e.g. stained nuclei within the image. In this case a separation of clumped objects is required, in order to avoid counting the wrong number of objects.

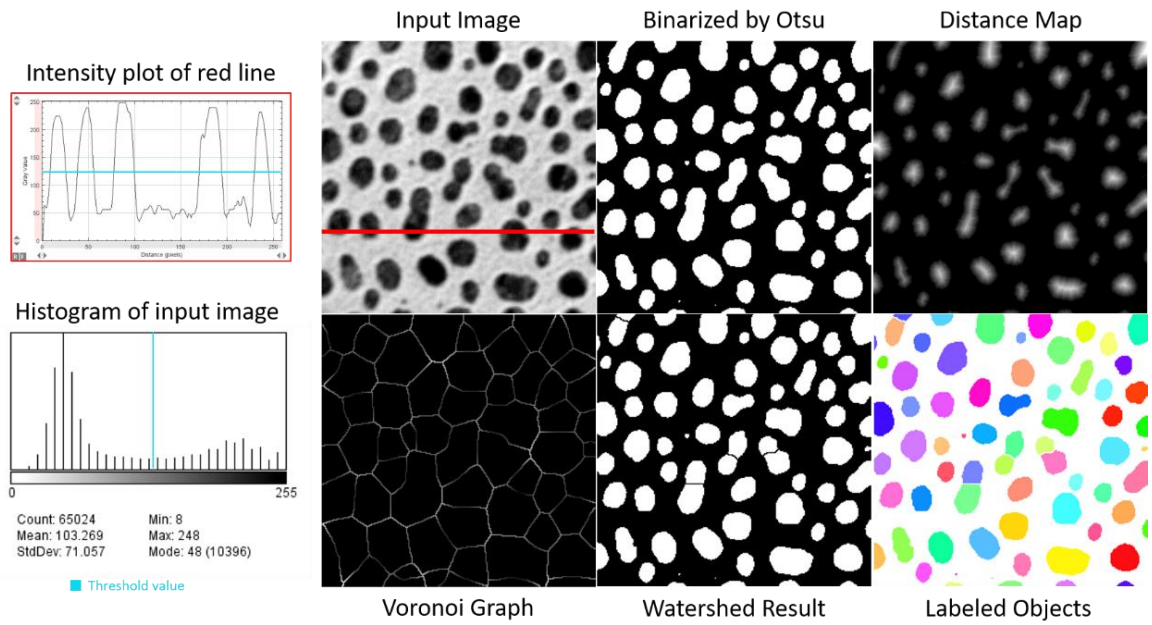

Figure 11 example of watershed segmentation workflow, used input image is from sample image datasets of Fiji

The main workflow of clumped object splitting by watershed segmentation (Barnes et al., 2014; Beucher & Lantuejoul, 1979) is illustrated in Figure 11. Here the binarized image transformed to a distance map via Otsu (Otsu et al., 1979) thresholding. The Otsu threshold is automatically calculated by minimization of intra-class intensity variance of the image. Distance map transformation is performed by sequential image erosion of the binarized image, where every step count is set as intensity on disappearing pixels. Borders of every probable object are highlighted by a Voronoi graph, which is the most distant line of local maxima from the distance map. Stepwise flooding the distance map with increasing intensity values results in splitting of the clumped objects by creation of artificial borders on half way distance to neighboring maxima.

CellProfiler (Carpenter et al., 2006; Kamentsky et al., 2011; McQuin et al., 2018; Stirling et al., 2021) is a modularized free open-source software, which allow for feature extraction and analysis of microscopic images, without knowledge in programming. This application was used to develop general image analysis pipelines, which could be also used by other users to analyze their images. Next to watershed segmentation, found objects could be further investigated by specific intensity or shape measurements, filtering and phenotypically classification.

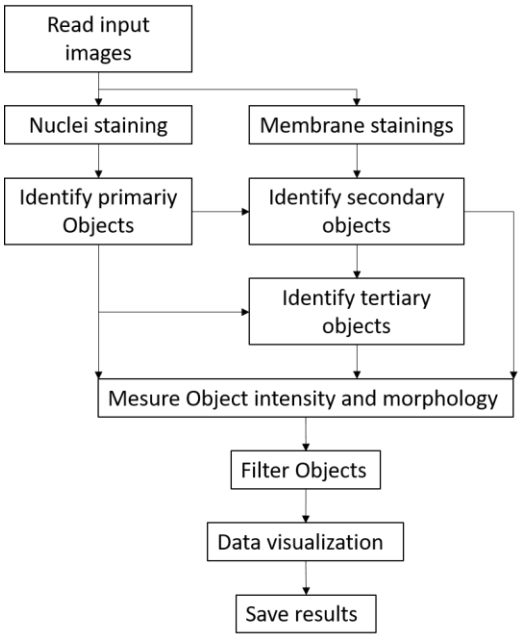

Figure 12 principle CellProfiler pipeline for cell identification and classification

The main modularized workflow of a segmentation and phenotypically classification pipeline in CellProfiler is shown in Figure 12. Briefly, input images are split by marker into nuclei or membrane, or optional structural markers, such as vessels. After nuclei detection (Identify primary Objects) via size or shape limited watershed segmentation, secondary objects are recognized. For this, nuclei objects are set as starting point/area in the desired membrane staining images and pixels in the allowed distance over defined threshold are included in these areas. Due to the fact, that membrane staining does not stain nuclei, the binarized membrane objects are subtracted by the nuclei area (Identify tertiary objects). The resulting objects are then measured in their intensity (mean, median, minimum, maximum, etc.) and morphology (shape, area, circularity, etc.) within the original input images. Based on all features, optional filtering can be applied to remove outliers or to phenotypically classify the single cell objects. Following data visualization and saving summarizes the whole process and enables further data analysis.

2.3.2. Seeded region growing

Fiji (Rueden et al., 2017; Schindelin et al., 2012) is a distribution of the open source image processing package ImageJ, which allows editing and analyzing images of different sources. For the purpose of fine structural object detection, I wrote a Java based ImageJ/Fiji PlugIn and applied a recursive seeded region growing algorithm (Shih & Cheng, 2005) before CellProfiler was introduced. This PlugIn was used to detect nuclei and membranes in MELC, which is why the PlugIn was called “MELC_Evaluation_Toolbox” (source code in Appendix).

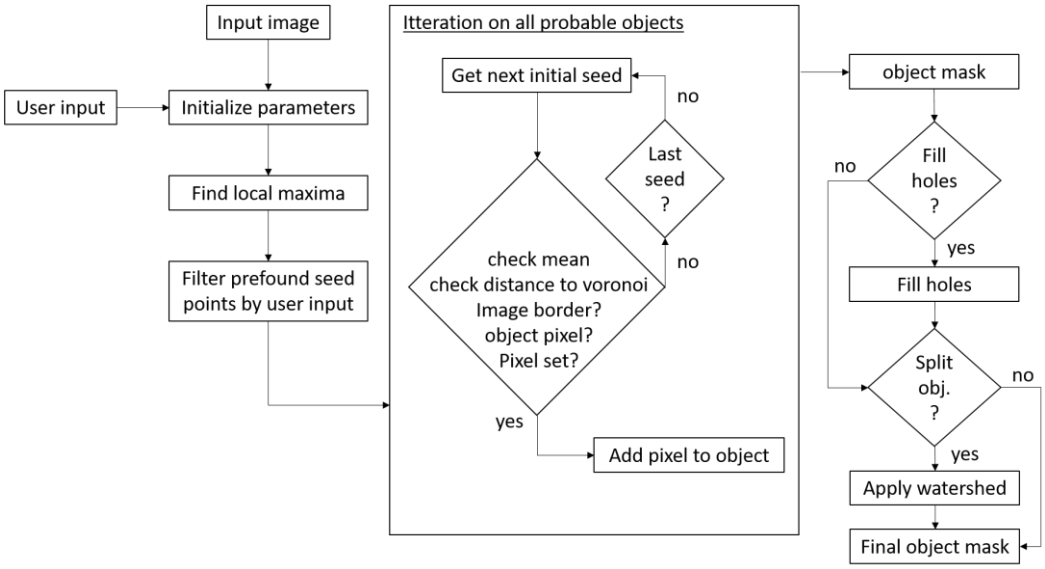

Figure 13 general workflow of developed seeded region growing ImageJ PlugIn

The basic concept of seeded region growing mask creation is illustrated in Figure 13. After image selection and user defined parameters for local maxima initialization, the starting (seed) points of probable objects are filtered by a preselected neighborhood distance. Sequentially growing of the objects create a binarized mask image. During the growing process, the pixels surrounding the object of interest are tested following the criteria:

- Intensity - should be similar to mean value in a four connected pixel neighborhood
- Distance – should be in range of the half distance of probable next cell
- Diameter – should be in range of the allowed user defined range
- Location – should not be at the image border
- Existence – should be an unlabeled pixel

If all the listed criteria are fulfilled, the current pixel is added to the object and the next growing round of the four connected neighborhood pixels is performed until no further expansion is possible. In the end, the object's missing pixels within the structure can be filled, or if more less roundish objects like nuclei should be found, watershed-based object splitting can be applied.

2.3.3. Random forest probability calculation – ilastik

Common image segmentation workflows or algorithms are based on differentially signal distribution and single manual featured limitation by image filtering (edge detection, distance or color information). Ilastik (Berg et al., 2019) is an interactive machine-learning-based tool that offers a different way of segmentation. Besides object detection via pixel classification, ilastik offers object classification and object tracking in multidimensional images

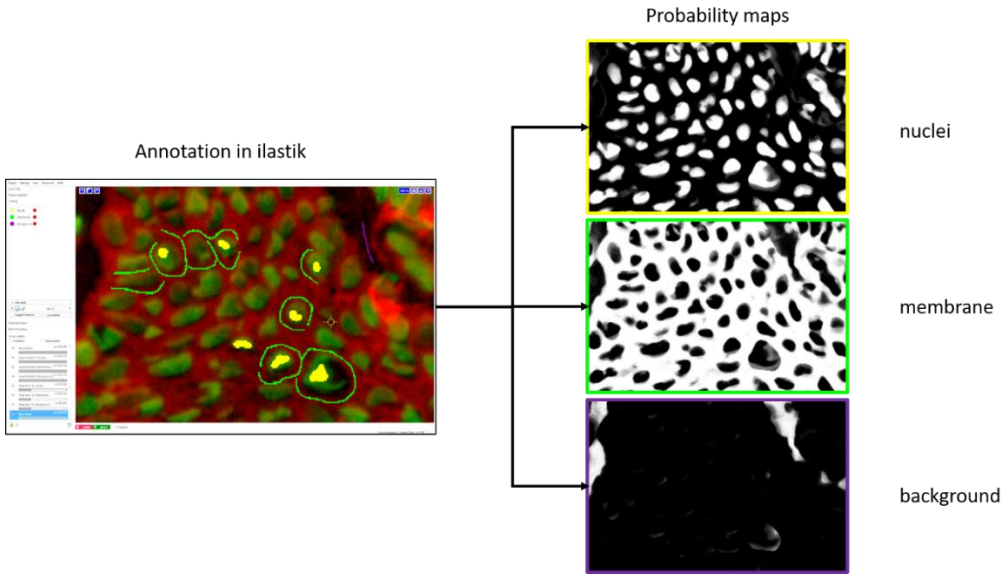

Figure 14 Overview of pixel classification by featured random forest algorithm inside application ilastik by manual annotation (left) and resulting probability maps (right)

In short, input image(s) are convolved at the beginning by a set of image filters at different sizes. Based on the structural information the sizes can be adapted by the user. These filters aim to enhance or suppress image features (edges, gradients, local maxima, etc.). By manual annotation (Figure 14, yellow, green and purple highlighted pixels) and classification (related labels) of singular or connected pixels, a random forest classifier calculates a model and predicts the probability of the unannotated pixels in the desired classed labels. Due to multi parameter weighted decision trees, the probability maps are robust against possible heterogeneous staining quality. The transformation of absolute intensities into probability values, in different classes allows for image segmentation by simple thresholding.

2.3.4. Data analysis environment

Besides devices for image capture and object related feature extraction, this section lists programs, programming languages or applied scripts used in this project, each of which supported biological data validation. Single packages/functions within used programming scripts can be found within the source code in appendix.

Anaconda

A programming environment with “Spyder” for Python script development including interactive data mining tool “Orange” (Demšar et al., 2013), which also can be extended by python scripts (https://www.anaconda.com/).

Scripts:

a) MELC preprocessing (see Appendix, p. 116)
b) LSFM-destriping and related SNR calculations (see Appendix, p.152, 162)

CellProfiler

A modularized free open-source software, for feature extraction and analysis of microscopic images (https://cellprofiler.org/)

Pipelines:

MELC images object identification based on raw intensities or probability maps and neighborhood analysis.

Eclipse

A programming environment for java based script development (https://www.eclipse.org/).

Scripts:

a) MELC evaluation toolbox development as Fiji/ImageJ PlugIn, containing seeded region growing object identification (see Appendix, p. 90).
b) Marker specified searching for combinatorial binarized phenotypically classification of multi-channel microscopy images (see Appendix, p. 107).
c) Shape Data Evaluation (SHADE) as Fiji/ImageJ PlugIn, which calculates Fourier coefficients for shape representation of single 2D image sets (see Appendix, p.84).

Fiji/ImageJ

Fiji is a distribution of the open source image processing package ImageJ, enabling interactive editing and analyzing images of different sources (https://fiji.sc/). Beside manual image processing, automated workflows can be programmed in imageJ macro language.

Scripts:

Fourier coefficients calculation of closed identified object's outlines (see Appendix, p. 113).

Ilastik

An interactive machine learning based tool for object detection via pixel classification, object classification and object tracking in multidimensional images (https://www.ilastik.org/).

Pipelines:

a) MELC images object pixel classification based on trained random forest classifier model of raw intensities from separated images of nuclei and membrane staining, additional background map for object separation.
b) LSFM image pixel classification used for signal to noise ratio calculation, signal to background ratio calculation respectively.

Loupe Browser

Application for data exploration of Spatial Transcriptomics (10x Genomics) data and preprocessing, e.g. manual tissue alignment, for Space Ranger. (https://www.10xgenomics.com/products/loupe-browser)

Matlab 2014b

A programming environment for Matlab based scripts in case of numerical calculations and image processing (https://de.mathworks.com/products/matlab.html).

Scripts:

a) Neighborhood analysis of MELC images, including import of CellProfiler output (see Appendix, p.133).
b) Randomnes test, validation of independent distribution observations (see Appendix, p. 133)

R Studio

A programming environment for statistically data analysis based on programming language R (https://www.rstudio.com/)

Scripts:

a) Spatial Transcriptomics data analysis including batch normalization, or data integration respectively, dimension reduction visualization and clustering (see Appendix, p.155).
b) MELC analysis in sense of ST data analysis (see Appendix, p. 130)
c) Gene set enrichment analysis (groupe based and single sample based), see Appendix p. 148, 150
d) Neighborhood analysis (see Appendix, p. 140)

Space Ranger

Command line controlled (Linux) application to process ST based measurements, including sample demultiplexing, image alignment, gene counting and barcode processing https://www.10xgenomics.com/).

3. Application and Results

The selection of a measuring device is related to the biological question that is being addressed in a certain experiment. The decision to use a particular microscope depends on several factors, including the tissue sample properties, required resolution, 2D vs. volumetric observation, number of different fluorescent markers, usable and available stainings, etc. Depending on the specific imaging techniques, image processing and data analysis needs to be adapted. This chapter describes the application and the development of different workflows used in this thesis. A variety of imaging techniques was used, each with a different application: Laser Scanning Microscopy (LSM), Multi Epitope Ligand Cartography (MELC) and Fluorescent Light Sheet Microscopy (LSFM) give information about imaged samples on the protein level, Spatial Transcriptomic (ST) yields information about the sample on the transcriptional level.

3.1. Image analysis of LSM data

Conventional fluorescence microscope systems are limited in the number of separable wavelength related fluorescence channels and are therefore limited in the number of simultaneously detectable cell types. Depending on the object of interest, a single channel image could be enough to extract features for characterization of biological phenomena.

In order to analyze morphological differences in microglia in a dementia model, we acquired confocal images of the front parietal cortex of young (6 months) and old (20 months) Iba-1-EGFP reporter mice. Here 30 µm thick tissue samples were imaged at 0,42 x 0,42 x 0,87 µm³ voxel resolution, resulting in 3D image stacks. The maximum intensity projections were calculated to retain spatially oriented processes and cell bodies of microglia in one final image per field of view (FOV).

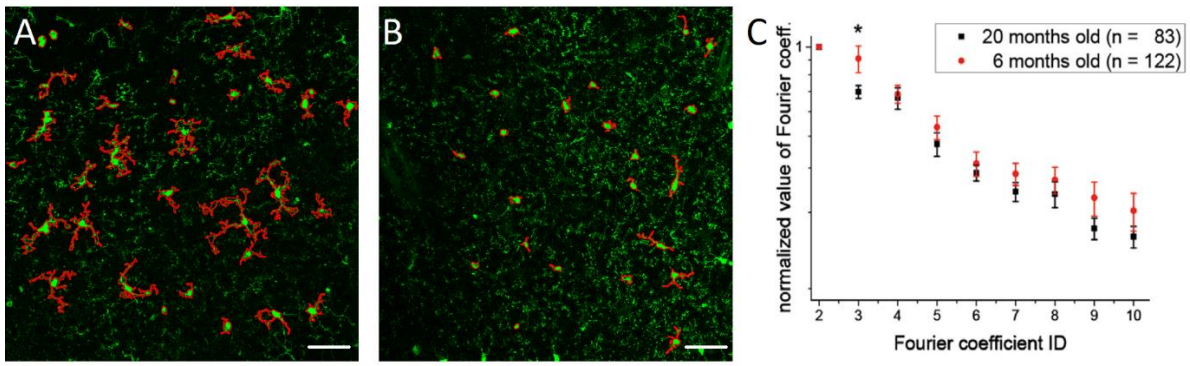

Figure 15 Depicted maximum intensity projections of young (A) and old (B) mice's microglia, segmentation result in red outlines, averaged and normalized Fourier coefficients (C), comparing young and old mice, scale bar 50 µm

The identification and segmentation of microglia was obtained by applying a self-written Java based ImageJ PlugIn, where a seeded region growing algorithm was implemented to ensure fine structural preservation of the processes in case they were present (Figure 15). To objectively quantify the estimated difference in morphology, the outlining of the connected

pixels of every single object were converted to discrete Fourier transformation coefficients (shape descriptors). The higher the coefficient ID by higher value (amplitude) at the same time, the more complex the microglia shape became. Up to 20 normalized coefficients were used to determine shape information. This approach could be used to show a loss in microglial processes in aged mice.

3.2. Device optimization of the MELC system

MELC is a highly multiplexed cycled immunofluorescence (IF) based microscopy technique that generates a large amount of data, due to the image acquisition procedure. Within one MELC run, four steps are sequentially repeated. A) loading the fluorescence-coupled antibody incubation and washing, B) microscope-sample autofocusing referenced by a phase contrast image from the field of view acquired at the beginning of the cycle and following 3D image acquisition of fluorescence images, C) fluorescent signal removal via photo-bleaching, D) another phase contrast based autofocusing step for bleached image acquisition. Besides the fluorescence images, all phase contrast images of the field of view are recorded.

Performing MELC experiments requires proper sample handling on the one hand, and on the other hand the right functionality and communication of all related devices. Over the time, parts of the system have been changed, which has led to enhanced image quality and reduced hardware-dependent error sources.

3.2.1. Camera

One of the most important devices on a wide-field microscope is the camera. It digitalizes the visual observations, which can be saved, shared and automatically analyzed. The performance of such a device determines the detectable phenomena in the examined sample.

The pre-installed camera of the toponome-image-cycler (TIC) was a 14-bit Apogee KX4 CCD device with 2032x2044 pixels, where every pixel had a size of 9x9 µm². During image acquisition a manual shutter opened and the light sensitive CCD-sensor detected the intensity information.

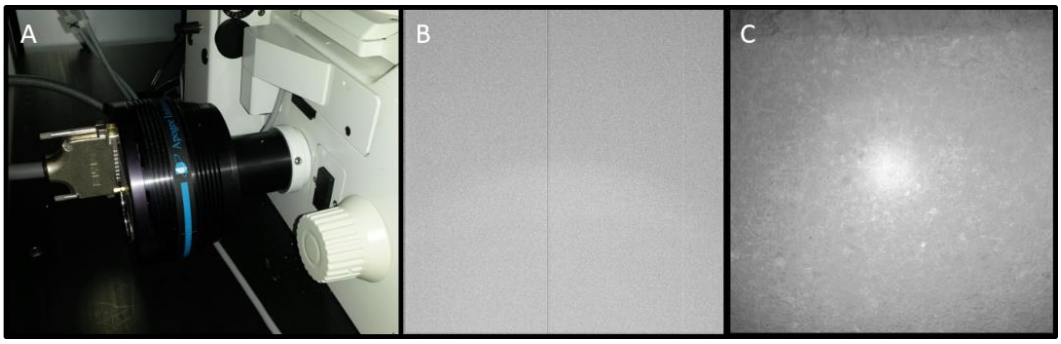

Figure 16 Pre-installed camera system of toponome-image-cycler (TIC), (A) camera connected to microscope, (B) dark image, (C) low concentration fluorescence staining with centered bright spot artifact

Quality control of the MELC system's eye showed camera related artifacts, including external light contribution, missing pixel information (Figure 16B) and artificial centered bright spot

illumination (Figure 16C) during image acquisition. In case of the two first artifacts, a simple light path covering tube would have minimized outer light influence. Single lines to an amount lower than 1% of missing information on a whole image, caused by camera pixel defects, would have a lower impact on the following image analysis. However, in case of Figure 16C, the detected image information is affected by hardware failure and therefore distorted image content. The motorized shutter on the camera exhibited a signal delay that caused the CCD chip's exposure time to vary, and also a non-fully closing shutter that produced a steady exposure that caused the centered bright spot illumination. This is why a new camera was required, to not only get better quality images, but to ensure reproducible experiments as well. Based on the requirements, we chose a Hamamatsu 16-bit ORCA-Flash4.0 LT scientific CMOS camera. Compared to the old camera, the new one offered more pixels (2048x2048) at smaller size (6,5 x 6,5 µm²) by lower dark current (0.6 < 3,5 electrons/pixel). Unlike the motorized hardware shutter, an electronically time-controlled readout shutter was used.

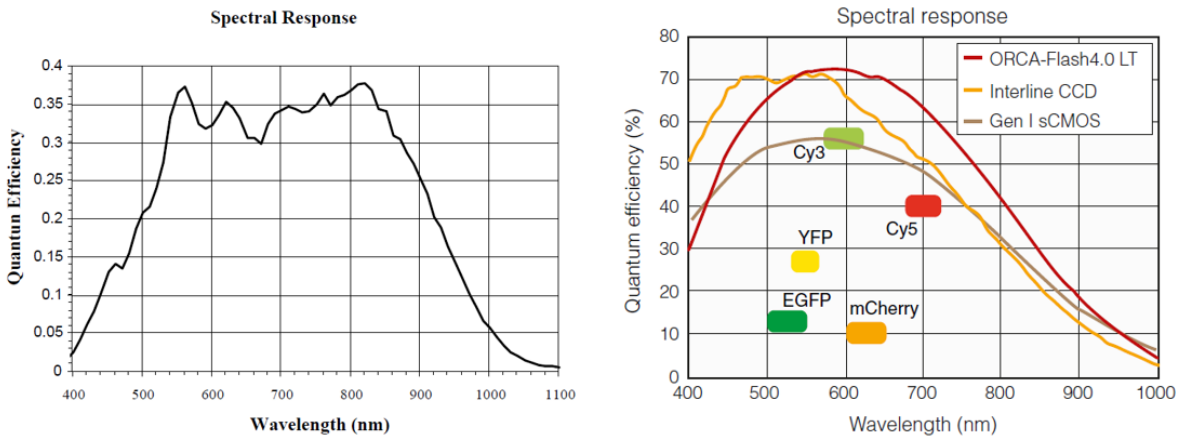

Figure 17 Quantum efficiency plots of (left) Apogee KX 4 CCD camera and (right) Hamamatsu ORCA-Flas4.0 LT CMOS camera, plots are taken from manufactures manual

For image acquisition at low fluorescence signal amplitudes in MELC experiments, one of the most important decision parameters was quantum efficiency. In contrast to the old camera, the quantum efficiency was higher as a function of all wavelengths examined and distributed over an extended range (Figure 17), which resulted in a double photon number at intermediate wavelengths.

Additionally, due to the smaller form factor of the camera, the position of the camera could also be changed. Instead of the side port, it was possible to switch to the bottom port. This increased the transmission power from 80% to 100%, resulting in higher sensitivity. Consequently, weak signals could be detected better.

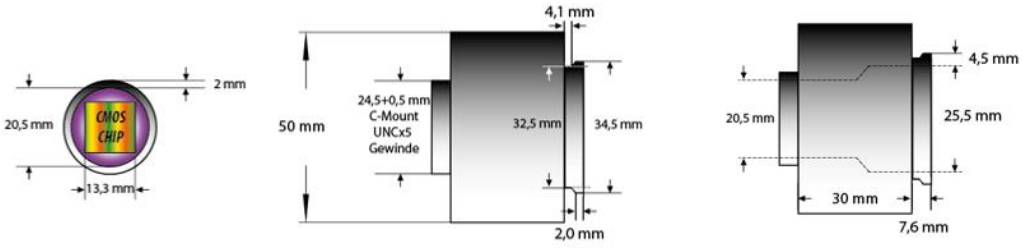

Figure 18 The sketch of self-designed and manufactured camera-microscope-adapter on lathe

The new camera was connected to the microscope using a new, self-constructed aluminum adapter that was manufactured on the lathe (Figure 18) and anodized to avoid reflections. The new adapter manufactured showed no disadvantages in terms of sample illumination, or image corner intensity loss.

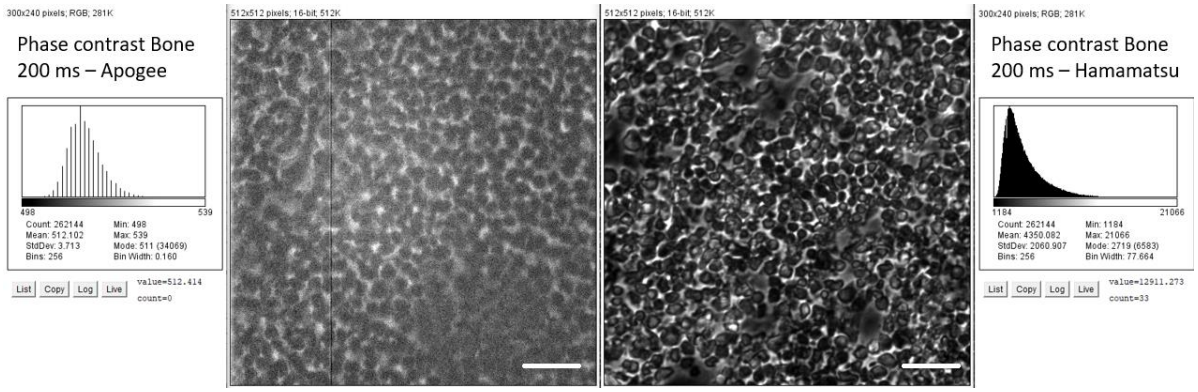

Figure 19 Phase contrast images of mice's bone marrow as comparison between old (left) and new (right) camera setup - samples are acquired under same light and exposure time conditions, scale bar 100 µm

In order to test the enhanced camera properties and related image quality, phase contrast images of mouse bone were acquired and compared (Figure 19). These images were independent of fluorescent staining and sample quality. The overall image intensities and dynamic range were dramatically increased under same condition. A cloudy plastic pane in front of the old camera was observed during replacement, which may have caused additional light dimming.

Altogether, the images showed higher dynamic range of intensity compared to previous camera setup under the same illumination conditions. Due to the widened quantum efficiency wavelength band, fluorescence signals from far red and near blue could be detected, allowing the use of new fluorescent coupled antibodies, extending further multi-parameter functional analysis.

3.2.2. Light source

The inverted microscope used for these experiments used a pre-installed arc lamp as light source for fluorescence excitation. The lamp was incorporated in the MELC system, and was

thus located within the closed box, together with the microscope, well plate holder for staining reagents and robotics.

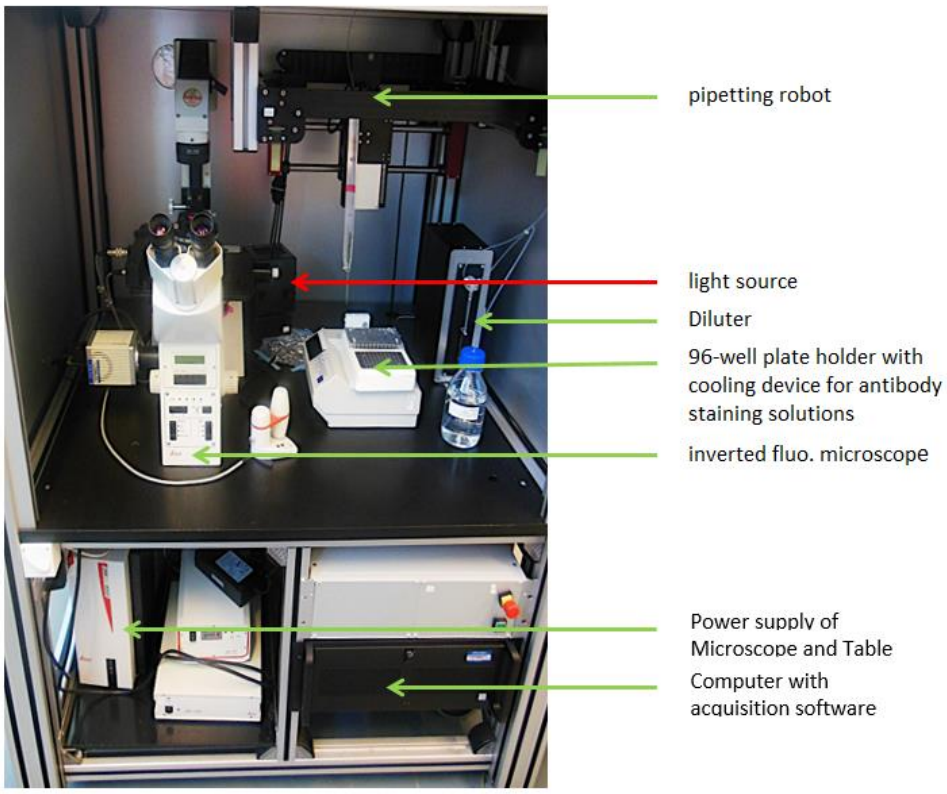

Figure 20 overview of closed MELC system inside box, directly connected light source highlighted by red arrow

Due to the location of the fluorescent light source (Figure 20), the internal temperature of the closed measuring station increased to up to 35 °C ± 2 °C at a humidity of about 20 % ± 4 % during a MELC experiment, whereas the usual room temperature was around 20 °C. Along the usual incubation times of several minutes, this led to the sample drying out. Suppressing dry and hot air with additional cooling devices such as fans or ice inside the box would have created additional problems such as dust flying around inside the sample environment or convection on optical components. Therefore, the light source was replaced by an external one (Leica EL 6000 equipped with Osram HXP R 120W/45C VIS), in which the light was transported via a liquid-filled cable. In addition, there was no need to readjust the arc lamp. The excitation spectra were increased in dynamic range and amplitude, which resulted in a higher signal quality along the wavelength band.

3.2.3. Optical components

Besides the camera, the related optics are the most important components of a microscope. They deliver the excitation light to the sample and transmit the emitted light to the camera. Light gets focused through the objective lens, onto the sample.

The fluorescence filter revolver of the utilized inverted widefield microscope was equipped with four filter cubes. These cubes work as a bandpass filter configuration and split the excitation light path from the emission detection light path depending on wavelength. Only three out of four possible filter cubes were used here, since one position was used for the phase contrast image configuration. In this setup, the microscope handler could see the phase contrast images using white light. In addition to that, these images were not dimmed by any optics, which gained the image intensity required for the camera to operate and for the automated focusing procedure during MELC experiments.

Table 1 Installed fluorescence filter cubes at MELC system and related maximum excitation and emission wavelengths [nm] including bandwidth

Filter name	Excitation max. [nm]	Emission max. [nm]
DAPI	350/50	460/50
FITC	475/40	520/40
PE	546/10	585/40
APC	640/30	690/50

Due to the higher light sensitivity of the new camera setup, the empty filter cube slot was configured by an additional filter (DAPI, Table 1), which increased the number of detectable fluorescently labeled antibodies. Phase contrast images didn't change in information content, only changed in observed color. The emitted light was bright enough for the camera to process it. In case filters were damaged, they were replaced. The bandwidth of all excitation and emission spectra around their maxima was adjusted to avoid possible signal spillover from fluorescence light.

In line with the new camera's smaller sCMOS pixel size (6,5 µm x 6,5 µm), the pre-installed objective lens (Leica HC PL FLUATOR 20x/0.50 PH 2) was replaced by an objective lens with higher NA at the same magnification (Leica HC PL APO 20x/0.80 PH 2). This allowed for the photon count entering the focal volume to be increased, which resulted in an enhanced emission photon count, and increased intensity of low signals within the microscopy images. An improved resolution was expected as well.

Table 2 lateral and axial resolution test of new equipped MELC system

Filter name / bead wavelength [nm]	old theoretical resolution [nm] lateral / axial	new theoretical resolution [nm] lateral / axial	FWHM distance [px] Lateral / axial	bead size [µm] lateral / axial
DAPI / 430	524,6 / 3440,0	327,9 / 1343,8	12,2 / 5,3	4,0 ± 0,2/ 3,9 ± 0,2
FITC / 515	628,3 / 4120,0	392,7 / 1609,4	10,7 / 6,4	4,2 ± 0,2/ 4,0 ± 0,2
PE / 580	707,6 / 4640,0	442,25 / 1812,5	9,3 / 7,2	4,1 ± 0,2/ 4,0 ± 0,2
APC / 680	829,6 / 5440,0	518,5 / 2125,0	8,2 / 8,5	4,2 ± 0,2/ 4,0 ± 0,2

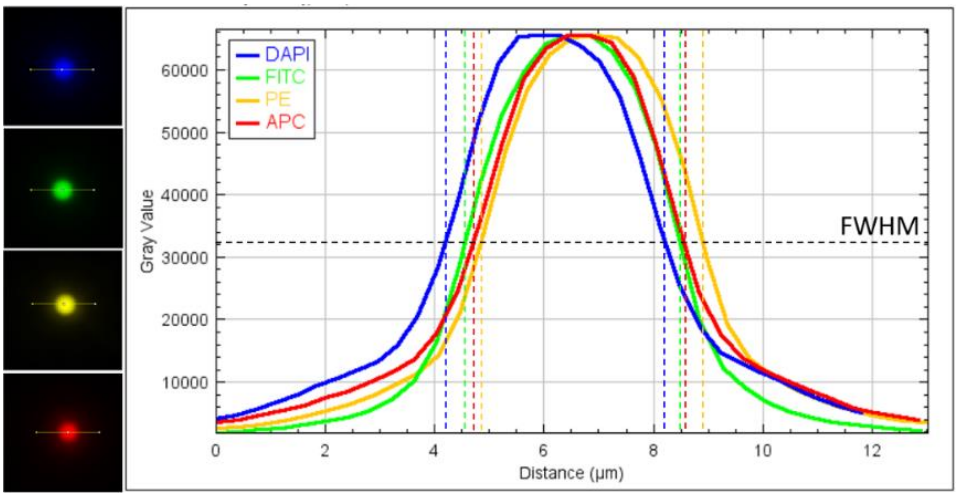

Figure 21 lateral resolution estimation on 4 µm beads embedded in agarose along all fluorescence channels

In order to estimate the resolving power, mid-range fluorescent polysterol beads (four micrometer in size) were imaged in all fluorescence channels, and measured along their lateral and axial cross sectional intensity profiles. These outcomes were compared to theoretical calculated resolving power based on Rayleigh criteria (Table 2, Figure 21). The beads were embedded in a one percent agarose gel, simulating tissue environment. The intensity profiles were fitted by a linear regression to Gaussian distribution function to calculate and measure bead diameter at full width at half maximum (FWHM). The resolution power improvement estimation confirmed expectations.

In total, the reconfigured optical settings improved lateral and axial resolution power by a factor of 60 % and 250 % respectively. Furthermore, the new fluorescence filter cube configuration extended the detectability of fluorescently labeled antibodies. A shift in the fluorescence intensity maxima of the beads was observed during the resolution estimation, indicating hardware-related tolerances in the placement of the filter cubes. This offset had to be taken into account in the subsequent image analysis in order to ensure the signal origin of the corresponding structures.

3.2.4. Sample holding and washing box

The cover slides prepared for MELC experiments contained the tissue sample and a 100 µl PBS reservoir made from silicone via the “press-to-seal” method. Due to the shorter free working distance (400 µm) of the objective lens (Leica HC PL APO 20x/0.80 PH 2), Menzel-Gläser cover slides of size 24x60x0.17 mm³ were used. These were held by a simple sample holder plate made from plastic. The application and suction of system liquid (PBS) required an additional box around the sample, where waste material could be placed without influencing the samples’ environment. The previous washing box had a limited volume, which created liquid overflow during long MELC experiments. Furthermore, leakage of the simple sample holder would also have led to liquid with respect to the microscope and caused damage

Figure 22 Self designed and 3D PLA printed equipment for MELC system, from left to right: sample holder, washing box and washing box cover

Therefore, the sample holder and the washing box were re-designed and 3D printed from polyactide (PLA, Figure 22), and equipped with drain, preventing the washing box from overflowing. The sample holder was provided with a recess for the coverslip and the washing box had been provided with a hole the size of the transmission light condenser along the optical axis, which held the washing box in place during sample movement through the microscope stage. In this configuration, longer automated runs were possible, avoiding frequent liquid level monitoring.

3.3. Image preprocessing of MELC data

The wide field microscope is subjected to physical limitations and thus image formation can be affected by aberrations. The most prominent factors leading to these phenomena include image transition in between fluorescence channels due to mechanical tolerances, background illumination, caused by spherical aberration of the optics, or varying liquid levels. The elimination of these artifacts guarantees standardized input for downstream quantified image analysis and thus enhance reproducibility of experiments. All image preprocessing steps were included in one Python script, which read the MELC run initialization file used for MELC experiment as well. Within this file, all information about the automated cycled image acquisition and bleaching of the fluorophores was defined, as well as the sequential naming of all the steps. Only at the beginning of script execution the user is asked to select secondary markers, since this information was not included in the file. All other preprocessing steps were performed automatically.

3.3.1. Registration

The image registration was adapted to the imaging procedure, where an iterative cross correlation auto-focusing procedure, with the support of reference phase contrast images tried to find same region of sample. Due to slight shifts in the FOVs mechanical tolerances and limitations, realignment of images was required.

Post image registration was performed in similar manner as the automated image acquisition, but with a sub-pixel resolution, up to one tenth, via a cross correlation algorithm, phase correlation respectively due to calculation in Fourier Domain.

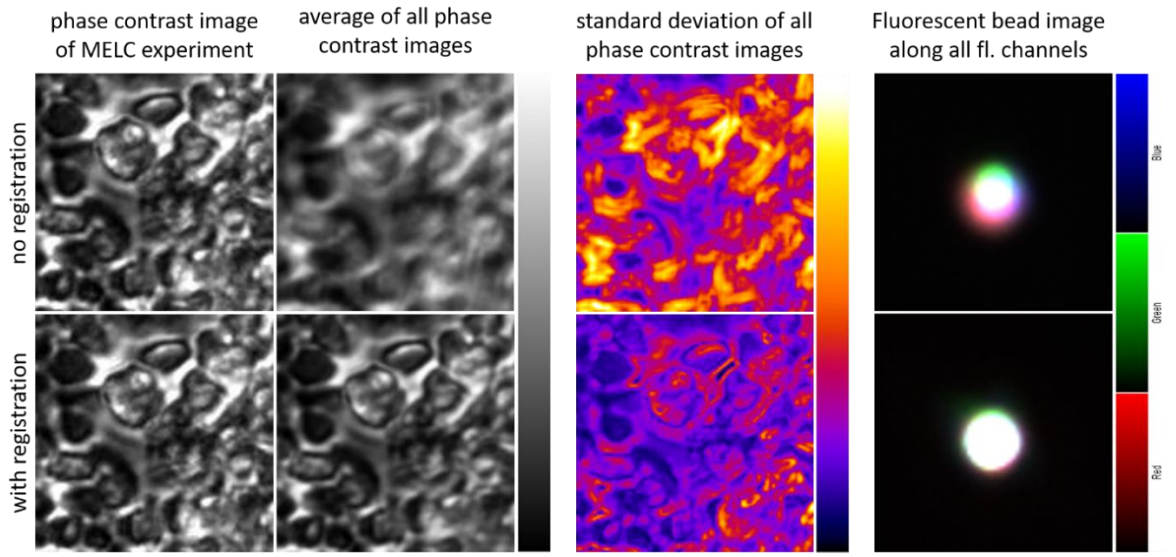

Figure 23 image registration quality check on phase contrast (column 1-3) and fluorescent (column 4) images, where first row are the original input images and in the second row the realigned images, registered respectively. Phase contrast images in gray scale, average images as well, standard deviation of images color coded in fire look up table, fluorescent images in all four channels (red, green, blue, gray)

Phase contrast and fluorescence images from a 32 marker MELC experiment, were used to test the performance of the developed registration workflow. In order to illustrate the image displacement and registration results, the average and standard deviation of the projection was calculated, as shown in Figure 23. All 32 displaced phase contrast images of the same fluorescent channel created a blurred results image. The local pixel information was shifted in depth and the information for all the fluorescently labelled antibodies was shifted laterally, which disabled accurate intensity profile measurements. As a measure of uncertainty, the standard deviation of the projection of these 32 phase contrast images highlighted the most displaced regions within one final image. Compared to the raw images, registered images appeared less blurry and displayed a lower standard deviation, which verified the inner channel shift correction. The same workflow was applied on fluorescent bead images from every fluorescent channel referenced on the same channel as in phase contrast images. The three-dimensional displaced bead signals could be registered as well.

Image registration verifies further signal co-localization analysis, enabling intensity profile comparisons on a pixel level within the set of fluorescent images of the same field of view.

3.3.2. Illumination correction

Working with high NA objectives and light point sources creates an uneven illumination of the imaged area. Even if the microscope is in the best condition and set to the Köhler illumination setting, the central area gets more illuminated than the edges, which causes the same cell to be differently illuminated depending on the location within field of view.

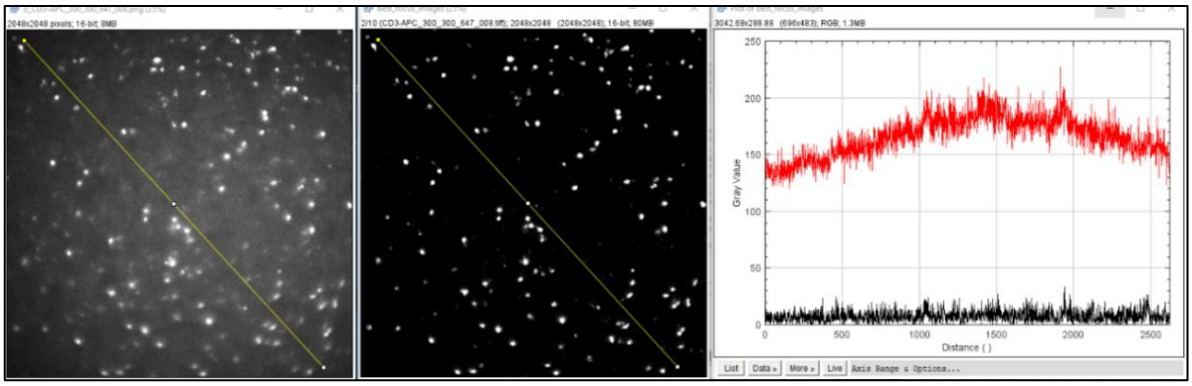

Figure 24 Illumination correction result, (left) original CD3-labelled image, (center) corrected CD3 image, (right) line plots along yellow selection of both background signals (red original CD3, black illumination corrected CD3), contrast are set to 0.35 % saturated pixels

Applying a flat field correction algorithm suppressed the main irregularity intensity distribution, as shown in Figure 24, where an image of cells stained with fluorescent anti-CD3 antibody is corrected.

Compared to primary antibody image illumination correction, where the fluorescent image was subtracted by the previous bleaching image and then divided by the mean-normalized background estimation, secondary antibody fluorescent images were subtracted by their primary fluorescent image (two steps before) and then divided by the mean normalized background estimation. In this way, all signals introduced by this pre-staining step were removed, leaving only the secondary antibody intensity information.

Polynomial background estimation based on the fluorescent bleaching image reflected the spherical aberration caused by the lens and the liquid level on top of tissue sample, but with enhanced noise in the image corners. Local SNR in these regions remained the same, due to the multiplicative illumination correction value. If the fluorescence signal was previously separable from the background, this fact holds true also after the illumination correction.

3.3.3. All in focus projection

Tissue sample preparation and the positioning of the sample under the microscope can cause variations of the focal plane, due to tilted table adjustments, wavy like tissue contact to the coverslip, uneven sample thickness, or object positions distribution within thicker tissue sample slides.

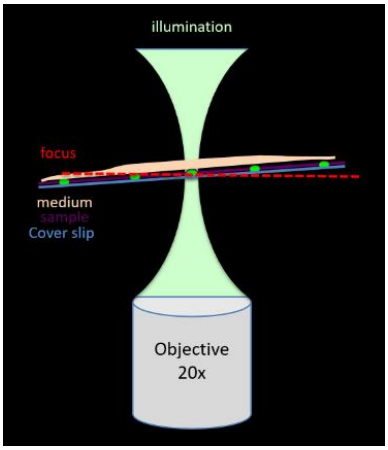

Figure 25 Sketch of tilted sample. Light green: illumination distribution, green: cells, red: focal plane, orange: liquid level, violet: sample, blue: coverslip

For these reasons, multiple images over the entire range of depth along the major focal plane (red line in Figure 25) were acquired, most of the time by the diameter of one nucleus, +/- 5 µm respectively. The so-called image stack contained three-dimensional information of the investigated tissue sample, which increased the opportunity for the sample to be in focus.
Next, an all-in-focus algorithm (based on “extended depth-of-field” (Pertuz et al., 2013)) was applied to project the depth information back into a two-dimensional image to compensate the tilted sample.

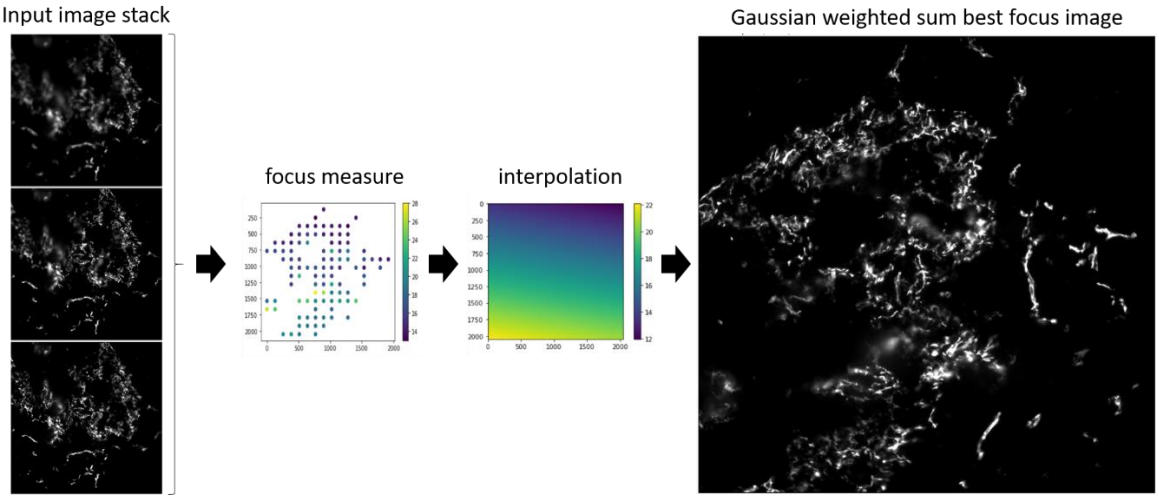

Figure 26 Workflow overview of best focus calculation

The workflow of the best in focus calculation (Figure 26) was split into three parts. Within the first step, the entire image stack was transversally divided in a 32 by 32 grid (64px by 64px). Here, the axial index of the best focus was found in each tiny stack by taking the maximum standard deviation index of the inspected slices when testing the background corrected mean intensity value in parallel. If the local mean intensity value was higher than the global mean intensity value of the whole image stack, the focus index was assigned for the second step. Tiny stacks which did not fulfilled the criteria were assumed to be background and are therefore rejected from the following interpolation step. Within the interpolation (second) step, a “least

square fitting" algorithm was applied to fit a second order polynomial plane through all possible indices, which was used to create a two dimensional focus map at the same lateral size of the input image stack. Instead of simple two dimensional plane estimation, second order polynomial plane estimation was applied, to improve possible image distortion or tissue bending caused by altered liquid levels. The values in this map were focal plane indices. In the third and last step, the values from the input image stack at the calculated focal planes were collected. Due to the one micrometer step width of every layer, intensity values around an axial Gaussian distribution with standard deviation of 1 were weighted, which reflected the intensity spreading initialized by the system PSF. A normalized weighted sum along the axial axis at the end generated the best in focus projected image.

The projected images reflected the sample content and eliminated the out of focus regions. In case of superimposed objects, the most likely focal plane index was chosen, based on the focus measure by standard deviation.

3.4. Image analysis of MELC data

Multi parameter images acquired with MELC contain separated information distributed in various dimensions, if every different fluorescent staining is defined as a dimension. In a combinatorial way, the information enables deeper insights than the single dimensions alone and retains the possibility to answer complex biological questions on a spatial localized protein level. Extraction of relevant information out of the images requires object detection, which makes quantification, expression level comparison, cell communication and tissue dependent hypothesis testing possible in the first place.

3.4.1. Image segmentation

In this work, one of the first segmentation pipelines was created in "CellProfiler" to identify stromal cell networks and cell subsets (Holzwarth et al., 2018) in six different MELC run image data sets from mouse bone marrow. Relying on image preprocessing, expression levels were comparable (as tested by signal to noise transformation). Therefore, manual threshold-based image binarization/segmentation of the network architecture could be applied and single cell object detection was performed via automated watershed segmentation based on the Otsu threshold calculation.

For quality control of the automated cell recognition pipeline, six trained raters were randomly selected and asked to count cells within MELC images, in which B220 and ckit stained cells represented the objects. The raters counted three different images, to compare the count performance under divergent conditions to those obtained by "CellProfiler" pipeline.

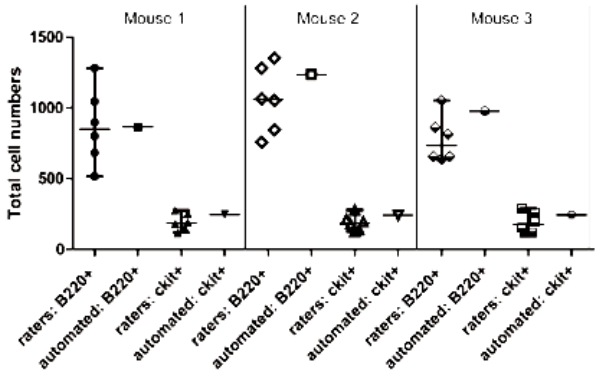

Figure 27 Trained raters vs. automated segmentation validation, (Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

The automated cell segmentation resulted in an average cell count within the range of trained raters in all cases (Figure 27), however there was a high variability in counting of each trained rater.

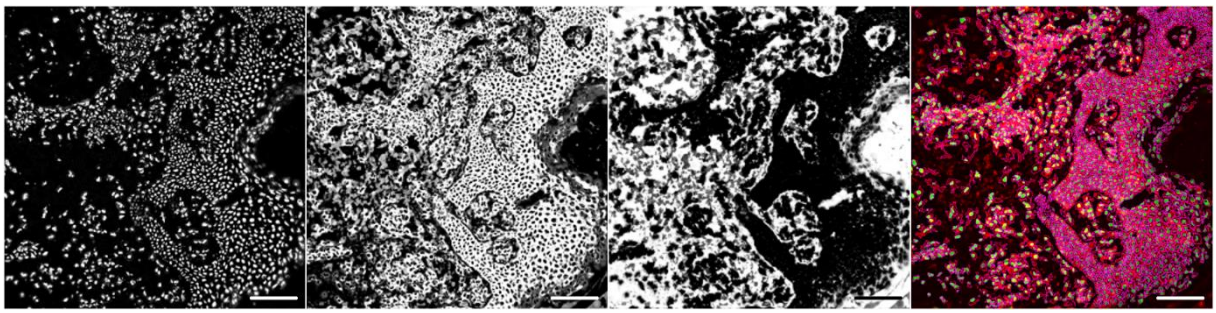

Figure 28 ilastik - CellProfiler segmentation, from left to right: probability maps of nuclei, (summed) membrane and extra cellular matrix staining, segmentation result, scale bar 100 µm

Compared to the used data set, other MELC data showed that tissue sample integrity or staining quality affects the intensity distribution within the single objects and thus throughout the entire images. Traditional watershed segmentation requires a homogenous signal contribution to identify single cell objects. Using the application “ilastik”, which uses a random forest algorithm, we could train the software to recognize nuclei, membrane, and extracellular matrix areas in our MELC images. Upon training, ilastik produced probability maps for each of the assigned pixels and related groups (classes), which were then used for further segmentation in CellProfiler (Figure 28Figure 29Figure 44). In order to obtain membrane and extra-cellular matrix segmentation, summed images of stainings related to the predefined cell region were used for training, to include all possible areas of fluorescent signal types.

Based on nuclei identification throughout adjusted watershed segmentation, cells positive for one particular staining could be found via manual thresholding within single membrane or nucleated regions, depending on the mean fluorescence intensity of the marker within these cell areas.

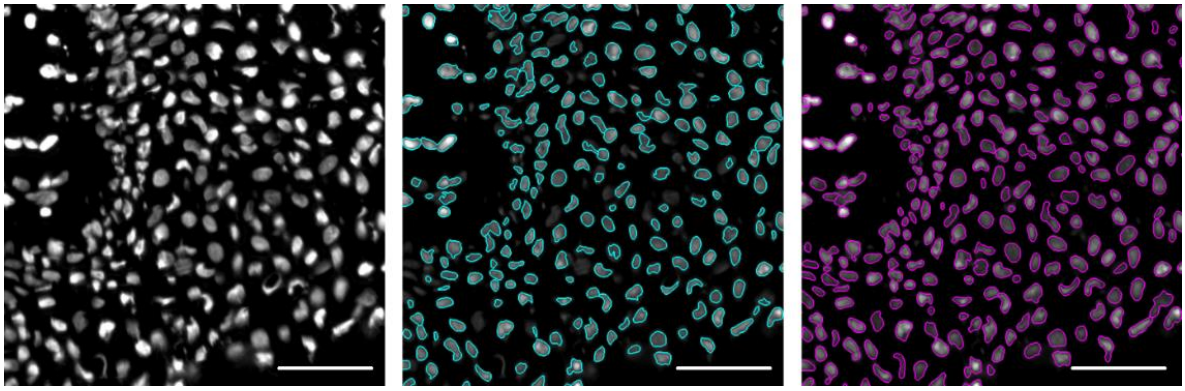

Figure 29 segmentation improvement, from left to right: original image input, global otsu thresholded watershed segmentation (found objects in cyan), ilastik and CellProfiler segmentation workflow (found objects in magenta), scale bar 50 µm

The combination of ilastik and CellProfiler showed an improvement in the segmentation quality. The number of counted cells increased approximately 8%, while the shape of the objects and clumped object separation improved as well (Figure 29).

3.4.2. Cell type identification – phenotypic classification

Compared to conventional microscope systems with a fixed number of fluorescent channels, MELC allows for the acquisition of a theoretically unlimited number of images of the same field of view using diverse fluorescent-labeled antibodies. With increasing number of stainings, the chance of identifying a specific functional cell type increases. Therefore, a “marker specified searching” plugin for ImageJ/Fiji was developed to find user defined cell types within segmented images.

As a test, the developed plugin was used for identification of B cells, plasma cells, T helper cells, cytotoxic T cells and ckit+ progenitor cells within different image stacks from mouse bone marrow (3.4.1). Only objects with nuclei were analyzed. The single combinations taken for different cell types are shown in Table 3.

Table 3 cell type combination matrix

Cell Type	B220	CD138	CD4	CD8	ckit	nuclei
B cell						
plasma cell						
T helper cell						
cytotoxic T cell						
ckit+						

The green highlighted boxes are interpreted as objects present in the image of the staining of interest, while the orange ones are interpreted as no object at the desired image location and staining. Only the cell objects fulfilling the criteria of present or absent objects in the chosen combination were counted and visualized in the image stack.

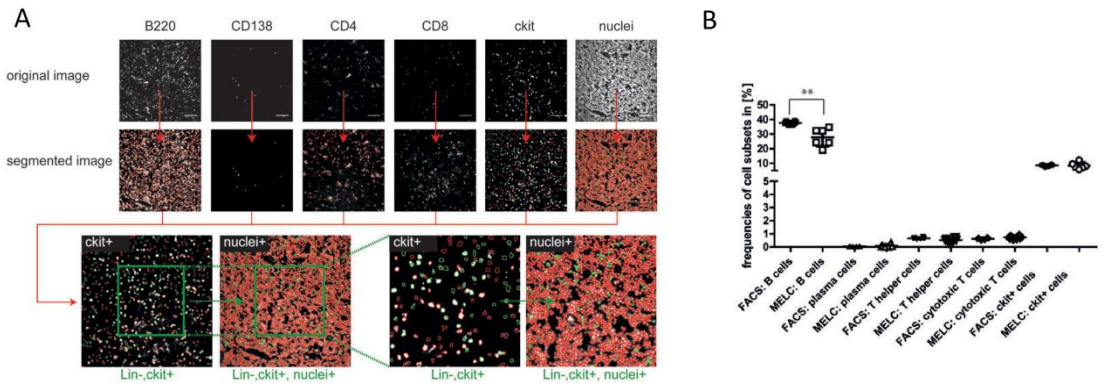

Figure 30 workflow of "marker specified searching" plugin applied on MELC images, validated by FACS. (A) Depicted MELC images of bone marrow from mouse, where red and green outlines represented identified objects. Green objects indicating found cells and related cell type (ckit+, nuclei+) including all other objects with no expression (Lin-); scale bars 100 µm (adopted from Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

Within six image stacks the single cell type frequencies were counted. The data obtained by MELC were compared with FACS data of the same organs to verify segmentation and combinatorial cell type identification via “marker specified searching” at the same time (Figure 30). All frequencies showed no significantly differences, as shown in Table 4.

Table 4 frequency comparison of cell type identification MELC vs. FACS

Cell Type	frequency MELC [%]	frequency FACS [%]
B cell	28.14 ± 5.81	37.80 ± 1.23
plasma cell	0.042 ± 0.12	0.044 ± 0.01
T helper cell	0.54 ± 0.16	0.67 ± 0.06
cytotoxic T cell	0.72 ± 0.08	0.65 ± 0.08
ckit+	7.57 ± 2.29	8.72 ± 0.64

The differences in B-cell abundance may have been caused by heterogeneous expression of B220 and higher variance in segmentation accuracy, but this tolerance of trained raters was taken into account and adjusted for cell identification (3.4.1).

During the cell type identification algorithm development, a manual gating strategy like in FACS analysis was applied based on single cell measured mean intensity to identify known cell types in a supervised manner within MELC image data sets. By manual gating, a combination of allowed minimum and maximum intensity (set by experts) of the single channel expression levels is determined. This extended colocalization analysis tests for appearance of objects and their intensity profile at the same time, allowing further characterization within the found cell types, and finding sub-populations. Throughout five human tonsil MELC data sets, a total of 6391 ± 350 cells could be identified and classified out of a 53 marker panel.

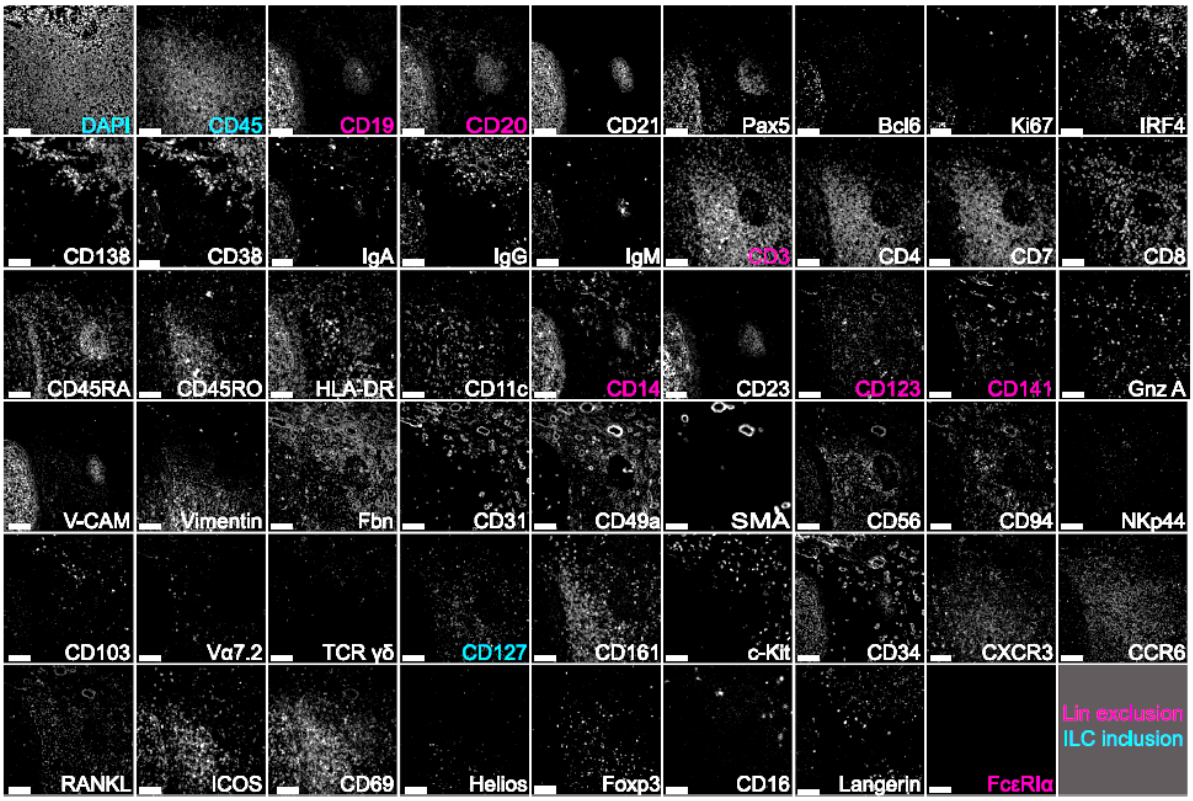

Figure 31 representative 53 marker MELC run panel for ILC identification and localization, scale bar 100 µm (Pascual-Reguant et al., 2021)

Besides the main cell types, such as B cells, plasma cells, T helper cells, cytotoxic T cells, myeloid cells and endothelial cells, we could identify rare innate lymphoid cells (ILCs) as demonstrated by the color code in Figure 31. In order to annotate a cell as ILC, the membrane surrounding a nucleus should have a high expression level of CD45 and CD127, and a low or non-existing level of the lineage (lin) panel (CD19, CD20, CD14, CD123, CD141 and FcεRIα (Pascual-Reguant et al., 2021)). This specific cell type only occurred in 0.15-0.50 % of analyzed data.

Next to the predefinition of known marker combinations a t-distribution stochastic neighbor embedding (t-SNE) by support of principal component analysis was applied to the whole data sets. This resulted in an automated identification workflow that replaced the time-consuming and potentially biased manual work of MELC users. The mean fluorescence intensities of different runs and markers were normalized to ensure comparability and interpretation of the data. Applied hierarchical clustering pre-structured similar related cell signals in a tree like relationship. Furthermore, k-means clustering offered at various cut-off levels major cell types, their sub-population and ILC population as well. Analysis was interactively performed within the application Orange.

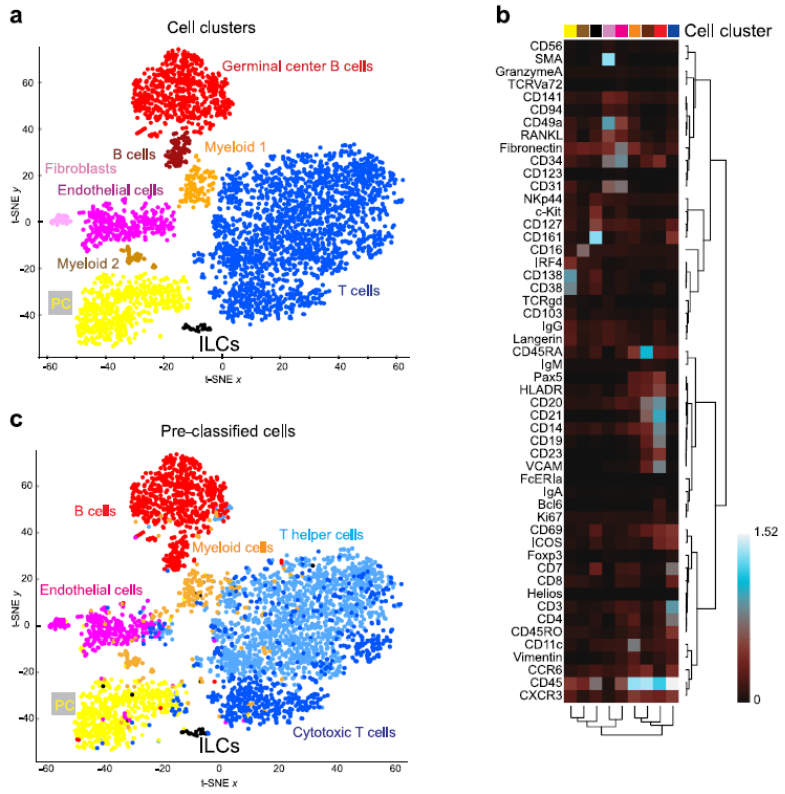

Figure 32 Dimensionality reduction (t-SNE maps) of (a) intensity profile clustering results vs (c) pre-defined cell types and (b) heatmap indicating sufficient overlap of manual and unsupervised clustering (Pascual-Reguant et al., 2021)

The automated and manual gating strategy are compared in Figure 32, validating the main cell type identification (germinal center B cells, B cells, myeloid 1/2, fibroblasts, endothelial cells, plasma cells, T cells). The expectation of rare cell type identification was confirmed as well, by the appearance of a spatially separated cluster cells bearing markers characteristic of ILCs. Cell type identification of the clustered cells was supported by heatmap representation of the used markers and found clusters. Heatmap representation promotes cell type subpopulation characterization as well, by visual co-expression of related cell and class signals. Increasing the number of clusters or reanalyzing only the desired populations can reveal additional relationships. Based on object features, like size, mean fluorescence intensity or location, the unique identifier makes a representation of clustered cells within the source images possible, supporting neighborhood analysis and visualization in tissue context.

3.4.3. Neighborhood Analysis

In addition to object identification, quantification and phenotypic classification of single-cell images, the spatial relationship of the objects in relation to neighboring signals improves the understanding of cell communication and its specific localization. In addition to that, multi-channel images of the same area support visualizing differences of similar stainings.

In order to investigate the spatial distribution of specific cells, a neighborhood analysis workflow was generated, applied and tested on stromal markers, i.e. CXCL12, BP-1, VCAM-1 and LpR. Images are part of MELC data sets used in 3.4.1.

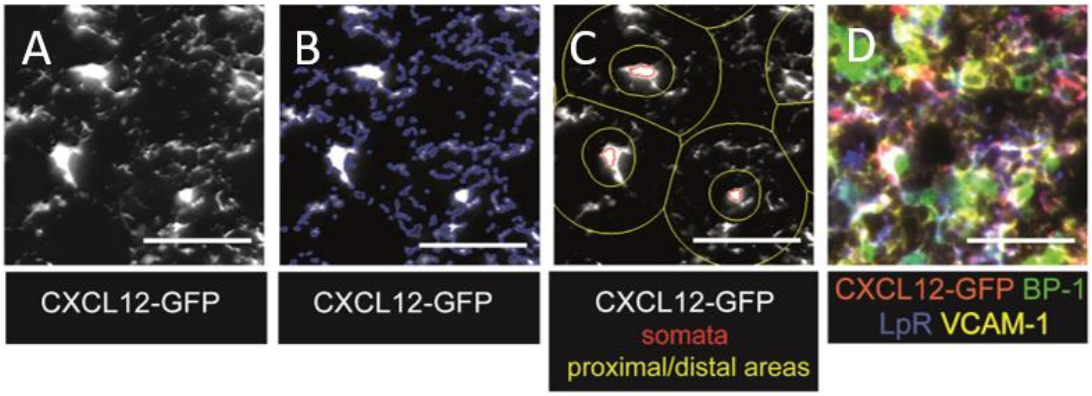

Figure 33 spatial distribution analysis workflow, (A) CXCL12 input image, (B) segmentation result, colored in blue outlines, (C) neighborhood regions (yellow) and central cell (red), (D) stromal matrix intensity distribution, scale bar 50 µm; (adopted from Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022

Here, CXCL12-producing stromal cells were used as central object to create two different circular regions with respect to the stromal cell soma: the proximal region with a radius of 0-10 µm and the distal region with a radius of 10-25 µm, as shown in Figure 33. Distance choices were made by assuming an average cell diameter of 10 µm and a calculated average half distance between adjacent stromal cell somata.

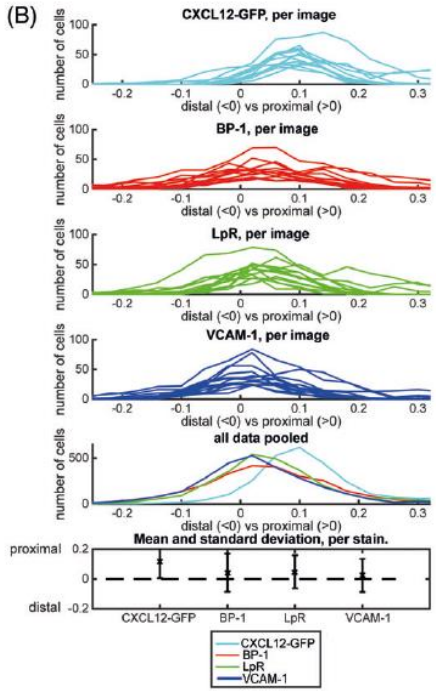

Figure 34 (top) stromal marker pixel density comparison, (bottom) colocalization analysis comparison; (adopted from Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

Within each proximal and distal region, the pixel density was calculated to compare signal contribution of the markers on a single cell level, twelve images in total were taken from

different samples. Here, pixel density is defined as the ratio of pixel count belonging to a certain object to the whole object-based area related pixel count. The difference from proximal and distal pixel density was calculated for every object and summarized for each image, visualized in Figure 34. Negative values highlight a larger distal signal distribution, positive values highlight a larger proximal signal distribution, and values around zero indicate no preferential signal distribution. In case of the GFP, BP-1 and LpR the main signal contribution occurs in proximal region, whereas signals from VCAM-1 seem more homogenously spread.

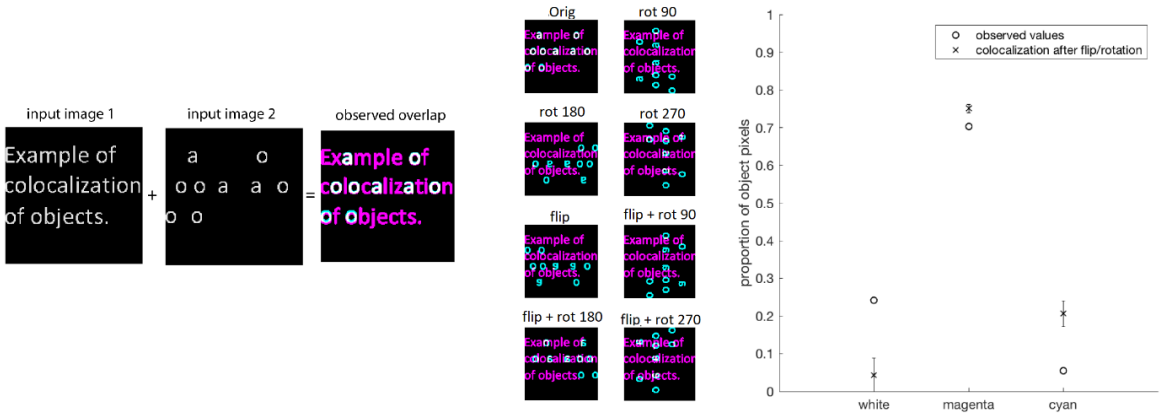

Figure 35 Working principle of randomness test. (A) colocalization of two binary masks, (B) calculated overlap in dependency to image transformation (rotation and flipping), (C) graph representation of calculated overlaps; (adopted from Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

As an additional test for dissimilar signal contribution of the different stromal markers, I applied colocalization analysis followed by a randomness test, to validate the results. Here, binary masks were created out of the segmented images (similar to binary stromal marker images, Figure 35A), which were used to calculate the overlap in dependency to a rotated or flipped version of the images (Figure 35B). Out of every possible image (stromal marker) combination, the overlap can be represented in a graph (Figure 35C).

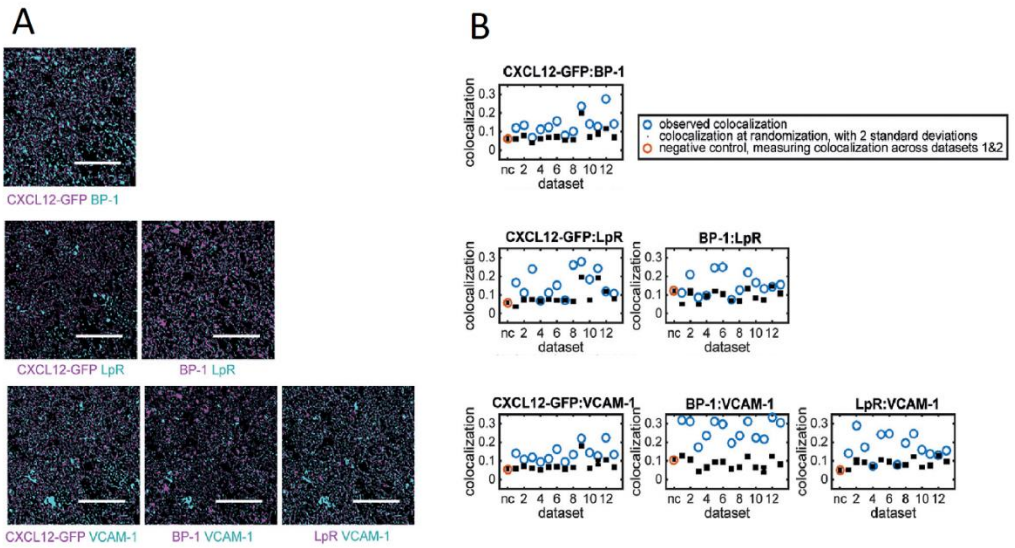

Figure 36 (A) example overlap images of one dataset from binary masks used for colocalization analysis, (B) summarized results plots of randomness test for all datasets; (adopted from Holzwarth, Köhler et. Al 2018) with permission from John Wiley and Sons (Sep, 05, 2022)

Even if all stainings are considered to be stromal markers, their heterogeneous spatial signal distribution could be shown within colocalization analysis (Figure 36A). Since calculated overlap (blue circles) is “non-zero”, partial co-expression indicates stroma localization. The subsequent randomness test verified that the observed colocalization was significantly higher compared to random positioning (black dots, Figure 36B). Negative control colocalization test (red circles, overlap of same markers from different data sets, Figure 36B) verified independent image data handling.

Altogether, these results indicate that the complex compartment of stromal markers within bone marrow is heterogeneously distributed, which was validated by colocalization analysis and the test of randomness.

Out of the tonsil data from 3.4.2, the rare ILC population was taken to test neighborhood analysis in between the semi-automatically found cell type objects, tissue related architecture signals from vessels and extra cellular matrix (ECM).

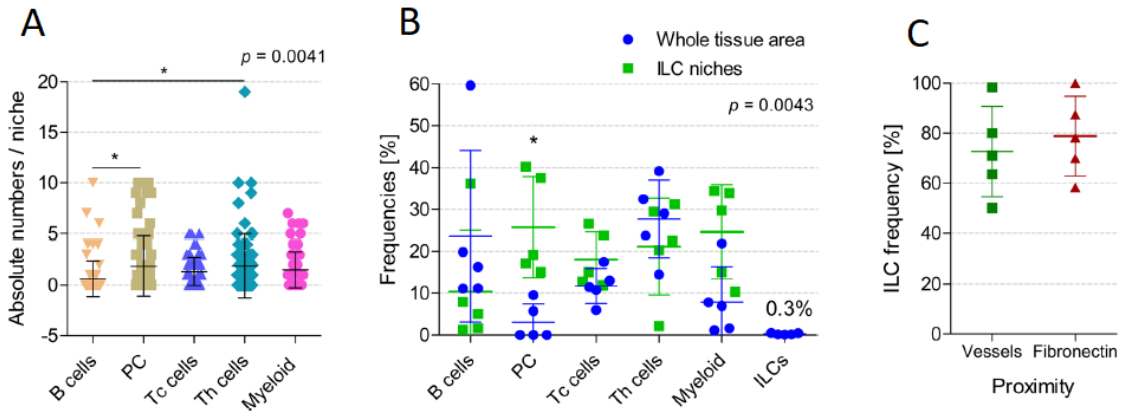

Figure 37 Neighborhood analysis of ILC population out of tonsil data set. A) absolute count on cell types of 74 ILC neighborhoods, B) separation of ILC neighborhood compared to whole area count on different cell types, C) localization of vessel and fiber signal in ILC neighborhood, (Pascual-Reguant et al., 2021)

Compared to the entire tonsil area imaged, ILC neighborhood analysis showed differences in frequencies from specific cell type quantification in the specific ILC niches (Figure 37). While there were higher amount of plasma cells (PC), T helper (Th) cells and myeloid cells found in the neighborhood, the numbers of B cells and cytotoxic T cells (Tc) were lower. Based on multi parameter spatial data, the vicinity of ILCs to vessels could be confirmed in 70 % of all ILC neighborhoods, in case of ILC in near distance to fibronectin fibers 80 % of all cases.

In summary, with the help of neighborhood analysis the micro-environmental characterization of specific cell types in a spatial domain extended colocalization analysis and showed the need of multi parameter image data, enabling cell communication observations.

3.5. Spatial Transcriptomics data analysis

Within a tissue remodeling study of human lungs, caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), 12 post-mortem samples were measured by ST. Along disease progression, these 12 samples were classified by disease duration into four groups, namely control (non-COVID-19-related pneumonia), acute (up to 15 days of disease duration), chronic (more than 15 day) and prolonged (in between 7 to 15 weeks).

3.5.1. Preprocessing ST data

After the sequencing, the data was demultiplexed and transferred into fastq format. In combination with the aligned microscopy images sequence, fiducial detection and unique molecular identifier (UMI) counting was performed via the software “Space Ranger”, referenced by human transcriptome (GRCH38-2020-A). Tissue detection and alignment were performed manually via the interactive software “Loupe Browser” before, since automated detection and alignment are not implemented for multichannel/multilayer fluorescence microscopy images.

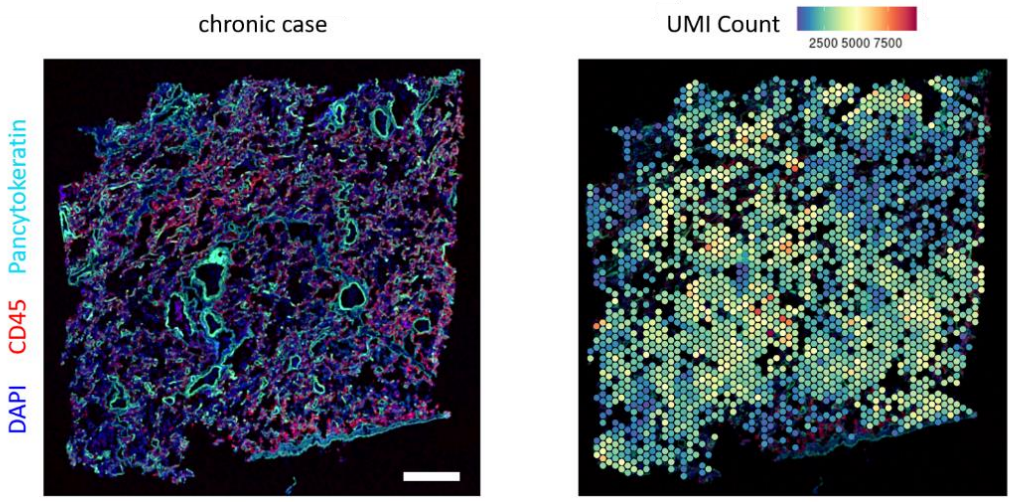

Figure 38 depicted images of a chronic case, (left) LSM image of human lung sample, (right) unique molecular identifier (UMI) count per spot, blue low UMI count, red high UMI count, scale bar 1mm

For each individual area captured, there was a results folder, some automated analysis, a web summary, and a “*.cloupe” file to interactively browse the results in the “Loupe Browser”. In the Loupe Browser the UMI count in every spot can be visualized (Figure 38), as well as the general quality statistics. Besides automated analysis, the most important files are those containing the featured barcode matrices, aligned spot positions and related images, which are required for downstream analysis.

3.5.2. Data Integration

Sample heterogeneity and technical artifacts, such as sequencing depth or detected gene counts make every single ST experiment unique. In order to investigate all 31801 found barcoded spatial spots covered by tissue at the same time, data integration was required. For this purpose an R script was written with the support of the Seurat package (Hao et al., 2021), which is a toolkit allowing single cell genomics analysis, especially with spatial related data.

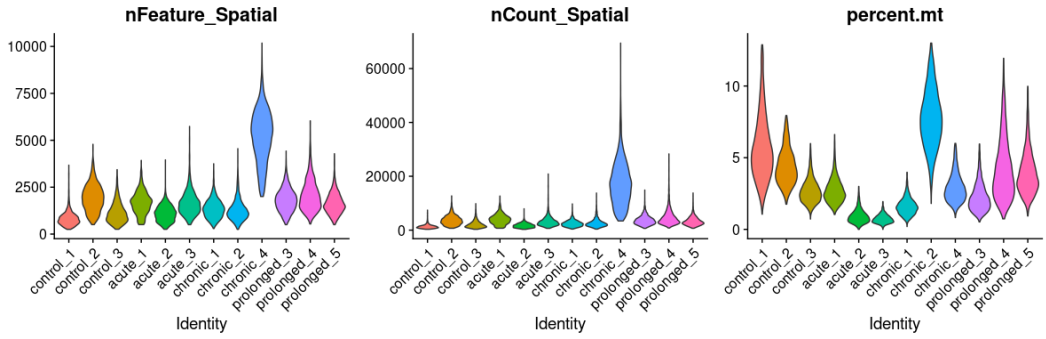

Figure 39 Violin plots of total gene count averaged per spot (nFeature), total amount of molecules averaged per spot (nCount) and relative mitochondiral gene amaount per spot [%] averaged (percent.mt) in dependency of samples, data already filtered

Filtering at the beginning ensured further analysis on only differentially expressed genes, where every spot contained 250 genes and less than 13 % of mitochondrial content (Figure 39). For every sample single spot variability was stabilized by SCTransform normalization (Hafemeister & Satija, 2019) and related captured areas as well. In this way, the local expression values were comparable to each other and true biological spot variation remained independent of the absolute number of genes or sequencing depth. Based on median variable feature rank across all datasets, 3000 top selected features were used to integrate data.

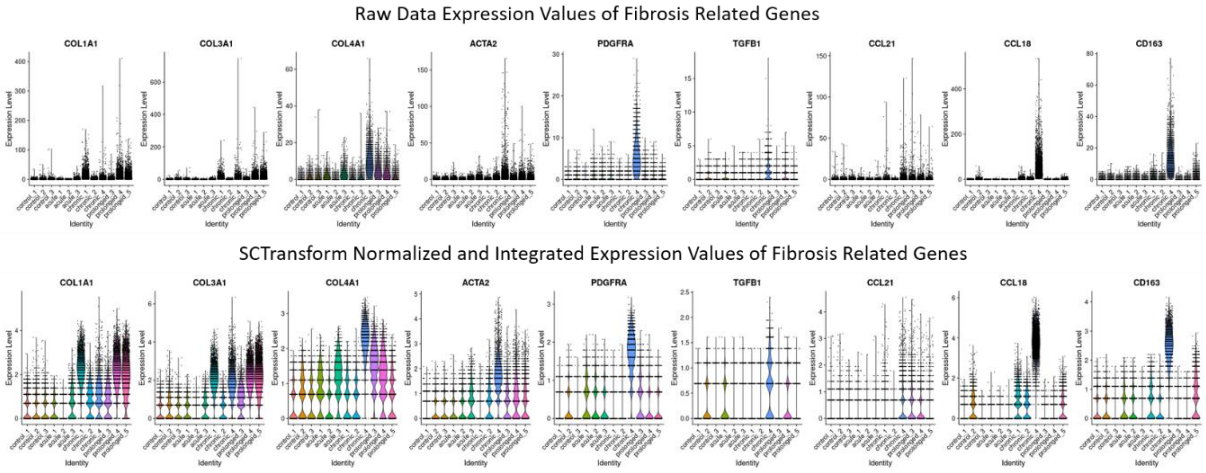

Figure 40 Raw data gene expression in comparison to SCTransform integrated gene expression of prominent fibrosis related genes

The expression values for prominent fibrosis related genes were used to visualize effect of SCTransform normalization and integration compared to raw data set in dependency of their disease sample identifier (Figure 40). At this stage every single sample can be interpreted as acquired with same sequencing depth. An increase of these fibrotic genes count along disease progression during SARS-CoV-2 infection supports hypothesis of ongoing lung fibrosis.

3.5.3. Dimensionality reduction and clustering

The amount of data from ST experiments makes manual single spot investigation time demanding and possibly biased. Therefore, dimensionality reduction and clustering were applied as an automated, objective and reproducible analysis pipeline, which was added to the integration R script to perform cell type identification and related gene contribution investigation of single spots.

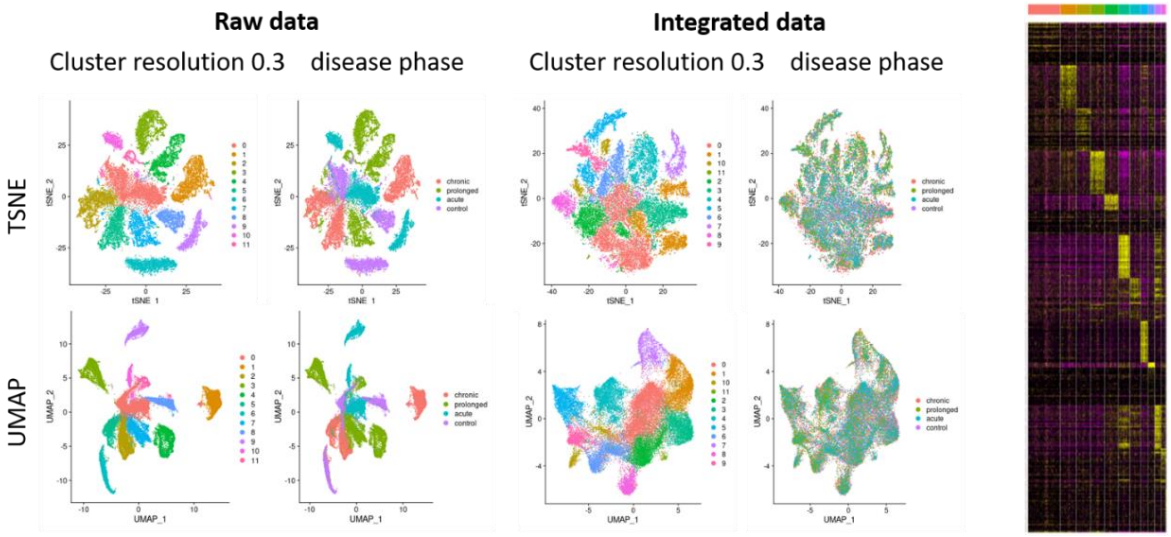

Figure 41 dimensionality reduction (TSNE and UMAP) applied to raw and integrated data. Color code of clusters in left panel from "Raw data" represents each of the 12 samples. On the right side of "Raw data" and "Integrated data" panel color code represents the four disease phase related samples. Color code of left panel "Integrated data" represents 12 major cell types as in heatmap (right).

Principle component analysis (PCA) reduced the normalized gene expression matrix dimensionality down to the top 50 components, which were used for shared nearest neighborhood (SNN) clustering initialization, followed by t-distribution stochastic neighbor embedding (t-SNE) and uniform manifold approximation and projection (UMAP) visualization (Figure 41). Along the various cluster resolutions and nearest neighbor settings calculations, heatmap representation was used to determine final values (resolution set to 0.3 at 30 PCA dimensions) to identify the 12 major cell types within the data. Because of the high gene count, only the top 25 ranked differentially expressed genes of every found cluster were taken, and displayed on the heatmap representation.

Both dimensionality reduction visualizations demonstrated the need for integration. In case of raw data, the differences between samples were found and clustered. In case of integrated data, similar expressed gene expression profiles were clustered together, as confirmed by heatmap representation. Compared to t-SNE, UMAP calculation was faster and preserved inner data structure at default values better. With a higher perplexity value, t-SNE could achieve similar data structure visualization, but at the expense of computation time.

3.5.4. Gene set enrichment analysis

Gene set enrichment analysis (GESA) is one way to analyze gene related diseases in a compact way. Here, a whole set of a priori defined genes that drive a disease or tissue state are tested and compared simultaneously, in this specific case along the four disease progression groups of SARS-CoV-2 infection.

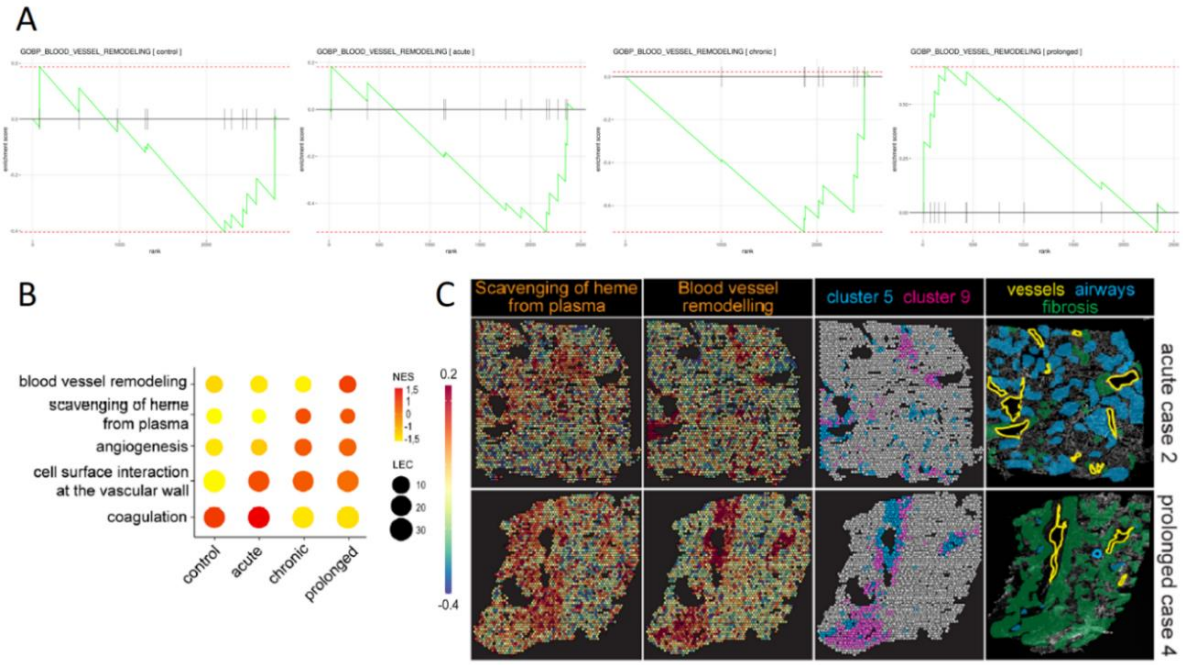

Figure 42 Gene Set Enrichment Analysis (GSEA) of Spatial Transcriptomic (ST) data. (A) Enrichment plots of GOBP_BLOOD_VESSEL_REMODELING pathway along disease progression (control, acute, chronic, prolonged). (B) Dot plot of selected pathways, color coded in their normalized enrichment scores (NES) and leading edge counts (LEC). (C) localization of NES from two pathways within example images of two disease stages in combination with related clusters found and annotation of vessels, airways, fibrosis areas (adopted from Mothes et al., 2022)

First GSEA was applied only on the expression data from all samples grouped by disease phase, resulting in four enrichment plots for every pathway, e.g. shown in Figure 42A were blood vessel remodeling gene set was taken. The highest enrichment scores (ES) (peak value of the plot) were normalized and combined in a dot plot (Figure 42B), allowing multiple pathway observations on a small visualization scale. Dot sizes were adjusted to the number of leading-edge counted genes (number of genes until peak of ES is reached). Within the tissue remodeling study, the up- or down regulation of gene sets could be found. Out of the four

example pathways in Figure 42B, the disease phase dependent tissue remodeling could be explained. We found that the broad dysfunction of the endothelial barrier at the beginning of the infection leads to vessel remodeling attempts linked to progressive tissue fibrosis.
Since the ST data also had spatial parameters, single sample gene set enrichment analysis (ssGSEA) was applied to localize pathway based enriched spots within the microscopy images (Figure 42C). Here, the comparison of the two disease phase groups acute and prolonged showed enriched expression in clusters 5 and 9, which was verified by annotation of specialist pathologists. Cluster 5 spots contained transcripts, including highly expressed actin alpha 2 (ACTA2), indicating that they were located around medium-sized vessels. Cluster 9 spots contained the highest expression of transcripts related to endothelial identity and complement activation. Due to fibrosis in the annotated areas, these results suggested that fibrosis starting in these vessel remodeled areas.
Overall applied GSEA and ssGSEA supported the identification and localization of gene set driven disease pathways within ST experiments.

3.6. Added value of correlative MELC and ST analysis

Within the same tissue remodeling study of human lungs, caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), consecutive slides of the same 14 post-mortem samples were also measured by a 44 markers panel MELC, 32 FOV. The samples were split by disease progression into four groups as well, namely control (non-COVID-19-related pneumonia), acute (up to 15 days of disease duration), chronic (more than 15 day) and prolonged (in between 7 to 15 weeks). To obtain both protein and transcriptional information, we employed a combination of MELC and ST that complement each other, on serial sections. Thereby, MELC provides a single-cell resolution using a panel of fluorochrome-conjugated antibodies that specifically bind to proteins of interest, whereas ST enables higher throughput to capture the whole transcriptome employing spatially resolved barcoded mRNA capture probes that hybridize with tissue sections. Both techniques have their own advantages and limitations, and the combination of these two techniques allowed us to gain a more comprehensive understanding of tissue remodeling in SARS-CoV-2 infected human lungs.

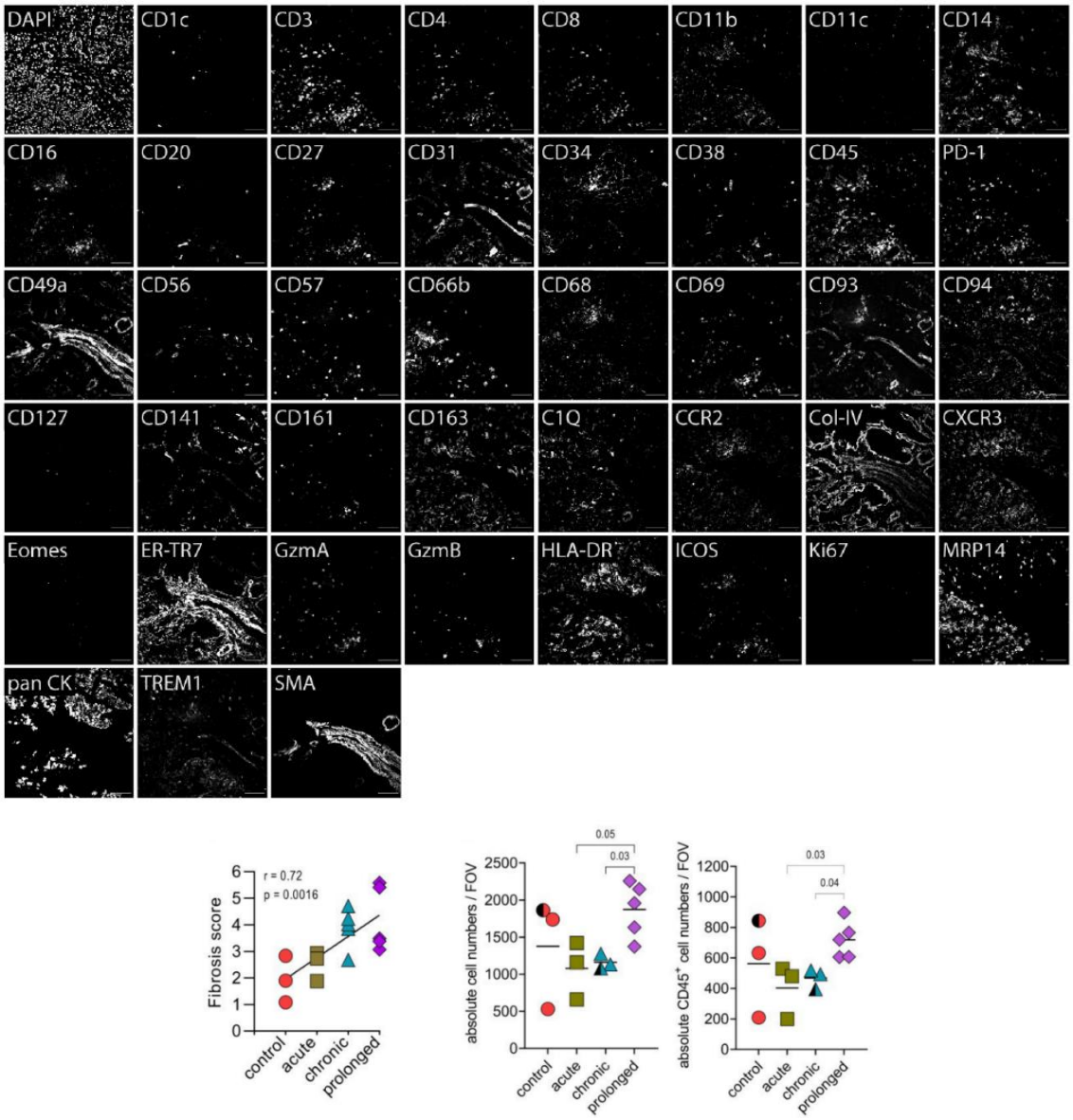

Figure 43 marker panel overview of one representative sample and cell quantification (with additional Kruskal-Wallis statistic), fibrosis score obtained by pathohistological examination of several lung sections from each donor based on Elastica van Gieson staining with trend line via nonparametric Spearman correlation; image size 665 µm x 665 µm; (adopted from Mothes et al., 2022)

After image acquisition and image preprocessing, cells were segmented and quantified (Figure 43). An increase of fibrosis score, total cell numbers and immune cells (CD45 positive cells) were observed in prolonged cases. As before, mean fluorescent signals were taken for unsupervised clustering and dimensionally reduction representation for further cell type identification via heatmap visualization. Instead of t-SNE, uniform manifold approximation and projection (UMAP) was used, because of the calculation speed and a better preservation of the data's global structure within the dimensionally reduction map. Furthermore, all analysis steps after segmentation and mean fluorescence intensity measurement were combined and applied in the previously created ST analysis pipelines within "R". Here the data integration

scripts from ST experiments were adapted to MELC data, allowing for visualization, functionalities and comparisons in the same environment.

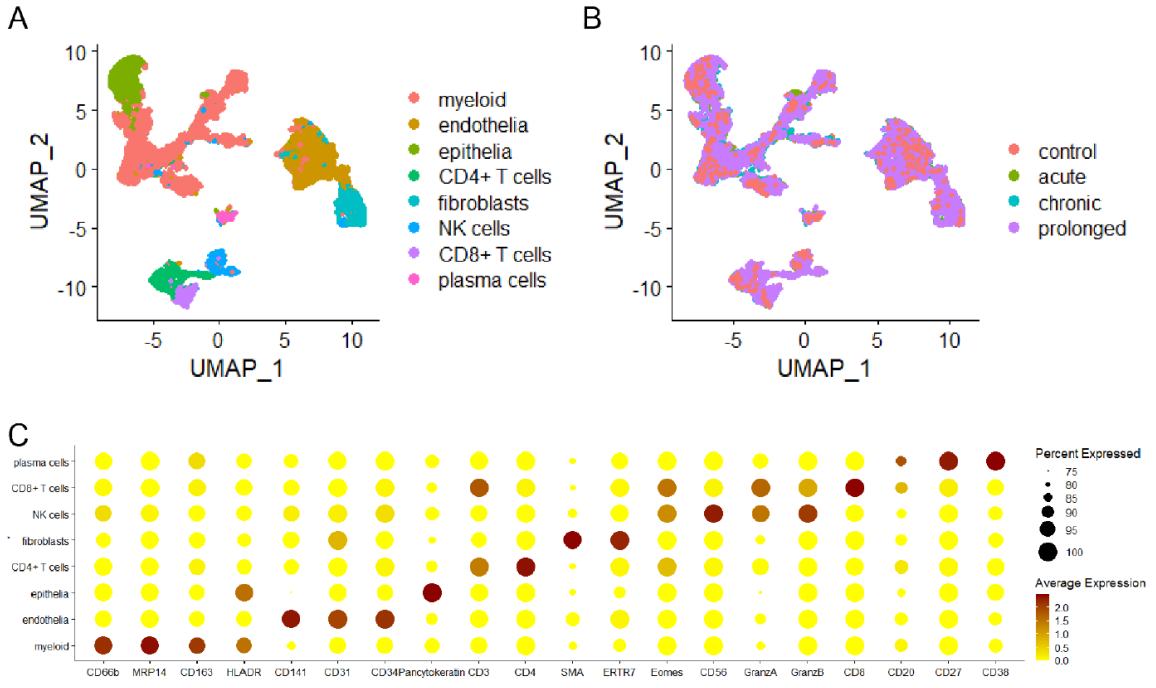

Figure 44 Uniform Manifold Approximation and Projection (UMAP) of identified cell types (A) and disease groups (B) out of 14 samples imaged via MELC. (C) Dot plot representation of identified cell clusters along marker profile expression (adopted from Mothes et al., 2022)

The main clusters found are shown color coded in Figure 44A. Additional to cell type identification, group independent data handling was tested and visualized by Figure 44B, where the same UMAP was used for disease group coloring. Besides heatmap representation, dot plots represent not only the existence of desired markers in clusters and their average expression level by color, they visualizes the expressed frequencies in size as well (Figure 44C). Therefore, the main contributing markers considered to their clusters verified the data integration.

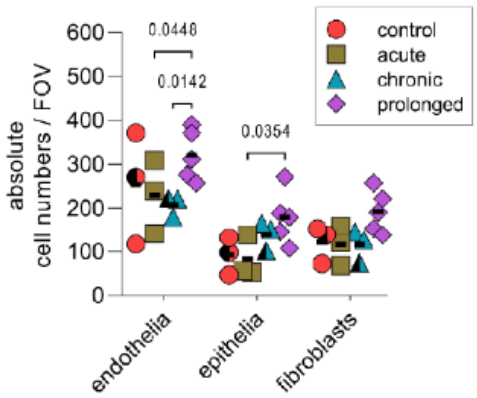

Figure 45 Quantification results of endothelial cells, epithelial cells and fibroblasts inside the found clusters dependent on disease progression. Data from 32 samples analyzed by two-way ANOVA with Fisher's LSD test, (adopted from Mothes et al., 2022)

With respect to clustered cells, quantification of absolute cell numbers (Figure 45) showed an increase of endothelial cells, epithelial cells and fibroblasts along disease progression.

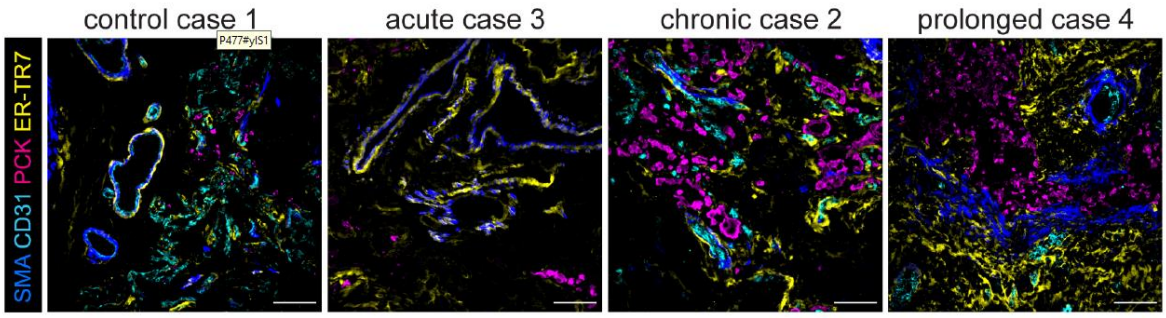

Figure 46 representative MELC images from smooth muscle actin (αSMA) in blue, CD31 in cyan, pancytokeratin (PCK) in magenta and ER-TR7 in yellow in an example field of view (FOV) in lung tissue at different disease stages, scale bar 100 µm (Mothes et al., 2022)

In line with the increase of fibrosis associated cells, stainings of αSMA and ER-TR7 showed an allocated clumping and higher contribution of marker specific signal within later disease stages on a single cell level (Figure 46). Epithelial tissue remodelling caused by structural changes were visualized by PCK staining.

Altogether, applying ST analysis to MELC data confirmed and disclosed a dysregulated immune response due to COVID-19 infection, resulting in a cascade of processes initializing and ending in tissue remodeling. The combination of both spatial multiplex methods benefited from the created R script, which ranged from a high number of parameters (genes) in case of ST, to single cell resolution in case of MELC.

3.7. LSFM image data analysis

The low numerical aperture cylindrical lenses architecture creating the light sheet in this microscope results in a much longer light path, which increases the probability of light scattering through the differentially distributed refractive index and related tissue density. This also increases the likelihood of artifacts. This becomes evident in femoral and tibia bones, in the context of a bone regeneration model, where characterization of fluorescence-stained cells in contrast to their unspecific signal was affected.

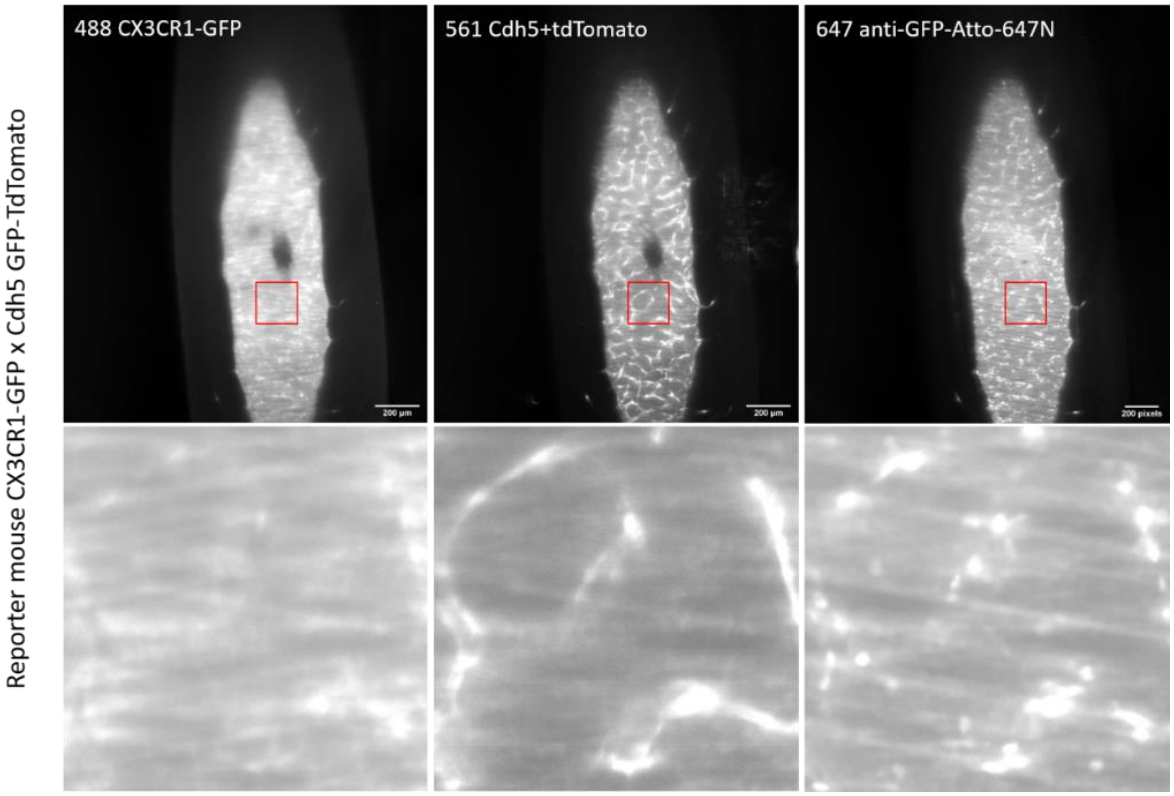

Figure 47 LSFM measured tibia from reporter mouse illustrates stripe artifact along all fluorescence channels, red boxes indicates enlarged area (images at the bottom), scale bar 200 µm, depicted images from layer z=275

The images with enhanced stripes in Figure 47 were taken from a test data set, where a CX3CR1-GFP x Cdh5 GFP-TdTomato reporter mouse's tibia was imaged at a voxel resolution of 0.755x0.755x3.000 µm³, final image size 2560x1280x719 px³ for each fluorescence channel. Stripes occurred along all six illumination directions, being related to the angles of the three laser beams that form the light-sheet from left and right sides. Due to the heterogeneous tissue-based refractive index distribution and related refractive indices, the stripes showed no regularity, and appeared dependent on tissue layers at different positions.

In order to investigate the staining performance and related image quality along different measurements of young and old mice within the bone regeneration model, signal to noise ratio (SNR) calculations should be used, but this required stripe free images. For this, a stripe suppression workflow, which preserves the main signal, had to be developed. Morphological filter operations would have handled the image locations differently depending on existing stripes. Because of this, image processing was shifted to the Fourier domain, since it was expected that repetitive stripes would be associated with specific frequencies that are dominant in the Fourier spectrum.

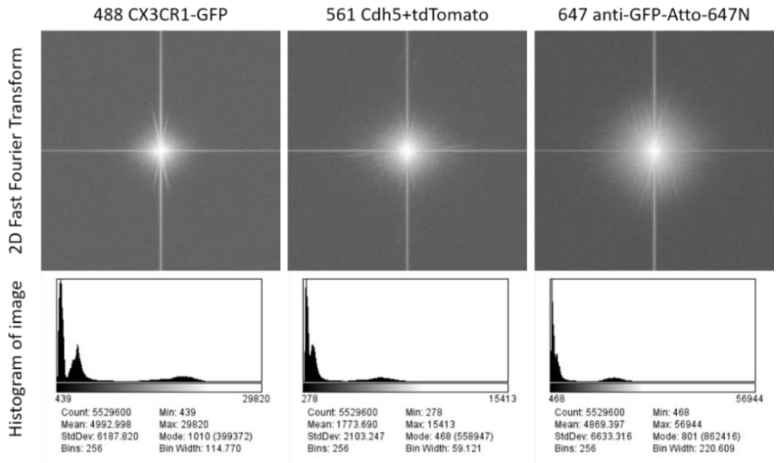

Figure 48 2D FFT and histogram of the depicted images at layer z=275, calculated and displayed in ImageJ/Fiji

Numerous stripes had similar angles. After two-dimensional fast Fourier transformation (2D FFT), stripes contributed frequencies highlighted in the same angular direction (Figure 48). Due to FFT calculations, the angles were rotated by 90°. Histograms of the different fluorescence channel images showed the three areas of the acquired tissue sample. Low values represented the overall background signal. The midrange intensity values or second peak illustrated the hard tissue (cortex) surrounding the soft bone marrow tissue. The last very broad spread peak contained all the tissue related signals (endosteal bone and bone marrow), including the stripes information, which caused a broaden distribution.

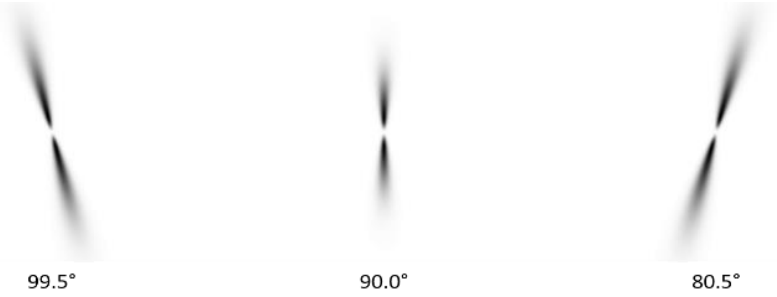

Figure 49 filter masks for each stripe angle

Stripe suppression in the Fourier domain was created based on the device-related properties and the measured angles within the image data. The central area of the FFT image contain the most important low frequencies to reconstruct the image. This region needs to be untouched by the filter. However, high frequencies are also important to recover edges within the images. Due to similar signal height of the outer frequencies in the FFT images, a band pass filter design was chosen. Filter masks are represented in Figure 49.

In case of the low frequency band, the size of a circular area radius was calculated based on a Gaussian fit of the 45° line values of the FFT image. These values minimally contributed to the stripe-associated frequencies. The estimated standard deviation (sigma) was taken as the diameter for this region. In case of the high frequency band cut-off, the doubled standard deviation width estimation of a Gaussian fit along stripes values were used. In contrast to single

line selection and multiplication at the stripe angles with zero, a continuous Gaussian decreasing line function with increasing sigma in the direction of image boarders was used to avoid the aliasing (Gibbs phenomenon) artifacts after back transformation in real space, inverse FFT. The line width was adjusted to the light-sheet size, by increasing sigma the function recovered the scattered angle tolerance. All measurements and values are taken from the absolute FFT magnitude images and scaled logarithmically. Each angled stripe suppression was performed sequentially.

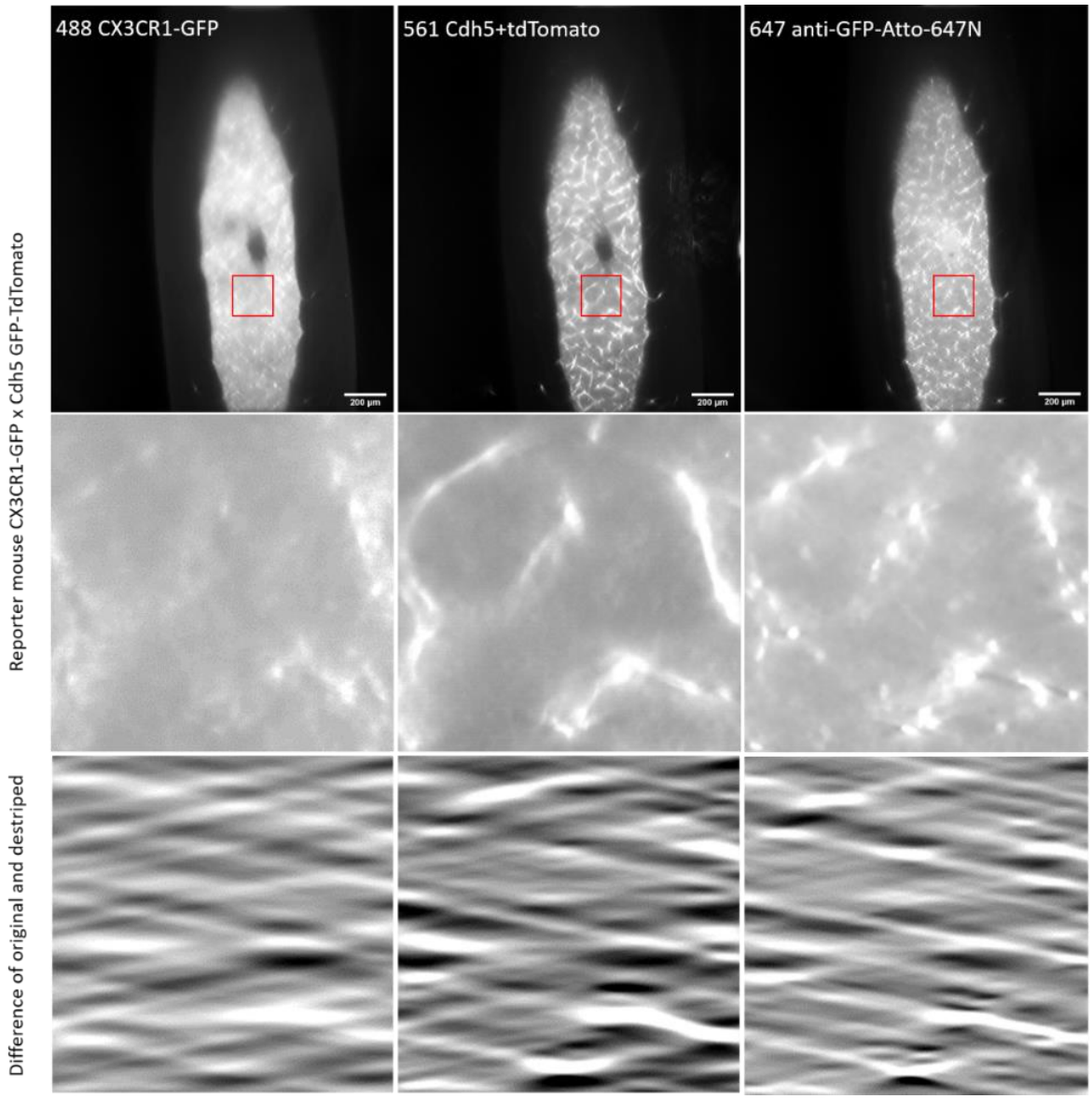

Figure 50 LSFM measured tibia from reporter mouse after FFT destriping approach along all fluorescence channels, red boxes indicates enlarged area (images at the bottom show striped illumination function), scale bar 200 µm, depicted images from layer z=275

The test images showed less striped signal within the tissue after the application of the FFT destriping approach (Figure 50). The subtraction difference between the original and the destriped images represented the striped illumination functions.

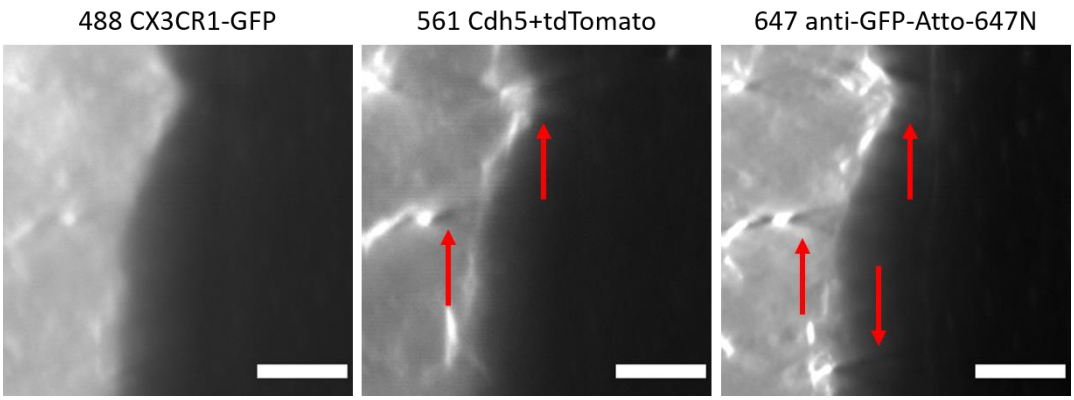

Figure 51 Enlarged area around the tissue sample's edge show additional stripes by filter mask, scale bars 50 µm

A closer look to the tissue sample's edges showed additional stripes that were produced by the FFT filter mask (Figure 51). They were found around high intensity structures as well. In case of the test image data set not only single angled stripes occurred, which resulted in a complex mixture of stripe signals due to higher harmonics of the original stripe frequencies. Application to other datasets was used to confirm the main working principle that provides stripe suppressed images.

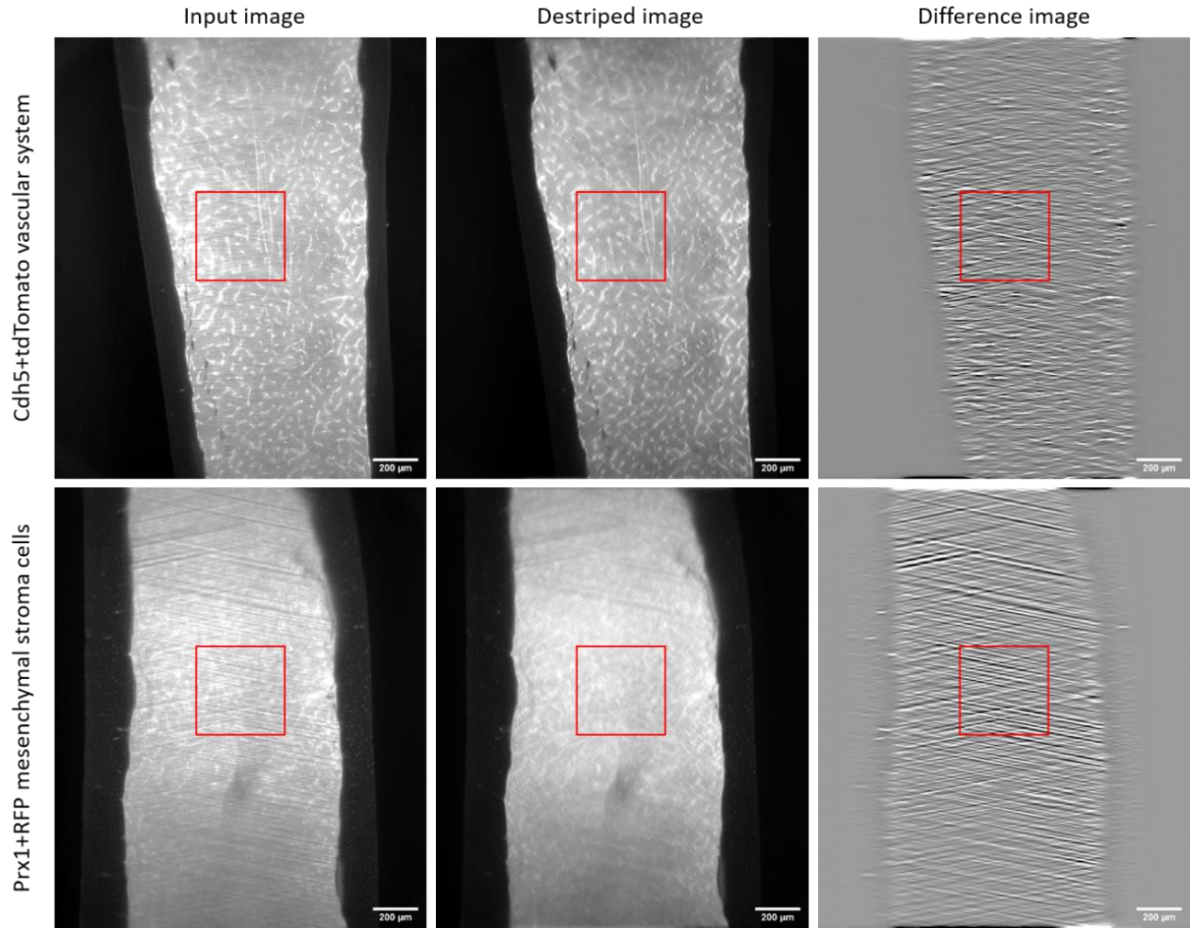

Figure 52 depicted images of LSFM measurements of two reporter mouse's femoral bone, scale bars 200 µm

Stripe suppression by directional FFT filtering of the LSFM measurements of femoral bone from two different reporter mice (Cdh5+tdTomato and Prx1+-RFP) in Figure 52 with predefined parameters showed a better destriping result compared to the tibia data set. Stripes of several micrometer thickness (top area of Figure 52 within Prx1+RFP data set) were not fully suppressed, which resulted in a smoothed long waved illumination function. However, signal preservation along the central mean line along this illumination function was still given. Image data sets were acquired using the same microscope parameters.

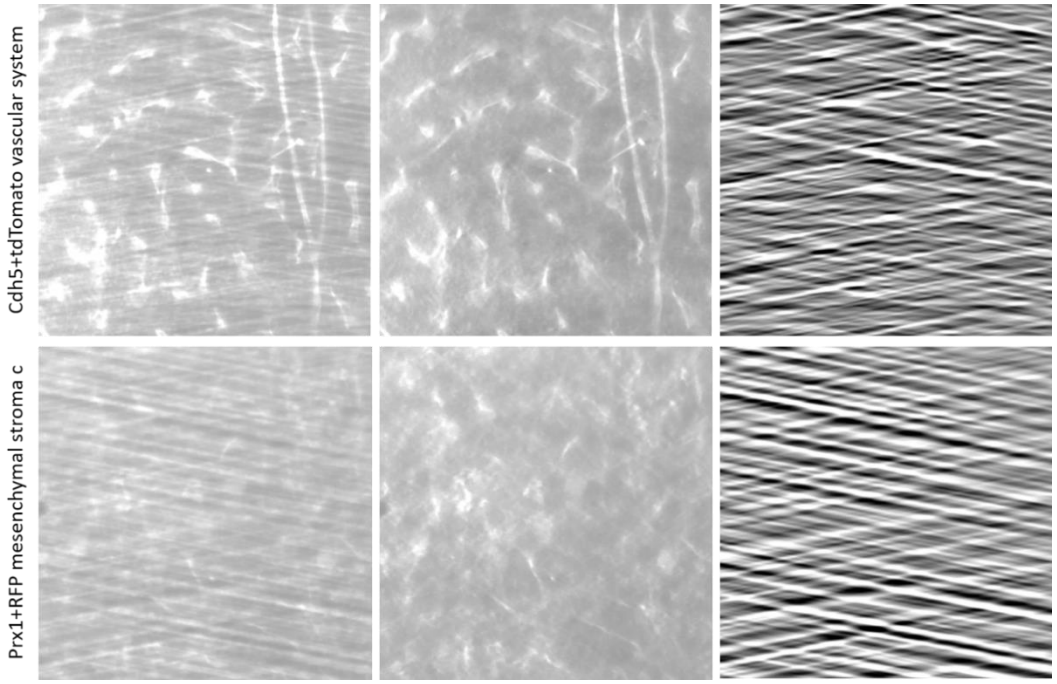

Figure 53 enlarged areas (387x387 µm²) of red boxed regions of previous figure

The relatively sharp stripe signal and constant orientation of the stripe signal resulted in a better differentiation between the imaged tissue sample and the artifacts. The underlying signal of cellular and vascular structures were preserved (Figure 53). The Gibbs phenomenon around the edges was still present, but to a much lower extent. The mean intensity background signal distribution outside the tissue showed no clear difference after FFT filtering. These results indicate a relationship between the applied filter mask mean amplitude offset dependency. The image destriping performance was considered to be sufficient for further image analysis, as subsequent SNR calculation or object identification was no longer influenced by the stripe artefacts.

In order to calculate the SNR, object identification was required to differentiate signals from structural components, tissue background or area outside the sample. Therefore, a random forest algorithm within the application ilastik was trained to classify pixels by manual annotation in the destriped test images. The classified pixels were used to identify necessary mean object intensities and standard deviation values of the images. In addition, the destriping approach performance could be quantified.

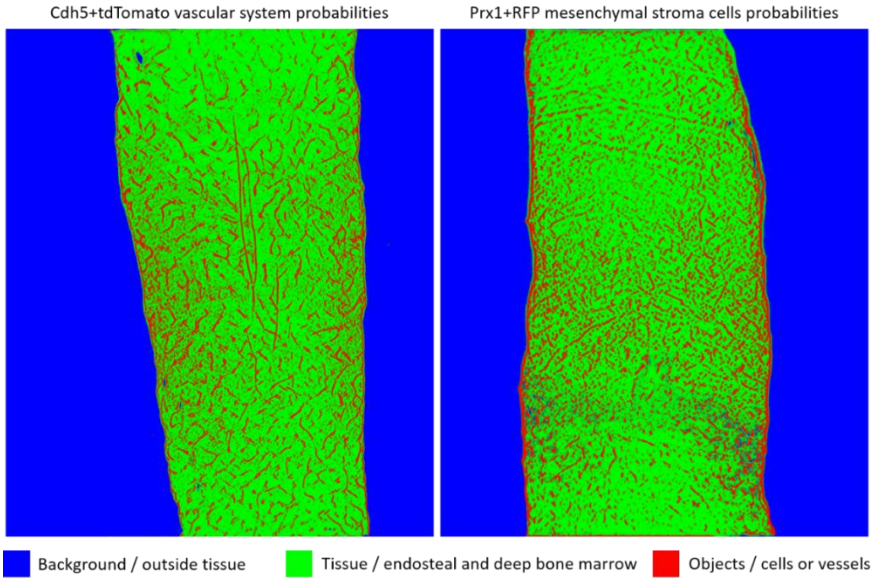

Figure 54 pixel classification result masks from reporter mice, where blue indicates probability of outer tissue signal, green represents tissue area and red shows main vascular or cellular probability

Out of the generated probability maps (Figure 54), objects of the desired classes were created by simple thresholding. Because of the three classes (background, tissue, objects), pixels of the related object classes were taken at a probability higher than 66 % and used for mean intensity (μ) measurements or standard deviation (σ) calculation in the destriped images and original input images, respectively. Due to average SNR calculation, no further object separation of single cells or vessels was required.

Table 5 PSNR, SNR and SBR measurements from original and destriped images

sample	layer	PSNR_orig	PSNR_destr	SNR_orig	SNR_destr	SBR_orig	SBR_destr
160101	Z0250	13,61	13,45	6,92	6,87	1,30	1,29
160101	Z0300	13,74	13,50	7,29	7,25	1,29	1,27
160101	Z0350	17,46	17,08	7,66	7,65	1,22	1,21
160101	Z0400	19,16	18,74	7,63	7,63	1,24	1,22
160101	Z0435	18,81	18,43	8,06	8,10	1,24	1,22
160101	Z0450	17,62	17,34	8,44	8,49	1,24	1,22
160101	Z0500	12,97	12,81	7,03	7,01	1,27	1,26
AVG		**16,20**	**15,91**	**7,58**	**7,57**	**1,26**	**1,24**
stdDev		**2,46**	**2,37**	**0,51**	**0,54**	**0,03**	**0,03**
sample	**layer**	**PSNR_orig**	**PSNR_destr**	**SNR_orig**	**SNR_destr**	**SBR_orig**	**SBR_destr**
170717	Z0150	6,71	6,59	6,74	6,89	1,16	1,13
170717	Z0175	10,52	10,31	5,54	5,48	1,13	1,10
170717	Z0200	12,58	12,21	5,25	5,22	1,10	1,07
170717	Z0225	13,96	13,62	5,36	5,34	1,09	1,07
170717	Z0235	13,38	13,09	5,31	5,32	1,08	1,06
170717	Z0250	13,31	12,93	5,15	5,16	1,06	1,04
170717	Z0275	11,48	11,03	6,11	6,10	1,04	1,02
AVG		**11,71**	**11,40**	**5,64**	**5,64**	**1,10**	**1,07**
stdDev		**2,32**	**2,24**	**0,54**	**0,58**	**0,04**	**0,03**

Calculation of the features:

$$Peak\ Signal\ to\ Noise\ Ratio\ (PSNR) = \frac{\mu_{signal}}{\sigma_{background}}$$

$$Signal\ to\ Noise\ Ratio\ (SNR) = \frac{\mu_{signal}}{\sigma_{tissue}}$$

$$Signal\ to\ Background\ Ratio\ (SBR) = \frac{\mu_{signal}}{\mu_{tissue}}$$

Several destriped images of the two data sets were depicted and used to calculate the signal relations based on the three object classes (Table 5). The measurements seem to be independent of existing stripes, indicating that stripe structures are a result of interference, illumination enhancement and the suppression around tissue's local mean intensity. Further SNR measurements at different disease phases (imaged tissue sample at different points in time in a bone regeneration model) showed no loss in signal performance. However, destriping was essential for object identification and related quantification, neighborhood analysis, or single voxel analysis.

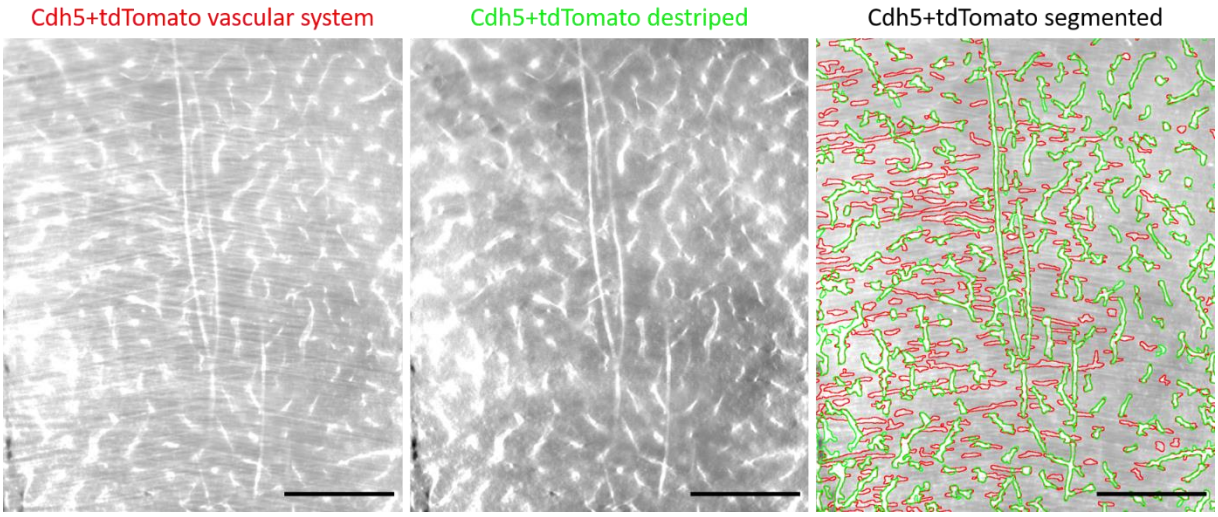

Figure 55 segmentation comparison of striped and destriped depicted Cdh5+tdTomato vascular image. Red outlines illustrate objects found in striped image, whereas green outlines represent objects found in destriped image, scale bars 200 µm

In order to show the influence of the stripes during segmentation, the top images of the vascular system and cells in both cases (striped and destriped) were segmented using the developed object identification workflow (Figure 55). Thereby, random forest training within the application ilastik was performed on each image case individually to guarantee best probability map calculation and increase segmentation performance. Due to similar signal of stripes and objects, results showed that objects found in the striped images cross real object boundaries, while objects were separated from tissue background signal in the destriped images. As shown in the SNR calculation, found objects can then be used for real signal measurements and further cell-object-based communication analysis.

4. Discussion and future prospects

Developments in recent years have highlighted the importance of image data analysis and associated reliable data pre-processing - a task that cannot be achieved without a thorough understanding of the physics behind the microscopic image. The principle of image acquisition in fluorescence microscopy is general and applicable to microscope-based methods such as MELC, LSM or LSFM, and results in a matrix-structured representation of the extracted features - the images. In addition, recent developments have made it possible to combine the detection of differentially expressed genes by sequencing with spatial information, such as ST. The information obtained from each device differs in system-dependent characteristics. Therefore, each analysis pipeline is unique to its device characteristics and conditions (Miura & Sladoje, 2019), so the pre-processing should be adapted to the data acquisition scheme, where physical aspects lead to image aberrations. However, images and their associated data provide a solid foundation for quantifying biologically relevant data.

In this thesis, image data from various microscopy techniques were used to create supervised- and unsupervised bioimage analysis workflows validating biological questions, which were conceptually developed on the improved MELC system. Independent of single cell or tissue structures, features such as morphology, signal distribution and localization are present in these multi-parameter data sets. Additional hardware-related image artifacts could also be identified, categorized, suppressed, or removed. Furthermore, the application of signal-dependent classification in MELC and ST-related data enabled linkage at the protein and transcriptional level in biological tissues. Conceptually, the bioimage data analysis workflows created in the different projects followed the same scheme: data acquisition, pre-processing, object identification, classification and representation of the results. Based on the biological question and available data, a possible object interaction realized by neighborhood analysis was applied.

4.1. Seeded region growing and DFT shape descriptors for analysis of microglia in LSM images

Compared to other acquisition methods that require a detailed artifact model description for image data preparation, LSM images within the dementia model could be pre-processed by maximum intensity projections to sufficiently enhance object details while reducing noise. Since these images were acquired with only one fluorescent channel, image registration was not necessary.

Simultaneously with improvement of the MELC system and the associated image analysis, a self-written “Seeded Region Growing” (SRG) algorithm as part of the “MELC_Evaluation_Toolbox” was used to extract microglial objects via GFP detected

expression from these LSM images. Due to the SRG algorithm used, the fine structures could be found better. With the objects, the complex shapes of the microglia in different aging mouse brain samples could be described by closed object contours, avoiding the use of the ImageJ/Fiji PlugIn “ABsnake” (Andrey & Boudier, 2007) as used in “SHADE” (Kriegel et al., 2018b), where contour features were smoothed due to the object shape fitting model. In addition, the PlugIn enabled the identification and extraction of all microglia and associated outlines in one image, whereas SHADE required individual object images. Even if the class Microglia is defined by the object itself, the loss of processes with increasing age was described using the DFT shape descriptors. Thereby the reduction of the complex shape information to a set of 20 coefficients was used to observe differences, store and represent the results in a compact way (Radbruch et al., 2017). Furthermore, these descriptors could also be used to train machine learning models for cluster analysis and group based information (Kriegel et al., 2018a).

4.2. Enhanced MELC analysis workflow for spatially automated phenotypic classification and cellular communication investigation

Among other things, MELC imaging is affected by mechanical tolerances and therefore requires image pre-processing. Image registration, compared to similar automated or manual imaging methods (Eng et al., 2022; Gut et al., 2018; Lin et al., 2015; Wählby et al., 2002), ensures that the signal contribution of each pixel detected by different stainings originates from the same location. Thereby, the application of the phase cross correlation (Guizar-Sicairos et al., 2008) using phase contrast images allowed image registration accuracy down to subpixel resolution, in this case by a tenth of the pixels size due to FFT based computation. Compared to segmentation based methods of fluorescence stainings, or those in ST data pre-processing (fiducial image alignment), which rely on pattern recognition, an increase of ten times higher accuracy was obtained. Artefacts originating from the excitation illumination function interaction with the tissue sample were also observed in MELC image data sets. Even when the updated lens and camera adapter were corrected for possible spherical distortions, the changing liquid level on the sample during the cyclic measurement caused additional optics-related light focusing and thus a change in the illumination function. These effects were eliminated by adapted “flat field correction” algorithm (Herman, 1980). Here, the subtraction of the previous bleaching image from the current fluorescence image removed resting signal and dark image offset, where regression based cubic spline interpolation was used to flatten the illumination function based on the bleaching image. Due to a self-designed and printed sample holder or sample preparation, possible tissue sample tilting or bending was corrected by an “all in focus projection”. Contrary to “extended depth of field” (Pertuz et al., 2013), standard deviation focus measurements were used to find the best tilted focal plane and project only the

signal related to the PSF of the system into a final two-dimensional image. In this way out of focus regions did not contribute to the final projection.

In case of nucleated cells or structural signal distributed over the image, the application “CellProfiler” was introduced as modularized segmentation tool, which allowed microscope users upon training the interactive creation and fine tuning of identification and measurement pipelines. Based on the image pre-processing, a watershed segmentation by manual thresholding was sufficient, which led to similar counting results in terms of variance compared to trained raters (Holzwarth et al., 2018). Nevertheless, the possibility of longer measurements due to external light source usage and the extension of fluorescent markers along MELC experiments, an increase in intensity distribution variance was observed. In order to improve segmentation accuracy related to signal variability, the application “ilastik” was introduced (Schapiro et al., 2017), to pre-classify the image pixels in relation to nuclei, extra cellular matrix, background or other structures. In addition to ilastik, the imageJ/Fiji PlugIn “Trainable WEKA segmentation” (Arganda-Carreras et al., 2017) should also be mentioned, which works in a similar way in principle, but was not used here. By conversion of real valued measurements to featured variables through different sized image filters, a random forest model could be trained, where class labels were set manually by drawing based on the desired images of a single fluorescence channels. In case of fluorescence images representing same class regions by staining, summed images were used to increase the probability of detection. Using the resulting probability maps as input for image segmentation in CellProfiler, the object count within the images could be increased by about eight percent. The additional resolution improvement of the MELC system by 60% laterally and 250 % axially, achieved by replacing the camera and objective lens also supported the segmentation performance.

In case of objects from MELC, data sets contained distributed information along several protein levels, which enabled the phenotypical classification of similarly shaped objects. It thereby allowed cell differentiation at a higher degree, compared to other physical limited fluorescence microscopy or morphology based methods (Fekri-Ershad, 2021; Kriegel et al., 2018b). The first steps in cell classification development were adopted from the idea of binary decision trees and were part of the “MELC_Evaluation_Toolbox”. Here, colocalization of signals was used to distinguish individual cell objects. In addition to the identification of colocalizing objects, mean or median fluorescence intensity measurements of the objects in desired images of various markers using CellProfiler increased the level of subpopulation differentiation, as shown by others (Gerner et al., 2012; Schapiro et al., 2017). In accordance to FACS analysis, manual gated signal evaluation and object classification within microscopy images lead to identification of major cell types in tissue samples of mouse, e.g. bone marrow, and human, e.g. tonsil, but also in rare cellular subsets such as ILCs, which occurred in 0.15-0.50 % of the analyzed data (Holzwarth et al., 2018; Mothes et al., 2023; Pascual-Reguant et al., 2021). With the constantly

growing number of possible marker combinations, the individual step sequences for manual gating also grew. In order to circumvent this, dimensionality reductions such as PCA, t-SNE and UMAP followed by clustering analysis were used to create an unsupervised, automated workflow reducing the risk of biased data handling and project the complex structured data in to a two dimensional map. The resulting cell class clusters were still set by manual interpretation of experts, but could show in comparison to manual gated data sufficient overlap, especially in case of rare cell subsets, which were also promoted by heatmap representation revealing other cell subpopulations. In addition to single cell analysis including phenotypically classification and colocalization, neighborhood analysis was applied to extend the micro environmental characterization of specific cell types in a spatial domain, which enabled the investigation of cell communication processes. Thereby the heterogeneously distribution of stromal markers within mouse bone marrow could be shown (Holzwarth et al., 2018), as well as the near distance of ILCs to vessels and fibronectin fibers (Pascual-Reguant et al., 2021). Validation of the distribution results from applied neighborhood analysis were confirmed by a randomness test. In contrast to a previous method (Zehentmeier et al., 2014) in which individual objects in the microscopic images were randomly positioned multiple times, the approach along the Stroma marker and ILC projects used image transformations, such as rotation or flipping, as a combinatorial computational basis for possible colocalizations. In that way, calculation time could be decreased and the relation to other tissue sample images containing different stainings or objects could be shown.

4.3. Comprehensive workflow for object characterization and phenotypic identification in Spatial Transcriptomics

Objects from ST experimental datasets benefit from the workflow developed by MELC, including pre-processing and object characterization, which is even more necessary due to the number of expressed genes (Hao et al., 2021). It should be noted that the objects (spots) in ST were given by the method itself and could contain multiple cells due to their size (55µm in diameter), but also had to be aligned due to their unique barcoded fiducials.

In contrast to MELC datasets, where Min-Max, SNR or arcsinh normalization and transformation of mean-object-signals were sufficient for phenotypic classification along different measurements, ST data sets exhibited high variance not only per measurement but due to data acquisition per spot. For this reason, data normalization was performed via SCTransform to improve visibility of highly differentially expressing genes and remove outliers by harmonization of Pearson residuals (Hafemeister & Satija, 2019). Furthermore, normalization was required for whole data integration, which enabled the investigation of pathomechanisms in all 12 SARS-CoV-2 post-mortem human lung samples at a time. Besides the upregulation of endothelial and fibrosis related genes found in later disease phases,

ssGSEA was applied to localize pathway based enriched spots within the microscopy images suggesting that fibrosis starts in these vessel remodeled areas. ST analysis was confirmed by MELC and disclosed a dysregulated immune response due to COVID-19 infection, resulting in a cascade of processes initializing and ending in tissue remodeling (Mothes et al., 2023)

4.4. Improved object identification accuracy through FFT-based filtering for stripe artifact suppression in LSFM measurements

The spatial to frequency domain conversion was also used for stripe artifact suppression in LSFM measurements, because in the centered frequency domain, similarly oriented stripe signals accumulate along the axes of the incident laser beams and can be directly filtered while preserving the main signal. In contrast to compressed sensing methods (Schwartz et al., 2019), where eliminated frequencies were reconstructed by total variation minimization, the filter was designed, based on known device properties and estimated tissue sample interactions. Other “wavelet”- or “non-subsampled contourlet” transformation based stripe elimination methods (Kang et al., 2015; Li et al., 2015; Liang et al., 2016; Münch et al., 2009) follow the same concept in core, but with prior consideration of the transformation axes and aliasing effect reduction by Gaussian weighted distributions. Furthermore, the calculation time increased to several minutes per image due to the different transformation steps, which led to an additional exclusion criterion. However, these methods showed no beneficial increase in stripe suppression compared to the self-designed FFT filter mask, which results allowed to improve segmentation performance.

4.5. Standardized analysis pipelines for data integration of multiple complementary spatial technologies - augmenting access to information

Overall, the pre-processing of the image data in all projects ensured the removal of artifacts and signal enhancement through the application of system-specific analysis pipelines related to physical image formation. Although the individual algorithms used differed in their application, they were all necessary for the subsequent image segmentation step.

In order to perform quantitative image analysis, the identification of objects within microscopy images was essential. Therefore, different segmentation tools and algorithms were applied to obtain single cell objects or structures from image data. Combining system specific pre-processed data, semi-automated feature conversion in ilastik and object identification in CellProfiler in one workflow enabled quantitative image analysis across projects (Mothes et al., 2023; Pascual-Reguant et al., 2021), and allowed users of different microscopes (LSM, LSFM, etc.) the chance of object related feature extraction (such as mean fluorescence intensity

values, location information or morphology related properties) upon a short image analysis introduction.

Once the objects were found, they had to be phenotypically identified. Using the workflow developed and adapted in "R", even rare subsets could be identified. The combination of the two spatial multiplexing methods contributed significantly. The advantages of both methods, the high number of parameters (genes) and the observation of the whole tissue in the case of ST with the single cell resolution at the protein level in the case of MELC benefited from the applied dimension reduction and cluster analysis. This also allowed simultaneous validation within the data. With the help of the developed image analysis workflow, it is thus possible to structure a lot of information from different imaging techniques data and extract features.

The last optionally applied neighborhood analysis made it possible to capture the communication between objects and the relation to their spatial arrangement within large measurement data. Another reason why spatial imaging is necessary is that it allows objects to be observed within their "natural" environment. Especially since neighborhood analysis can also be applied to live cell imaging. Based on located and identified objects, their spatial relation to the environment.

4.6. Prospects for future multiplexing projects

Multiplexing as part of microscopy, or as part of spatial localized gene expression detection, is a continuously evolving field in biology, medicine or bioimage data analysis. Thereby, the associated development of hardware and software solutions is an ongoing process as well. With improving resolutions, increasing number of parameters and advanced algorithms, the insights to functional biological processes may become visible at higher degree of detail in future. This is why I will introduce some thoughts and ideas for future projects in the next paragraphs.

One example refers to the MELC system. Typically a measurement requires roughly one hour for each cycle, because of incubation time, image acquisition, bleaching time and pipetting. This time increases with the raising number of FOV. If the sample could imaged and bleached along the entire sample in one cycle, the information content and related time could be improved. Therefore the excitation light has to be applied at all positions of the sample equally. Unfortunately, the required amount of light is limited by the system and an increase of power could also result in phototoxic or destructive processes in the tissue sample. An alternating laser, which randomly bleaches certain regions could be a solution. Additional to this, spectral unmixing, as utilized in Rakhymzhan et al., 2021, could also be used, to reduce the bleaching time and decompose the different signals by that. Furthermore, the applied image registration could be extended and applied right at the beginning of the MELC run, which would allow to

relocate and reuse an already used cover slide, because of the tissue preservation after each MELC run.

As another example, object identification by use of convolutional neuronal networks could replace semi-automated image segmentation workflows via ilastik and CellProfiler, based on pre-trained models, as shown in Du et al., 2021, where nuclei were found and used to quantify human cells from skin and confirm FACS experiments, similar to 3.4.1. Due to the fact, that the segmentation results were best for nuclei only, the ImageJ/Fiji Plugin “StarDist” (Schmidt et al., 2018) was not used in the presented projects. Newer pre-trained models, such as “Cellpose” (Stringer et al., 2021) also allow the detection of other structural objects and could improve the image segmentation in later projects. However, the use of these models require besides common objects, like nuclei, the re-training with annotated image data from the utilized microscopes. On the other hand, pixel based classification of all available parameters of a location would avoid any segmentation approaches and could highlight differences at sub cellular resolution.

Finally, an idea for a future project that requires all the tools developed and used in this work could be the investigation, characterization and classification of epithelial or endothelial cells around vessels in already available data sets from infected human lungs in terms of disease progression and tissue remodeling caused by SARS-CoV-2 infection. In contrast to the nicely oriented and aligned cell distribution of such cells in the control data sets, a disruption associated with dysfunction in relation to lung fibrosis was visually seen by using LSFM, MELC and ST (Mothes et al., 2023) at later disease progressions. Combining image segmentation, morphology classification via DFT shape descriptors and neighborhood analysis could further validate the results and would allow a closer look into the cellular communication processes. In addition, the extension of the existing two-dimensional algorithms to three-dimensional applications with regard to LSM, LSFM and MELC could increase the information content obtained.

5. Summary

Title: Bioimage analysis linking information at protein and transcriptional level in tissues

The optical resolution by human eyes is limited. This is why microscopes were invented, to enable the observation of smaller objects through a specific arrangement of optical components, as well as the investigation of biological processes regarding disease or treatment. Through the microscope's eye, the camera, images of various tissue stainings can be digitalized and stored on computers. At this time, image interpretation is often performed by the investigator. Manual annotation or counting is a subjective process by the observer, time consuming and therefore perhaps biased. Additional physical limitations increase the chance of misinterpretation regarding aberrations or other distortion artifacts, as well as complexity, which increases by the growing number of available parameters from the same location, notably in spatial gene expression detection. Bioimage analysis, as a newly emerging field in recent years, combines data analysis and imaging methods related to the study of biological processes and has led me to this work, where image data from various microscopy techniques were used to create supervised- and unsupervised bioimage analysis workflows validating biological questions in tissue samples from mouse and human, which were conceptually developed on the MELC system. Thereby, the MELC-system was improved to its overall image quality regarding resolution (laterally 60%, axial 250%), duration of MELC-experiments (reduction of heat and related drying of tissue sample, increase of detectable fluorescence signals) and signal evaluation (artifact removal). The application of system related artifact removal algorithms, such as image registration, illumination correction, image projections or destriping were part of required image pre-processing, which standardized subsequent image segmentation in ilastik and CellProfiler. Classification of found objects or cells benefited from clustering or dimensionality reduction methods simplifying complexity of multi parameter data sets. The application of neighborhood analysis supported the investigation with regard to spatial localizations and interaction. Based on quantitative image analysis workflows developed, including image pre-processing, image segmentation, object identification, classification and representation of the results, a general path from image acquisition to meaningful data evaluation as tool for users could was realized in case of LSM, MELC and LSFM data. Furthermore, the application of MELC and ST linked the information obtained between the protein and transcriptional level. In this way several observations were made, which helped to answer questions in immunology and neuroscience. Segmented LSM images of microglia from aging mice revealed the morphological changes and loss of processes during the course of dementia, using DFT descriptors as a compact way to describe complex shapes. The distribution of the complex compartment of stromal markers in the bone

marrow of mice was heterogeneous, which can provide insights into their roles in hematopoiesis and immune cell development. Supervised and unsupervised image analysis workflows identified rare ILC populations in MELC images of mouse tonsils. In addition, their location around vessels and fibronectin fibers was detected by neighborhood analysis. Finally, SARS-CoV-2-induced tissue remodeling was observed in human lung samples from MELC and ST experiments.

6. Zusammenfassung

Titel: Biobildanalyse, die Informationen auf Protein- und Transkriptionsebene in Geweben verknüpft

Die optische Auflösung des menschlichen Auges ist begrenzt. Deshalb wurden Mikroskope erfunden, um die Beobachtung kleinerer Objekte durch eine bestimmte Anordnung optischer Komponenten sowie die Untersuchung biologischer Prozesse im Hinblick auf Krankheit oder Behandlung zu ermöglichen. Mit Hilfe des Auges des Mikroskops, der Kamera, können Bilder von verschiedenen Gewebefärbungen digitalisiert und auf Computern gespeichert werden. Zu diesem Zeitpunkt wird die Bildinterpretation häufig vom Untersucher selbst vorgenommen. Die manuelle Markierung oder Zählung ist ein subjektiver Prozess des Beobachters, zeitaufwändig und daher möglicherweise voreingenommen. Zusätzliche physikalische Begrenzungen erhöhen das Risiko von Fehlinterpretationen in Bezug auf Aberrationen oder andere Verzerrungsartefakte sowie die Komplexität, die durch die wachsende Anzahl verfügbarer Parameter am selben Ort zunimmt, insbesondere bei der Erkennung der räumlichen Genexpression. Die Biobild-Analyse, ein in den letzten Jahren neu aufkommender Bereich, kombiniert Datenanalyse und bildgebende Verfahren zur Untersuchung biologischer Prozesse und hat mich zu dieser Arbeit geführt, bei der Bilddaten aus verschiedenen Mikroskopietechniken verwendet wurden, um überwachte und unüberwachte Biobild-Analyse-Arbeitsabläufe zur Validierung biologischer Fragen in Gewebeproben von Maus und Mensch zu erstellen, die konzeptionell auf dem MELC-System entwickelt wurden. Dabei wurde das MELC-System hinsichtlich seiner Gesamtbildqualität in Bezug auf Auflösung (lateral 60%, axial 250%), Dauer der MELC-Experimente (Reduktion der Hitze und damit verbundener Trocknung der Gewebeprobe, Erhöhung der detektierbaren Fluoreszenzsignale) und Signalauswertung (Artefaktentfernung) verbessert. Die Anwendung systembezogener Algorithmen zur Beseitigung von Artefakten, wie Bildregistrierung, Beleuchtungskorrektur, Bildprojektionen oder Entstreifung waren Teil der erforderlichen Bildvorverarbeitung, die die anschließende Bildsegmentierung in ilastik und CellProfiler standardisierte. Die Klassifikation der gefundenen Objekte oder Zellen profitierte von Clustering- oder Dimensionalitätsreduktionsverfahren, die die Komplexität der mehrparametrigen Datensätze vereinfachten. Die Anwendung von Nachbarschaftsanalysen unterstützte die Untersuchung im Hinblick auf räumliche Lokalisierungen und Interaktionen. Basierend auf den entwickelten quantitativen Bildanalyse-Workflows, die Bildvorverarbeitung, Bildsegmentierung, Objektidentifikation, Klassifikation und Ergebnisdarstellung umfassen, konnte im Falle von LSM-, MELC- und LSFM-Daten ein allgemeiner Weg von der Bildaufnahme bis zur aussagekräftigen Datenauswertung als Werkzeug für die Nutzer realisiert werden. Darüber hinaus wurden durch die Anwendung von MELC und ST die gewonnenen Informationen zwischen der Protein- und

der Transkriptionsebene verknüpft. Auf diese Weise konnten mehrere Beobachtungen gemacht werden, die zur Beantwortung von Fragen in der Immunologie und den Neurowissenschaften beitragen. Segmentierte LSM-Bilder von Mikroglia aus alternden Mäusen zeigten die morphologischen Veränderungen und den Verlust von Prozessen im Verlauf der Demenz, wobei DFT-Deskriptoren als kompakte Methode zur Beschreibung komplexer Formen verwendet wurden. Die Verteilung des komplexen Kompartiments stromaler Marker im Knochenmark von Mäusen war heterogen, was Einblicke in ihre Rolle bei der Hämatopoese und der Entwicklung von Immunzellen geben kann. Durch überwachte und nicht überwachte Bildanalyse-Arbeitsabläufe wurden seltene ILC-Populationen in MELC-Bildern von Mäusemandeln identifiziert. Darüber hinaus wurde ihre Lage in der Nähe von Gefäßen und Fibronektinfasern durch Nachbarschaftsanalyse ermittelt. Schließlich wurde in menschlichen Lungenproben aus MELC- und ST-Experimenten ein SARS-CoV-2-induzierter Gewebeumbau beobachtet.

7. References

Abbe, E. (1873). Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung: II. Die dioptrischen Bedingungen der Leistung des Mikroskops. *Archiv Für Mikroskopische Anatomie*, *9*(1), 418–440. https://doi.org/10.1007/BF02956174

Alberts, B., Johnson, A., Lewis, J., Morgan, D., Raff, M., Roberts, K., & Walter, P. (2014). *Molecular Biology of the Cell*. Garland Science.

Andrey, P., & Boudier, T. (2007). *Adaptive active contours (snakes) for the segmentation of complex structures in biological images*. https://www.researchgate.net/publication/252008688_Adaptive_active_contours_snakes_for_the_segmentation_of_complex_structures_in_biological_images#fullTextFileContent (last opend May, 25th 2023)

Anuta, P. E. (1970). Spatial Registration of Multispectral and Multitemporal Digital Imagery Using Fast Fourier Transform Techniques. *IEEE Transactions on Geoscience Electronics*, *8*(4), 353–368. https://doi.org/10.1109/TGE.1970.271435

Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A., & Seung, H. S. (2017). Trainable Weka Segmentation: A machine learning tool for microscopy pixel classification. *Bioinformatics*, *33*(15), 2424–2426. https://doi.org/10.1093/bioinformatics/btx180

Barnes, R., Lehman, C., & Mulla, D. (2014). Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers and Geosciences*, *62*, 117–127. https://doi.org/10.1016/j.cageo.2013.04.024

Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., Eren, K., Cervantes, J. I., Xu, B., Beuttenmueller, F., Wolny, A., Zhang, C., Koethe, U., Hamprecht, F. A., & Kreshuk, A. (2019). Ilastik: Interactive Machine Learning for (Bio)Image Analysis. *Nature Methods*, *16*(12), 1226–1232. https://doi.org/10.1038/s41592-019-0582-9

Beucher, S., & Lantuejoul, C. (1979). Use of Watersheds in Contour Detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation* (pp. 12–21). http://www.citeulike.org/group/7252/article/4083187

Boas, D. A., Pitris, C., & Ramanujam, N. (2016). *Handbook of Biomedical Optics*. CRC Press. https://books.google.de/books?id=IEUECp8OHWEC

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Burger, W., & Burge, M. J. (2016). *Digital Image Processing (second Edition)*. Springer Verlag.

Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D. A., Chang, J. H., Lindquist, R. A., Moffat, J., Golland, P., & Sabatini, D. M. (2006). CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, *7*(10). https://doi.org/10.1186/gb-2006-7-10-r100

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Ensemble Machine Learning. *Ensemble Machine Learning*, *January*. https://doi.org/10.1007/978-1-4419-9326-7

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., & Zupan, B. (2013). Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research*, *14*, 2349–2353. http://jmlr.org/papers/v14/demsar13a.html

Eng, J., Bucher, E., Hu, Z., Zheng, T., Gibbs, S. L., Chin, K., & Gray, J. W. (2022). A framework for multiplex imaging optimization and reproducible analysis. *Communications Biology*, *5*(1), 1–11. https://doi.org/10.1038/s42003-022-03368-y

Ertel, W. (2011). *Introduction to Artificial Intelligence Series editor (second Edition). Springer*.

F.R.S., K. P. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Fahrmeir, L., Heumann, C., Künstler, R., Pigeot, I., & Tutz, G. (2016). Statistik | Der Weg zur Datenanalyse. In *Springer Spektrum (Springer Verlag)*.

Feigenbaum, E. A., & Simon, H. A. (1961). *Performance of a Reading Task by an Elementary Perceiving and Memorizing Program*. RAND Corporation.

Fekri-Ershad, S. (2021). Cell phenotype classification using multi threshold uniform local ternary patterns in fluorescence microscope images. *Multimedia Tools and Applications*, *80*(8), 12103–12116. https://doi.org/10.1007/s11042-020-10321-w

Gerner, M. Y., Kastenmuller, W., Ifrim, I., Kabat, J., & Germain, R. N. (2012). Histo-Cytometry: A Method for Highly Multiplex Quantitative Tissue Imaging Analysis Applied to Dendritic Cell Subset Microanatomy in Lymph Nodes. *Immunity*, *37*(2), 364–376. https://doi.org/10.1016/j.immuni.2012.07.011

Gerstner, A. O. H., Trumpfheller, C., Racz, P., Osmancik, P., Tenner-Racz, K., & Tárnok, A. (2004). Quantitative histology by multicolor slide-based cytometry. *Cytometry Part A*, *59A*(2), 210–219. https://doi.org/10.1002/cyto.a.20054

Giesen, C., Wang, H. A. O., Schapiro, D., Zivanovic, N., Jacobs, A., Hattendorf, B., Schüffler, P. J., Grolimund, D., Buhmann, J. M., Brandt, S., Varga, Z., Wild, P. J., Günther, D., & Bodenmiller, B. (2014). Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry. *Nature Methods*, *11*(4), 417–422. https://doi.org/10.1038/nmeth.2869

Gonzalez, R. C., & Woods, R. E. (2002). *Digital Image Processing*. Prentice Hall.

Goodman, J. W. (2005). *Introduction to Fourier Optics*. THE McGRAW-HILL COMPANIES, INC.

Guizar-Sicairos, M., Thurman, S. T., & Fienup, J. R. (2008). Efficient subpixel image registration algorithms. *Optics Letters*, *33*(2), 156. https://doi.org/10.1364/OL.33.000156

Gut, G., Herrmann, M. D., & Pelkmans, L. (2018). Multiplexed protein maps link subcellular organization to cellular states. *Science*, *361*(6401). https://doi.org/10.1126/science.aar7042

Hafemeister, C., & Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biology*, *20*(1), 1–15. https://doi.org/10.1186/s13059-019-1874-1

HANSER, B. M., GUSTAFSSON, M. G. L., AGARD, D. A., & SEDAT, J. W. (2004). Phase-retrieved pupil functions in wide-field fluorescence microscopy. *Journal of Microscopy*, *216*(1), 32–48. https://doi.org/10.1111/j.0022-2720.2004.01393.x

Hao, Y., Hao, S., Andersen-Nissen, E., III, W. M. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zagar, M., Hoffman, P., Stoeckius, M., Papalexi, E., Mimitou, E. P., Jain, J., Srivastava, A., Stuart, T., Fleming, L. B., Yeung, B., … Satija, R. (2021). Integrated analysis of multimodal single-cell data. *Cell*. https://doi.org/10.1016/j.cell.2021.04.048

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. In *Revista Espanola de las Enfermedades del Aparato Digestivo* (Vol. 26, Issue 4). Springer New York. https://doi.org/10.1007/978-0-387-84858-7

Herman, G. T. (1980). *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press.

Heumann, C., Schomaker, M., & Shalabh. (2016). Introduction to Statistics and Data Analysis. In *Journal of Association of Physicians of India* (Vol. 64, Issue July). Springer International Publishing. https://doi.org/10.1007/978-3-319-46162-5

Hinton, G., & Roweis, S. (2003). Stochastic neighbor embedding. *Advances in Neural Information Processing Systems*.

Holzwarth, K., Köhler, R., Philipsen, L., Tokoyoda, K., Ladyhina, V., Wählby, C., Niesner, R. A., & Hauser, A. E. (2018). Multiplexed fluorescence microscopy reveals heterogeneity among stromal cells in mouse bone marrow sections. *Cytometry Part A*, *93*(9), 876–888. https://doi.org/10.1002/cyto.a.23526

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417.

Im, K., Mareninov, S., Diaz, M. F. P., & Yong, W. H. (2019). An Introduction to Performing Immunofluorescence Staining. *Methods in Molecular Biology (Clifton, N.J.)*, *1897*(1), 299–311. https://doi.org/10.1007/978-1-4939-8935-5_26

Kamentsky, L., Jones, T. R., Fraser, A., Bray, M. A., Logan, D. J., Madden, K. L., Ljosa, V., Rueden, C., Eliceiri, K. W., & Carpenter, A. E. (2011). Improved structure, function and compatibility for cellprofiler: Modular high-throughput image analysis software. *Bioinformatics*, *27*(8), 1179–1180. https://doi.org/10.1093/bioinformatics/btr095

Kang, W., Yu, S., Seo, D., Jeong, J., & Paik, J. (2015). Push-broom-type very high-resolution satellite sensor data correction using combined wavelet-fourier and multiscale non-local means filtering. *Sensors (Switzerland)*, *15*(9), 22826–22853. https://doi.org/10.3390/s150922826

Kriegel, F. L., Köhler, R., Bayat-Sarmadi, J., Bayerl, S., Hauser, A. E., Niesner, R., Luch, A., & Cseresnyes, Z. (2018a). Cell shape characterization and classification with discrete Fourier transforms and self-organizing maps. *Cytometry Part A*, *93*(3), 323–333. https://doi.org/10.1002/cyto.a.23279

Kriegel, F. L., Köhler, R., Bayat-Sarmadi, J., Bayerl, S., Hauser, A. E., Niesner, R., Luch, A., & Cseresnyes, Z. (2018b). Morphology-Based Distinction Between Healthy and Pathological Cells Utilizing Fourier Transforms and Self-Organizing Maps. *Journal of Visualized Experiments : JoVE*, *140*. https://doi.org/10.3791/58543

Kubat, M. (2017). An Introduction to Machine Learning. *Springer International Publishing*. https://doi.org/10.1007/978-3-319-63913-0

Lakowicz, J. R. (2013). *Principles of Fluorescence Spectroscopy*. Springer US.

Li, C. ., Yang, H. ., Cai, Y. ., & Song, B. (2015). Image Denoising Algorithm Based on Non-Subsampled Contourlet Transform and Bilateral Filtering. *Proceedings of the International Conference on Computer Information Systems and Industrial Applications*, *18*(Cisia), 666–669. https://doi.org/10.2991/cisia-15.2015.182

Liang, X., Zang, Y., Dong, D., Zhang, L., Fang, M., Yang, X., Arranz, A., Ripoll, J., Hui, H., & Tian, J. (2016). Stripe artifact elimination based on nonsubsampled contourlet transform for light sheet fluorescence microscopy. *Journal of Biomedical Optics*, *21*(10), 106005. https://doi.org/10.1117/1.jbo.21.10.106005

Lichtman, J. W., & Conchello, J. A. (2005). Fluorescence microscopy. *Nature Methods*, *2*(12), 910–919. https://doi.org/10.1038/nmeth817

Lin, J. R., Fallahi-Sichani, M., & Sorger, P. K. (2015). Highly multiplexed imaging of single cells using a high-throughput cyclic immunofluorescence method. *Nature Communications*, *6*, 1–7. https://doi.org/10.1038/ncomms9390

MacQueen, J. B. (1967). Some Methods for Classification and Analysis of MultiVariate Observations. In L. M. Le Cam & J. Neyman (Eds.), *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297). University of California Press.

Masters, B. R. (2008). History of the optical microscope in cell biology and medicine. *ELS*.

McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. http://arxiv.org/abs/1802.03426 (last opened May, 25th 2023)

McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., Doan, M., Ding, L., Rafelski, S. M., Thirstrup, D., Wiegraebe, W., Singh, S., Becker, T., Caicedo, J. C., & Carpenter, A. E. (2018). CellProfiler 3.0: Next-generation image processing for biology. *PLOS Biology*, *16*(7), e2005970. https://doi.org/10.1371/journal.pbio.2005970

Miura, K., & Sladoje, N. (2019). *Bioimage Data Analysis Workflows*. Springer International Publishing.

Moses, L., & Pachter, L. (2022). Museum of spatial transcriptomics. *Nature Methods*, *19*(5), 534–546. https://doi.org/10.1038/s41592-022-01409-2

Mothes, R., Pascual-Reguant, A., Koehler, R., Liebeskind, J., Liebheit, A., Bauherr, S., Dittmayer, C., Laue, M., Manitius, R. von, Elezkurtaj, S., Durek, P., Heinrich, F., Heinz, G. A., Guerra, G. M., Obermayer, B., Meinhardt, J., Ihlow, J., Radke, J., Heppner, F. L., … Hauser, A. E. (2022). Local CCL18 and CCL21 expand lung fibrovascular niches and recruit lymphocytes, leading to tertiary lymphoid structure formation in prolonged COVID-19. *MedRxiv*, 2022.03.24.22272768. https://doi.org/10.1101/2022.03.24.22272768

Mothes, R., Pascual-Reguant, A., Koehler, R., Liebeskind, J., Liebheit, A., Bauherr, S., Philipsen, L., Dittmayer, C., Laue, M., von Manitius, R., Elezkurtaj, S., Durek, P., Heinrich, F., Heinz, G. A., Guerra, G. M., Obermayer, B., Meinhardt, J., Ihlow, J., Radke, J., … Hauser, A. E. (2023). Distinct tissue niches direct lung immunopathology via CCL18 and CCL21 in severe COVID-19. *Nature Communications*, *14*(1), 791. https://doi.org/10.1038/s41467-023-36333-2

Münch, B., Trtik, P., Marone, F., & Stampanoni, M. (2009). Stripe and ring artifact removal with combined wavelet-Fourier filtering. *EMPA Activities*, *17*(2009-2010 EMPA ACTIVITIES), 34–35. https://doi.org/10.1364/oe.17.008567

Narasimha Murty, M., & Krishna, G. (1980). A computationally efficient technique for data-clustering. *Pattern Recognition*, *12*(3), 153–158. https://doi.org/10.1016/0031-3203(80)90039-4

Olsen, T. K., & Baryawno, N. (2018). Introduction to Single-Cell RNA Sequencing. *Current Protocols in Molecular Biology*, *122*(1). https://doi.org/10.1002/cpmb.57

Otsu, N., Smith, P. L., Reid, D. B., Environment, C., Palo, L., Alto, P., & Smith, P. L. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *C*(1), 62–66.

Pascual-Reguant, A., Köhler, R., Mothes, R., Bauherr, S., Hernández, D. C., Uecker, R., Holzwarth, K., Kotsch, K., Seidl, M., Philipsen, L., Müller, W., Romagnani, C., Niesner, R., & Hauser, A. E. (2021). Multiplexed histology analyses for the phenotypic and spatial characterization of human innate lymphoid cells. *Nature Communications*, *12*(1), 1–15. https://doi.org/10.1038/s41467-021-21994-8

Pertuz, S., Puig, D., Garcia, M. A., & Fusiello, A. (2013). Generation of All-in-Focus Images by Noise-Robust Selective Fusion of Limited Depth-of-Field Images. *IEEE Transactions on Image Processing*, *22*(3), 1242–1251. https://doi.org/10.1109/TIP.2012.2231087

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/bf00116251

Radbruch, H., Mothes, R., Bremer, D., Seifert, S., Köhler, R., Pohlan, J., Ostendorf, L., Günther, R., Leben, R., Stenzel, W., Niesner, R. A., & Hauser, A. E. (2017). Analyzing nicotinamide adenine dinucleotide phosphate oxidase activation in aging and vascular amyloid pathology. *Frontiers in Immunology*, *8*(JUL), 1–12. https://doi.org/10.3389/fimmu.2017.00844

Rakhymzhan, A., Acs, A., Hauser, A. E., Winkler, T. H., & Niesner, R. A. (2021). Improvement of the similarity spectral unmixing approach for multiplexed two-photon imaging by linear dimension reduction of the mixing matrix. *International Journal of Molecular Sciences*, *22*(11). https://doi.org/10.3390/ijms22116046

Reinartz, J., Bruyns, E., Lin, J. Z., Burcham, T., Brenner, S., Bowen, B., Kramer, M., & Woychik, R. (2002). Massively parallel signature sequencing (MPSS) as a tool for in-depth quantitative gene expression profiling in all organisms. *Briefings in Functional Genomics and Proteomics*, *1*(1), 95–104. https://doi.org/10.1093/bfgp/1.1.95

Rueden, C. T., Schindelin, J., Hiner, M. C., DeZonia, B. E., Walter, A. E., Arena, E. T., & Eliceiri, K. W. (2017). ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics*, *18*(1), 1–26. https://doi.org/10.1186/s12859-017-1934-z

S., L. R. S. R. (1896). XV. On the theory of optical images, with special reference to the microscope. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *42*(255), 167–195. https://doi.org/10.1080/14786449608620902

Sanger, F., Nicklen, S., & Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, *74*(12), 5463–5467. https://doi.org/10.1073/pnas.74.12.5463

Schapiro, D., Jackson, H. W., Raghuraman, S., Fischer, J. R., Vito, R., Zanotelli, T., Schulz, D., Giesen, C., Catena, R., & Varga, Z. (2018). *Europe PMC Funders Group miCAT : A toolbox for analysis of cell phenotypes and interactions in multiplex image cytometry data*. *14*(9), 873–876. https://doi.org/10.1038/nmeth.4391.miCAT

Schapiro, D., Jackson, H. W., Raghuraman, S., Fischer, J. R., Zanotelli, V. R. T., Schulz, D., Giesen, C., Catena, R., Varga, Z., & Bodenmiller, B. (2017). HistoCAT: Analysis of cell phenotypes and interactions in multiplex image cytometry data. *Nature Methods*, *14*(9), 873–876. https://doi.org/10.1038/nmeth.4391

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P., & Cardona, A. (2012). Fiji: an open-source platform for biological-image analysis. *Nature Methods*, *9*(7), 676–682. https://doi.org/10.1038/nmeth.2019

Schmidt, U., Weigert, M., Broaddus, C., & Myers, G. (2018). Cell detection with star-convex polygons. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11071 LNCS*, 265–273. https://doi.org/10.1007/978-3-030-00934-2_30

Schubert, W., Bonnekoh, B., Pommer, A. J., Philipsen, L., Böckelmann, R., Malykh, Y., Gollnick, H., Friedenberger, M., Bode, M., & Dress, A. W. M. (2006). Analyzing proteome topology and function by automated multidimensional fluorescence microscopy. *Nature Biotechnology*, *24*(10), 1270–1278. https://doi.org/10.1038/nbt1250

Schubert, W., Gieseler, A., Krusche, A., Serocka, P., & Hillert, R. (2012). Next-generation biomarkers based on 100-parameter functional super-resolution microscopy TIS. *New Biotechnology*, *29*(5), 599–610. https://doi.org/https://doi.org/10.1016/j.nbt.2011.12.004

Schwartz, J., Jiang, Y., Wang, Y., Aiello, A., Bhattacharya, P., Yuan, H., Mi, Z., Bassim, N., & Hovden, R. (2019). Removing Stripes, Scratches, and Curtaining with Non-Recoverable Compressed Sensing. *Microscopy and Microanalysis*, *25*(S2), 174–175. https://doi.org/10.1017/s1431927619001600

Shih, F. Y., & Cheng, S. (2005). Automatic seeded region growing for color image segmentation. *Image and Vision Computing*, *23*(10), 877–886. https://doi.org/https://doi.org/10.1016/j.imavis.2005.05.015

Ståhl, P. L., Salmén, F., Vickovic, S., Lundmark, A., Navarro, J. F., Magnusson, J., Giacomello, S., Asp, M., Westholm, J. O., Huss, M., Mollbrink, A., Linnarsson, S., Codeluppi, S., Borg, Å., Pontén, F., Costea, P. I., Sahlén, P., Mulder, J., Bergmann, O., … Frisén, J. (2016). Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. *Science*, *353*(6294), 78–82. https://doi.org/10.1126/science.aaf2403

Stirling, D. R., Swain-Bowden, M. J., Lucas, A. M., Carpenter, A. E., Cimini, B. A., & Goodman, A. (2021). CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics*, *22*(1), 1–11. https://doi.org/10.1186/s12859-021-04344-9

Stringer, C., Wang, T., Michaelos, M., & Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, *18*(1), 100–106. https://doi.org/10.1038/s41592-020-01018-x

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., Lao, K., & Surani, M. A. (2009). mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, *6*(5), 377–382. https://doi.org/10.1038/nmeth.1315

Umbaugh, S. E. (2017). *Digital Image Processing and Analysis: Applications with MATLAB and CVIPtools*. CRC Press. https://books.google.de/books?id=ZflADwAAQBAJ

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Wählby, C., Erlandsson, F., Bengtsson, E., & Zetterberg, A. (2002). Sequential immunofluorescence staining and image analysis for detection of large numbers of antigens in individual cell nuclei. *Cytometry*, *47*(1), 32–41.

Zehentmeier, S., Roth, K., Cseresnyes, Z., Sercan, Ö., Horn, K., Niesner, R. A., Chang, H. D., Radbruch, A., & Hauser, A. E. (2014). Static and dynamic components synergize to form a stable survival niche for bone marrow plasma cells. *European Journal of Immunology*, *44*(8), 2306–2317. https://doi.org/10.1002/eji.201344313

Zernike, F. (1942). Phase contrast, a new method for the microscopic observation of transparent objects. *Physica*, *9*(7), 686–698. https://doi.org/10.1016/S0031-8914(42)80035-X

Zernike, F. (1955). How I Discovered Phase Contrast. *Science*, *121*(3141), 345–349. https://doi.org/10.1126/science.121.3141.345

8. Appendix – Source codes

8.1. Source codes

8.1.1. SHADE – Fiji/imageJ PlugIn

```
/*
 * SHADE  (Shape Data Evaluation) plugin for ImageJ/Fiji
 *
 * Copyright (C) 2017 Ralf Köhler(1), Fabian Kriegel(2) and Dr. Zoltán Cseresnyés
 *
 * (1) German Rheumatism Research Centre Berlin, Immune Dynamics
 * (2) German Federal Institute for Risk Assessment, Department of Chemical and
 * Product Safety
 * (3) Hans Knöll Institute Jena, Applied Systems Biology
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation
 * (http://www.gnu.org/licenses/gpl.txt )
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public
 * License along with this program.  If not, see http://gnu.org/licenses/gpl.html
 *
 * */

import java.awt.Polygon;
import java.io.File;

import ij.IJ;
import ij.ImagePlus;
import ij.Prefs;
import ij.gui.GenericDialog;
import ij.gui.PolygonRoi;
import ij.gui.Roi;
import ij.gui.WaitForUserDialog;
import ij.measure.Measurements;
import ij.measure.ResultsTable;
import ij.plugin.PlugIn;
import ij.plugin.filter.Analyzer;
import ij.plugin.frame.RoiManager;
import ij.process.ImageConverter;
import ij.process.ImageProcessor;

/*
 * This Plugin was developed to characterize the shape in 2D projections of 3D
 * cell images recorded with a two photon microscope in discrete fourier
 * transformed components. In just a few components (we used 20) the whole shape
 * can be saved and reproduced within a small memory usage. Special shape
 * properties are visible insight these values. With all components a database
 * can be build creating machine learning approaches.
 *
 * For more information you can read in following publication:
 * Cell shape characterization and classification with discrete
 * Fourier transforms and Self-Organizing Maps - Fabian L. Kriegel et al.;
```

```
 * (Cytometry, Part A)
 *
 * */

public class SHADE_  implements PlugIn {

        //initialize the global variables
        boolean checkbatch, checkintermediate, checkSave, checkDarkBackground;
        double numbergradient, iterationnumber, dilatations;
        int checkintermediate_wert;
        String dir2, imageName;
        ResultsTable bigResultsTable = new ResultsTable();

    public void run(String arg) {

        //graphical user interface for user defined values
        GenericDialog gd = new GenericDialog("SHADE");
         // open Dialogwindow with Title "SHADE"

        gd.addMessage("Set up the SHADE for finding edges");
        gd.addMessage("----------------------------------------");
        //gd.addCheckbox("Step-by-Step-Mode (manualy saving) ?",false);
        gd.addNumericField("Enter gradient threshold  ",40,0);
        gd.addNumericField("Enter number of iterations",25,0);
        gd.addNumericField("Number of dilatations", 10, 0);
        gd.addCheckbox("Dark Background?", false);
        gd.addCheckbox("Do you want to see the intermediate results ?", true);
        gd.addCheckbox("Do you want to save Result Tables?", true);
        gd.showDialog();

        if(gd.wasCanceled()) return;

        numbergradient = gd.getNextNumber();
        iterationnumber = gd.getNextNumber();
        dilatations = gd.getNextNumber();
        checkDarkBackground = gd.getNextBoolean();
        checkintermediate = gd.getNextBoolean();
        checkSave = gd.getNextBoolean();

        if(checkintermediate==true)
        checkintermediate_wert=1;
        else checkintermediate_wert=0;

        // get directory of single cell images (tif file format required)
        String dir1 = IJ.getDirectory("Choose Source Folder");
        if(dir1 == null) return;
        String[] list1 = new File(dir1).list();
        if(list1 == null) return;
        IJ.log("image source folder: "+dir1);

        // get saving directory if option is checked
        if(checkSave==true){
            dir2 = IJ.getDirectory("Choose Results saving folder");
            IJ.log("Results Table saving folder: "+dir2);
        }

        int[] roiLoc = new int[4];

        // loop through every image and process
```

```
        for (int l = 0; l < list1.length; l++) {

            if(list1[l].endsWith(".tif")){

                IJ.open(dir1+list1[l]);

                ImagePlus img = IJ.getImage();
                imageName = img.getShortTitle();

                // first image is used to get user defined ROI
                if(!ROIisSet(roiLoc)) roiLoc = getInputROI(img);
                img.setRoi(roiLoc[0],roiLoc[1],roiLoc[2],roiLoc[3]);

                //run absnake plugin with preprocessed images
                startPreprocess(img);
                ResultsTable rt = new ResultsTable();
                rt = getABSnakeCoordsAndCalc(img.getTitle());

                if(checkintermediate==true)
                rt.show("Results of : " + list1[l]);

                if(checkSave==true)
                rt.save(dir2+"Results_of_DFT_calc_"+(l+1)+"_"+imageName+".csv");
                IJ.run("Close All", "");
            }
        }
        bigResultsTable.show("All DFT components");
        if(checkSave==true) bigResultsTable.save(dir2
             +"Result_collection_of_all_DFT_calculations.csv");
        IJ.error("DFT calculations are done");
    }

    // check if ROI is set in a first image event
    public boolean ROIisSet(int[] roiLoc){

        if(roiLoc[0]>0 && roiLoc[1]>0 && roiLoc[2]>0 && roiLoc[3]>0){
            return true;
        }
        return false;
    }

    // create ROI only around cell -> uninteresting parts will be cropped
    public int[] getInputROI(ImagePlus img) {

        ResultsTable rt = new ResultsTable();
        int measurements = Measurements.RECT;
        Analyzer analyzer = new Analyzer(img, measurements, rt);
        int[] roiPosition = new int[4];

        new WaitForUserDialog("First Image Event", "draw "
                + "rectangle surounding cell !\n"
                + "then you can go on by clicking 'OK' ").show();

        RoiManager rm = RoiManager.getInstance();
        if(rm==null) rm = new RoiManager();
        rm.addRoi(img.getRoi());
        analyzer.measure();
        roiPosition[0] = (int)rt.getValue("BX", 0);
        roiPosition[1] = (int)rt.getValue("BY", 0);
```

```
        roiPosition[2] = (int)rt.getValue("Width", 0);
        roiPosition[3] = (int)rt.getValue("Height", 0);
        img.deleteRoi();
        rm.select(0);
        rm.runCommand(img,"Deselect");
        rm.runCommand(img,"Delete");
         return roiPosition;
    }
// crop image, make selection and run absnake plugin adding results to ROI manager
    public void startPreprocess(ImagePlus img){

        ImagePlus inputImg = img.crop();
        img.close();
        ImageConverter cv = new ImageConverter(inputImg);
        cv.convertToGray8();

            if (!checkDarkBackground) {
                IJ.setAutoThreshold(inputImg, "RenyiEntropy");
            }
            else {
                IJ.setAutoThreshold(inputImg,  "RenyiEntropy dark");
            }

        Prefs.blackBackground = false;
        IJ.run(inputImg, "Convert to Mask", "");

        ImagePlus duplImg = inputImg.duplicate();
        ImageProcessor ipDupl = duplImg.getProcessor();

        for(int i = 0; i<dilatations; i++) {
            ipDupl.dilate();
        }

        inputImg.deleteRoi();
        duplImg.deleteRoi();
        IJ.run(duplImg, "Create Selection" ,"");
        Roi roi = duplImg.getRoi();

        Polygon p = roi.getPolygon();
        inputImg.setRoi(new PolygonRoi(p.xpoints,p.ypoints,p.npoints,Roi.POLYGON))
;

        duplImg.close();
        RoiManager rm = RoiManager.getInstance();
        if(rm==null) rm = new RoiManager();
        rm.addRoi(inputImg.getRoi());
        inputImg.show();

        IJ.run("ABSnake", "gradient_threshold="
             +numbergradient+" number_of_iterations="
        +iterationnumber+" step_results_show ="
             +checkintermediate_wert+" draw_color=Red save_coords");

        rm.select(0);
        rm.runCommand(img,"Deselect");
        rm.runCommand(img,"Delete");

        inputImg.close();
```

```
    }

// read results from absnake txt file, calculate DFT components
// and add to Results Table
    public ResultsTable getABSnakeCoordsAndCalc(String imageTitle) {
        String pathfile = "ABSnake-r1-z1.txt";
        String filestring = IJ.openAsString(pathfile);
        String[] rows = filestring.split("\n");
        //String[] title_row = rows[0].split("\\s+");

        float[] num = new float[rows.length];
        float[] x = new float[rows.length];
        float[] y = new float[rows.length];
        float[] z = new float[rows.length];
        float[] xcal = new float[rows.length];
        float[] ycal = new float[rows.length];

            for (int r=1; r<rows.length; r++)
            {
            String[] data = rows[r].split("\\s+");
            String number = data[0];
            num[r] = Float.parseFloat(number);
            String xvalue = data[1];
            x[r] = Float.parseFloat(xvalue);
            String yvalue = data[2];
            y[r] = Float.parseFloat(yvalue);
            String zvalue = data[3];
            z[r] = Float.parseFloat(zvalue);
            String xcali = data[4];
            xcal[r] = Float.parseFloat(xcali);
            String ycali = data[5];
            ycal[r] = Float.parseFloat(ycali);

            }

        int n = x.length;
        double[] oreal = new double[n];
        double[] oimag = new double[n];
        double[] oAmpl = new double[n];
        String[] natemp = new String[20];
        double[] toreal = new double[20];
        double[] toimag = new double[20];
        double[] toAmpl = new double[20];
        double[] tnumber = new double[20];
        String name = "Fourier-Parameter";

            for (int k = 0; k < n; k++)
            {
            double creal = 0;
            double cimag = 0;

            for (int t = 0; t < n; t++) {
                creal = creal
+ x[t]*Math.cos(2*Math.PI * t * k / n) + y[t]*Math.sin(2*Math.PI * t * k / n);
                cimag = cimag
+ -x[t]*Math.sin(2*Math.PI * t * k / n) + y[t]*Math.cos(2*Math.PI * t * k / n);
                }

            oreal[k] = creal;
```

```
            oimag[k] = cimag;
            oAmpl[k] = Math.sqrt(creal*creal + cimag*cimag);
            }

        double[] timepoint = new double [19]; double time = 0;

            for (int ti=0; ti<19; ti++)
            {   timepoint[ti] = time ; time = time+1;   }

        ResultsTable frt = new ResultsTable();

            for (int temc=1; temc < 20; temc++){

            toreal[temc]=oreal[temc];
            toimag[temc]=oimag[temc];
            toAmpl[temc]=oAmpl[temc];
            natemp[temc]=name;
            tnumber[temc]=temc+1;

            String real_value = String.valueOf(toreal[temc]);
            String imag_value = String.valueOf(toimag[temc]);
            String ampl_value = String.valueOf(toAmpl[temc]);

            bigResultsTable.incrementCounter();
            bigResultsTable.addValue("Real", real_value);
            bigResultsTable.addValue("Imag", imag_value);
            bigResultsTable.addValue("Ampl", ampl_value);
            bigResultsTable.addValue("file name", imageTitle);
            frt.incrementCounter();
            frt.addValue("Real", real_value); // (first column, value column(r))
            frt.addValue("Imag", imag_value);
            frt.addValue("Ampl", ampl_value);

            }

            bigResultsTable.incrementCounter();
            bigResultsTable.addValue("Real", 0);
            bigResultsTable.addValue("Imag", 0);
            bigResultsTable.addValue("Ampl", 0);
        //frt.show("Results");
        return frt;
    }

}
```

8.1.2. MELC Evaluation Toolbox – Fiji/imageJ PlugIn

MELC_EvaluationToolbox (main class)

```
import ij.IJ;
import ij.ImagePlus;
import ij.gui.GenericDialog;
import ij.plugin.PlugIn;

public class MELC_EvaluationToolbox_ implements PlugIn {

    ImagePlus img;

    public void run(String arg) {

        ImagePlus img = IJ.getImage();
        getUserInputAndDetect(img);

    }

    void getUserInputAndDetect(ImagePlus inputImage){

        String[] anySizeChoice = {"yes","no"};

        GenericDialog gd = new GenericDialog("give me some input");
        gd.addNumericField("sigma (unsharp mask): ", 3, 2);
        gd.addNumericField("weight factor: ", 0.75, 2);
        gd.addNumericField("mean factor: ", 1.25, 2);
        gd.addNumericField("min Diameter [px]:" , 10, 1);
        gd.addNumericField("max Diameter [px]:", 30, 1);
        //gd.addChoice("fill holes", holesChoice, "no");
        //gd.addChoice("use watershed", watershedChoice, "no");
        gd.addChoice("show any size", anySizeChoice, "no");
        gd.showDialog();

        if(gd.wasCanceled()) return;

        detector d = new detector(inputImage);
        d.setParams((int)gd.getNextNumber(), (float)gd.getNextNumber(),
                    gd.getNextNumber(), (int)gd.getNextNumber(),
                   (int)gd.getNextNumber(), gd.getNextChoice());
        d.detectNuclei();

        if(d.anySizeChoice == "no"){
            d.show();
        }
        else d.showAnySizeCells();

      }
    }
```

cell class

```
import java.util.ArrayList;
import java.awt.Point;
import ij.gui.PointRoi;

public class cell extends MELC_EvaluationToolbox_ {


    int minx, miny, maxx, maxy;
    int startX, startY;
```

```
int centerX, centerY;
int area;

ArrayList<Point> points;
ArrayList<Integer> intensities;
ArrayList<Point> borderPoints;
//PointRoi cellRoi;

cell(int x, int y){

    minx = x;
    miny = y;
    maxx = x;
    maxy = y;

    centerX = x;
    centerY = y;

    points = new ArrayList<Point>();
    points.add(new Point(x,y));

    intensities = new ArrayList<Integer>();

    borderPoints = new ArrayList<Point>();

    area = 1;

}

void add(int x, int y){

    Point p = new Point(x,y);
    points.add(p);

    minx = Math.min(minx, x);
    miny = Math.min(miny, y);
    maxx = Math.max(maxx, x);
    maxy = Math.max(maxy, y);

    centerX = (minx + maxx) / 2;
    centerY = (miny + maxy) / 2;

    area += 1;
}

void add(int x, int y, int intensity){

    Point p = new Point(x,y);
    points.add(p);

    minx = Math.min(minx, x);
    miny = Math.min(miny, y);
    maxx = Math.max(maxx, x);
    maxy = Math.max(maxy, y);

    centerX = (minx + maxx) / 2;
    centerY = (miny + maxy) / 2;

    intensities.add(intensity);
```

```
        area += 1;
     }
    int mean(){

        int m = 0;
        double meanIntensity = 0;

        for(int i : intensities){

            m += i;
        }

        meanIntensity = m/intensities.size();
        return (int)meanIntensity;
    }

    int area(){

        return area;
    }

    boolean contains(int x, int y){

        boolean contain = false;

        for (Point p : points){

            if(p.x == x && p.y == y){

                contain = true;
                break;
            }
        }
        return contain;
    }

    ArrayList<Point> getOutline(){

        Point pn, pe, ps, pw; // north, east, south, west

        for(Point p : points){

            pn = new Point(p.x,p.y+1);
            pe = new Point(p.x+1,p.y);
            ps = new Point(p.x,p.y-1);
            pw = new Point(p.x-1,p.y);

            if(!points.contains(pn) || !points.contains(pe) ||
                    !points.contains(ps) || !points.contains(pw)){

                Point pp = new Point(p.x, p.y);
                borderPoints.add(pp);

            }

        }
        return borderPoints;
    }
}
```

detector class

```
import java.awt.Color;
import java.awt.Point;
import java.util.ArrayList;
import java.util.Collections;

import ij.IJ;
import ij.ImagePlus;
import ij.Prefs;
import ij.gui.Overlay;
import ij.gui.PointRoi;
import ij.gui.Roi;
import ij.measure.Measurements;
import ij.measure.ResultsTable;
import ij.plugin.filter.Analyzer;
import ij.plugin.filter.Convolver;
import ij.plugin.filter.ParticleAnalyzer;
import ij.process.ImageProcessor;

public class detector extends MELC_EvaluationToolbox_{

    ImagePlus img, maskImage, voronoiImage, outlineImage;
    ImageProcessor ip, maskIP, voronoiIP, outlineImageP;
    int sigma, minArea, maxArea, startPointIndex, minDiameter, maxDiameter;
    float weight;
    double meanTollerance, distLimit, backgroundMin;
    boolean holes = false;
    boolean watershed = false;
    String anySizeChoice;
    ArrayList<cell> cells = new ArrayList<cell>();
    ArrayList<Point> startPoints = new ArrayList<Point>();
    cell c;


    detector(ImagePlus inputImage){

        img = inputImage;
        ip = img.getProcessor();
        sigma = 3;
        weight = 0.75f;
        meanTollerance = 1.5;
        maxArea = 0;

    }

    void setParams(int sigma, float weight,
      double meanTollerance, int minDiameter,
            int maxDiameter, String anySizeChoice){

        this.sigma = sigma;
        this.weight = weight;
        this.meanTollerance = meanTollerance;
        this.minDiameter = minDiameter;
        this.maxDiameter = maxDiameter;
        this.anySizeChoice = anySizeChoice;
```

```
        minArea = (int)((minDiameter * minDiameter)*Math.PI/4);
        maxArea = (int)((maxDiameter * maxDiameter)*Math.PI/4);

        calcDistLimit(maxDiameter);
        IJ.log("----------------------------------------");
        IJ.log("set Parameters : ");
        IJ.log("input image           : " + img.getTitle());
        IJ.log("sigma  (unsharp mask) : " + sigma);
        IJ.log("weight (unsharp mask) : " + weight);
        IJ.log("min diameter          : " + minDiameter);
        IJ.log("max diameter          : " + maxDiameter);
        IJ.log("min area              : " + minArea);
        IJ.log("max area              : " + maxArea);
        IJ.log("any size choice       : " + anySizeChoice);
        IJ.log("----------------------------------------");
    }

    void detectNuclei(){

        IJ.showStatus("looking for seeds...");
        getStartPositions(img);
        createMask();
        //analyseMask();
    }

    void detectMembrane(Roi[] nucleiRoi, ArrayList<Point> centers,
                        double[] minNeighborDist){

        // new version with additional Roi[] Array
        // comming from the allowed centers you should start at boundary
        // pixels of this roi and let it grow, to avoid center segmentation

        startPoints = centers;
        backgroundMin = 0;
        Overlay ovl = new Overlay();
        startPointIndex = 0;
        maskImage = IJ.createImage("Membrane Mask",
                    img.getWidth(), img.getHeight(), 1, 8);
        maskIP = maskImage.getProcessor();
        voronoiImage = IJ.createImage("Voronoi Mask",
                    img.getWidth(), img.getHeight(), 1, 8);
        voronoiIP = voronoiImage.getProcessor();
        //voronoi image has to be build by placing all the positiv starting points
        //inside empty image

        for(Point sPoint : startPoints){

            voronoiIP.set(sPoint.x, sPoint.y, 255);
        }

        IJ.run(voronoiImage, "Convert to Mask", "");
        IJ.run(voronoiImage, "Voronoi", "");

        for(Point p : startPoints){

            PointRoi pr = new PointRoi(p.x,p.y);
            pr.setPointType(2);
            ovl.add(pr);
            distLimit = minNeighborDist[startPointIndex];
```

```
            c = new cell(p.x,p.y);
            c.add(p.x,p.y,ip.get(p.x, p.y));
            grow4N(p.x, p.y);
            //cells.add(c);
            startPointIndex++;
        }
        getCellOutlines();
        maskImage.setOverlay(ovl);
        maskImage.show();
    }

    void createMask(){

        maskImage = IJ.createImage("Nuclei Mask", img.getWidth(),
                    img.getHeight(), 1, 8);
        maskIP = maskImage.getProcessor();
        startPointIndex = 0;
        IJ.showStatus("cell regions grow...");

        for(Point p : startPoints){

            IJ.showProgress(startPointIndex+1, startPoints.size());
            IJ.showStatus("cell regions grow...");
            c = new cell(p.x,p.y);
            c.add(p.x,p.y,ip.get(p.x, p.y));
            grow4N(p.x, p.y);
            cells.add(c);
            startPointIndex++;
        }

        if(holes){
            IJ.run(maskImage, "Fill Holes", "");
        }

        if(watershed){
            IJ.run(maskImage, "Watershed","");
        }
        //get the real cell objects after cleanup from maskImage
        getCellOutlines();

    }

    ImagePlus getMask(){

        return maskImage;
    }

    void getStartPositions(ImagePlus img){

        // basic idea of this function is to get the start pixel map for
        // a region growing algorithm. To extend image signal it is filtered
        // with unsharp mask filter. At next thresholding creates binary image
        // where we calculate the ultimate points (max eroded distance map).
        // every pixel bigger 0 is a starting position

        ImagePlus imp = img.duplicate();
        ImageProcessor impIP = imp.getProcessor();

        //IJ.run(imp, "Mean...", "radius=2");
```

```
        IJ.run(imp, "Gaussian Blur...", "sigma=2");
        IJ.run(imp, "Unsharp Mask...", "radius="+sigma+" mask="+ weight);
        IJ.setAutoThreshold(imp, "Otsu dark");
        Prefs.blackBackground = true;
        backgroundMin = imp.getDisplayRangeMin();
        IJ.log("lower backround value : " + backgroundMin);
        IJ.run(imp, "Convert to Mask", "");
        IJ.run(imp, "Ultimate Points", "");
        for(int x=1;x<imp.getWidth()-1;x++){
            for(int y=1;y<imp.getHeight()-1;y++){

                int[] intensity = imp.getPixel(x, y);//.getPixel(x,y);

                if(intensity[0]>=1){

                    startPoints.add(new Point(x,y));
                    impIP.set(x, y, 255);
                }
            }
        }
        IJ.run(imp, "Convert to Mask", "");

        IJ.log("I found " + startPoints.size() + " start positions");

        cleanUpStartPoints();

        IJ.log("I cleaned start positions up to : " + startPoints.size());

        voronoiImage = IJ.createImage("Voronoi Mask", img.getWidth(),
                                      img.getHeight(), 1, 8);
        voronoiIP = voronoiImage.getProcessor();

        //voronoi image has to be build by placing all the positiv starting points
        //inside empty image

        for(Point sPoint : startPoints){

            voronoiIP.set(sPoint.x, sPoint.y, 255);
        }

        IJ.run(voronoiImage, "Convert to Mask", "");
        IJ.run(voronoiImage, "Voronoi", "");
    }

    void cleanUpStartPoints(){

        //ArrayList<Point> deletableStartPoints = new ArrayList<Point>();

        int startPointsLength = startPoints.size();

        //for(int s1 = 0; s1 < startPointsLength; s1++){
        for(int s1 = startPointsLength - 1; s1 >= 0; s1--){
            Point p = startPoints.get(s1);

            //for(int s2 = 0; s2 < startPointsLength; s2++){
            for(int s2 = startPointsLength - 1; s2 >= 0; s2--){
                Point pp = startPoints.get(s2);
                double measuredDist = dist(p.x,p.y,pp.x,pp.y);
```

```
            if((int)measuredDist <= minDiameter && (int)measuredDist != 0){

                //deletableStartPoints.add(pp);
                startPoints.remove(pp);
                //startPoints.set(s2, new Point(0,0));
                startPoints.add(new Point(0,0));
            }
        }
    }

    for(int loc = startPoints.size()-1; loc >= 0; loc--){

        Point zP = startPoints.get(loc);

        if(startPoints.contains(new Point(0,0))){
            startPoints.remove(zP);
        }

    }

    Overlay ovl = new Overlay(); //startPositions

    for(Point ppp : startPoints){

        PointRoi pRoi = new PointRoi(ppp.x,ppp.y);
        pRoi.setPointType(2);
        ovl.add(pRoi);

    }

    img.setOverlay(ovl);
    img.show();

}

void grow4N(int x, int y){


    boolean noBorders = checkBorders(x,y);
    boolean m = checkMean(x,y);
    boolean d = checkDistance(x,y);
    boolean s = isSet(x,y);
    boolean neighborhoodReached = checkNeighborhood(x,y);

    if(!s && d && m && noBorders && !neighborhoodReached){

        maskIP.set(x, y, 255);
        c.add(x, y, ip.get(x, y));
        grow4N(x+1,y);
        grow4N(x-1,y);
        grow4N(x,y+1);
        grow4N(x,y-1);
    }
}

void grow8N(int x, int y){


    boolean noBorders = checkBorders(x,y);
```

```
        boolean m = checkMean(x,y);
        boolean d = checkDistance(x,y);
        boolean s = isSet(x,y);
        boolean neighborhoodReached = checkNeighborhood(x,y);

        if(!s && d && m && noBorders && !neighborhoodReached){

            maskIP.set(x, y, 255);
            c.add(x, y, ip.get(x, y));
            grow8N(x+1,y);
            grow8N(x-1,y);
            grow8N(x,y+1);
            grow8N(x,y-1);

            grow8N(x+1,y+1);
            grow8N(x-1,y+1);
            grow8N(x+1,y-1);
            grow8N(x-1,y-1);
        }
    }

    boolean checkBorders(int x, int y){

        if(x<ip.getWidth()-1 && x>1 && y<ip.getHeight()-1 && y>1){
            return true;
        }
        else return false;

    }

    boolean checkDistance(int x, int y){

        //check if the distance is not to far away
        Point p = startPoints.get(startPointIndex);
        double d = dist(x,y,p.x,p.y);

        if(d>distLimit){
            return false;
        }
        else return true;

    }

    boolean isSet(int x, int y){

        try {
            int intens = maskIP.get(x, y);
            if(intens>0){
            return true;
            }
            else return false;
        }
        catch (ArrayIndexOutOfBoundsException e) {

        return false;
        }

    }
```

```
boolean checkMean(int x, int y){

    double mean = 0;
    //IJ.log("mean : " + c.mean());

    try {
        mean += (double)ip.get(x, y);
        mean += (double)ip.get(x+1,y);
        mean += (double)ip.get(x-1,y);
        mean += (double)ip.get(x,y+1);
        mean += (double)ip.get(x,y-1);
        mean = (mean*meanTollerance)/5;
        //IJ.log("current mean : " + mean);

        if(mean >= backgroundMin &&  c.mean() <= mean){

            return true;
        }
        else return false;

    }

    catch (ArrayIndexOutOfBoundsException e) {
        return false;
    }

}

boolean checkNeighborhood(int x, int y){

    // to accelerate computation create voronoi map and check if x,y
    // is touching the voronoi border

    boolean neighborReached = false;

    if(voronoiIP.get(x, y) > 1){
        neighborReached = true;
    }

    return neighborReached;
}

public static double dist(int x1, int y1, int x2, int y2){

    double d = Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
    return d;
}

void calcDistLimit(int maxArea){

    //distLimit = Math.sqrt(maxArea*4/Math.PI);
    distLimit = maxDiameter;///2;
}


void getCellOutlines(){

    ImagePlus borderImage = IJ.createImage("border mask", img.getWidth(),
                                                    img.getHeight(), 1, 8);
```

```
        ImageProcessor borderIP = borderImage.getProcessor();

        for(cell cc : cells){

            ArrayList<Point> outlinePixels = cc.getOutline();
            //IJ.log("outlinePixels size : " + cc.borderPoints.size());

            for(Point p : outlinePixels){

                borderIP.set(p.x, p.y, 255);
            }
        }

        borderImage.show();

    }
    void show(){

        IJ.log("total cell in cells array : " + cells.size());
        int cellCount = 0;

        for(cell cc : cells){

            if(cc.area >= minArea && cc.area <= maxArea){

                cellCount += 1;

            }

            else{

                for(int i = 0; i<cc.points.size(); i++){

                    Point p = cc.points.get(i);
                    maskIP.set(p.x, p.y, 0);
                }
            }

        }

        IJ.log("cells in alowed size : " + cellCount);
        maskImage.show();
    }

    void showAnySizeCells(){

        IJ.log("total cell in cells array : " + cells.size());
        int cellCount = 0;

        for(cell cc : cells){

            if(cc.area >1){

                cellCount += 1;
            }
        }

        IJ.log("cells in any size (bigger than 1 px) : " + cellCount);
        maskImage.show();
```

```
    }

    void analyseMask(){

        int options = ParticleAnalyzer.ADD_TO_MANAGER+
             ParticleAnalyzer.EXCLUDE_EDGE_PARTICLES+
             ParticleAnalyzer.INCLUDE_HOLES+
             ParticleAnalyzer.CLEAR_WORKSHEET;
        int measurements = Measurements.AREA+Measurements.CENTROID+
             Measurements.CIRCULARITY;
        ResultsTable rt = new ResultsTable();
        ParticleAnalyzer pa = new ParticleAnalyzer(options,measurements,rt,
            (double) minArea, (double) maxArea);
        pa.analyze(maskImage);

        rt.show("measurements of found objects");
    }
}
```

Plot2CH

```
import java.awt.Button;
import java.awt.Color;
import java.awt.Panel;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import ij.CompositeImage;
import ij.IJ;
import ij.ImagePlus;
import ij.ImageStack;
import ij.WindowManager;
import ij.gui.GUI;
import ij.gui.GenericDialog;
import ij.gui.Overlay;
import ij.gui.Plot;
import ij.gui.Roi;
import ij.plugin.frame.PlugInFrame;
import ij.process.ImageProcessor;

public class Plot_2CH extends PlugInFrame implements ActionListener {

    // defining global variables
    ImagePlus origImage;
    ImagePlus gatedCellImage;
    String[] roiPositions;
    int[] roiPositionsNumber;
    String[] stainingList;
    int[] stainingListNumber;
    Roi[] rois;
    float[][] plotValues;
    Overlay plotOvlContainer;

    String[] plotChoice;
    ImagePlus img;
    ImageStack ims;

    Panel panel;
    Button b1,b2;
```

```
String title = "Plot_2CH";
WindowManager wm;

public Plot_2CH(){

    // create little panel with buttons for plotting and gating tool
    super("Plot_2CH");
    this.setTitle("Plot 2 CH");
    this.setSize(200,200);
    panel = new Panel();
    b1 = new Button("plot");
    b1.addActionListener(this);
    b2 = new Button("get gated cells");
    b2.addActionListener(this);
    panel.add(b1);
    panel.add(b2);
    this.add(panel);
    GUI.center(this);//panel.setVisible(true);
}
public void actionPerformed(ActionEvent e){

    // button pressed action observation

    if(e.getSource() == this.b1){
        initImagesAndPlot();
    }

    if(e.getSource() == this.b2){
        showGatedCells();
    }
}

public void run(String arg) {
    // main function for ImageJplugin
    img = IJ.getImage();
    ims = img.getImageStack();
    origImage = img;

        if(!checkInput(img)){
            IJ.error("make sure your segmented images are in a stack");
            return;
        }
        else{
            //initImages();
            Plot_2CH p2ch = new Plot_2CH();
            p2ch.setVisible(true);
        }
    }

void initImagesAndPlot(){

    // initialize images, extract two channels, plot their mean intensity

    //get input from user which channels should be plotted
    //select stack positions, get rois of these stainings, plot them
    img = IJ.getImage();
    ims = img.getImageStack();
    origImage = img;
```

```
        Overlay ovl = img.getOverlay();
        rois = ovl.toArray();
        roiPositions = new String[rois.length];
        roiPositionsNumber = new int[rois.length];
        stainingList = new String[img.getNSlices()];
        stainingListNumber = new int[img.getNSlices()];

        for(int i = 0; i<rois.length; i++){

             roiPositions[i] = ims.getShortSliceLabel(rois[i].getPosition());
             roiPositionsNumber[i] = i;
             //IJ.log("roiPosition : " + roiPositions[i]);
        }

        for(int j=0; j<stainingList.length; j++){

            stainingList[j] = ims.getShortSliceLabel(j+1);
            stainingListNumber[j] = j;
            //IJ.log("stainingList : " + stainingList[j]);
        }

        plotChoice = get2CH(); // get the string which ch should be plotted

        if(plotChoice[0].equals("-1") && plotChoice[1].equals("-1")){
            return;
        }
        else{
            plot(plotChoice);
        }
    }

    boolean checkInput(ImagePlus img){

        // plugin requires segmented image stack
        boolean check;
        int sliceNum = img.getNSlices();
        Overlay ovl = img.getOverlay();

        if(sliceNum <= 1 || ovl == null){
            check = false;
        }
        else check = true;

        return check;

    }
    String[] get2CH(){

        // ask user which ch he would like to plot against each other
        String[] userChoice = new String[2];

        GenericDialog gd =
                    new GenericDialog("Which Channels you would like to plot?");
        gd.addChoice("CH 1 :", stainingList, stainingList[0]);
        gd.addChoice("CH 2 :", stainingList, stainingList[1]);
        gd.showDialog();

        if(gd.wasCanceled()){
            userChoice[0] = "-1";
```

```
            userChoice[1] = "-1";
        }

        if(gd.wasOKed()){
            userChoice[0] = gd.getNextChoice();
            userChoice[1] = gd.getNextChoice();
        }

        return userChoice;

    }

    void plot(String[] plotChoice){

        // plot the values of the two channels
        plotValues = getPlotValues(plotChoice);

        Plot p = new Plot("plot of "+ plotChoice[0] +
                    " and " + plotChoice[1], plotChoice[0], plotChoice[1]);
        p.addPoints(plotValues[0], plotValues[1], Plot.CROSS);
        p.show();

    }
    float[][] getPlotValues(String[] plotChoice){

        // extract the objects out of the stack and calculate their mean intensity
        // and create an image to show only the two channels and their selected
       // cell objects

        int countedRoi = 1;
        int stackPos1 = 0;
        int stackPos2 = 0;

        for(int s = 0; s < stainingList.length; s++){

            if(plotChoice[0] == stainingList[s]){
                stackPos1 = stainingListNumber[s];
            }

            if(plotChoice[1] == stainingList[s]){
                stackPos2 = stainingListNumber[s];
            }
        }

        ImagePlus imp = origImage.duplicate();//new ImagePlus("duplicated stack");
        ImageStack ims = imp.getImageStack();
        ImageProcessor ip1 = ims.getProcessor(stackPos1+1);
        ImageProcessor ip2 = ims.getProcessor(stackPos2+1);

        for(int i = 0; i<rois.length; i++){

            if(plotChoice[0].equals(roiPositions[i])){
                countedRoi += 1;
            }
        }

        float[][] plotValues = new float[2][countedRoi];
        float meanValue1, meanValue2, area;
        int roiCount = 0;
```

```
        plotOvlContainer = new Overlay();

        for(int j = 0; j<rois.length; j++){

            meanValue1 = 0;
            meanValue2 = 0;
            area = 0;

            if(plotChoice[0].equals(roiPositions[j])){

                plotOvlContainer.add(rois[j]);

                for(Point p : rois[j]){

                    meanValue1 += ip1.getf(p.x,p.y);
                    meanValue2 += ip2.getf(p.x, p.y);
                    area += 1;
                }

                plotValues[0][roiCount] = meanValue1/area;
                plotValues[1][roiCount] = meanValue2/area;
                roiCount++;
            }
        }

        ImageStack ims2 = new ImageStack(ip1.getWidth(),ip1.getHeight());
        ims2.addSlice(ip1);
        ims2.addSlice(ip2);
        gatedCellImage = new ImagePlus("gated cell image", ims2);
        plotOvlContainer.setStrokeColor(Color.YELLOW);
        gatedCellImage.setOverlay(plotOvlContainer);
        //IJ.run(gatedCellImage, "Make Composite", "display=Composite");

        CompositeImage ci = new CompositeImage(gatedCellImage);
        ci.setMode(CompositeImage.COMPOSITE);
        //ci.show();
        gatedCellImage = ci;
        gatedCellImage.show();

        return plotValues;

    }

    void showGatedCells(){

        // gating tool inside the plot

        ImagePlus pw;
        Roi plotRoi;
        Roi[] cellRois;
        Rectangle r;
        double xOrigin, yOrigin, xWidth, yHeigth;

        IJ.selectWindow("plot of "+ plotChoice[0] + " and " + plotChoice[1]);
        pw = IJ.getImage();
        plotRoi = pw.getRoi();

        if(plotRoi == null){
            IJ.error("Create a gate, please");
```

```
            return;
        }

        r = plotRoi.getBounds();

        //calculating the real position of the bounding rect in the scaled
        //plot image

        xOrigin = pw.getCalibration().xOrigin;
        yOrigin = pw.getCalibration().yOrigin;
        xWidth = pw.getCalibration().pixelWidth;
        yHeigth = pw.getCalibration().pixelHeight;

        cellRois = plotOvlContainer.toArray();
        Overlay gatedCellsOvl = new Overlay();

        for(int i = 0; i < plotValues[0].length-1; i++){

            double plotValueX = Math.round(plotValues[0][i]);
            double plotValueY = Math.round(plotValues[1][i]);
            int count = 0;

            for(Point p : plotRoi){

                double px = Math.round(((double)p.x - xOrigin)*xWidth);
                double py = Math.round((yOrigin - (double)p.y)*yHeigth);

                if(plotValueX == px && plotValueY == py && count == 0){

                    //IJ.log("plot x : " + Math.round(plotValues[0][i]));
                    //IJ.log("plot y : " + Math.round(plotValues[1][i]));
                    //IJ.log("p.x    : " + px);
                    //IJ.log("p.y    : " + py);
                    //IJ.log("match " + (count+1+i));

                    gatedCellsOvl.add(cellRois[i]);

                    count++;

                }
            }

        }

        gatedCellsOvl.setStrokeColor(Color.YELLOW);
        gatedCellImage.setOverlay(gatedCellsOvl);
        gatedCellImage.updateAndDraw();
    }


}
```

8.1.3. markerSpecifiedSearching – Fiji/imageJ PlugIn

```
import java.awt.Color;
import java.awt.Point;
import java.awt.Rectangle;
import java.util.ArrayList;

import ij.IJ;
import ij.ImagePlus;
import ij.ImageStack;
import ij.gui.GenericDialog;
import ij.gui.Overlay;
import ij.gui.Roi;
import ij.measure.ResultsTable;
import ij.plugin.PlugIn;

public class markerSpecifiedSearching implements PlugIn {

    Roi[] rois;
    boolean[] countedRoi;
    boolean[] markerValues;
    String[] imageNames;
    String[] imageList;
    ImagePlus resultImage;

    ArrayList<ArrayList<Integer>> neighborCollection;
    ArrayList<ArrayList<String>> neighborCollectionNames;

    public void run(String arg) {

        ImagePlus img = IJ.getImage();
        ImageStack ims = img.getImageStack();
        resultImage = img.duplicate();

        if(!checkInput(img)){
            IJ.error("make sure your segmented images are in a stack");
        }
        else {

            Overlay ovl = img.getOverlay();
            rois = ovl.toArray();
            imageNames = new String[rois.length];
            imageList = new String[img.getNSlices()];
            countedRoi = new boolean[rois.length];

            for(int i = 0; i<rois.length; i++){

                 imageNames[i] = ims.getShortSliceLabel(rois[i].getPosition());
                 countedRoi[i] = false;
            }
            for(int j=0; j<imageList.length; j++){

                imageList[j] = ims.getShortSliceLabel(j+1);

            }

            markerValues = new boolean[imageList.length];
            markerValues = getMarkerCombination();
            int gdCanceldCount = 0;
```

```
            for(int k = 0; k < markerValues.length; k++){

                if(!markerValues[k])
                gdCanceldCount += 1;
            }

            if(markerValues.length == gdCanceldCount) return;

            searchMarkerCombination();
        }
    }

    boolean checkInput(ImagePlus img){

        boolean check;
        int sliceNum = img.getNSlices();
        Overlay ovl = img.getOverlay();

        if(sliceNum <= 1 || ovl == null){
            check = false;
        }
        else check = true;

        return check;
    }

    boolean[] getMarkerCombination(){

        boolean[] markerValue = new boolean[imageList.length];

        //IJ.log("get user information about allowed marker combination...");
        GenericDialog gd = new GenericDialog("Set Marker Combination");
        gd.addMessage("checkbox checked means positive");
        gd.addMessage("checkbox unchecked means negative");

        for(int i=0; i<imageList.length; i++){

            gd.addCheckbox(imageList[i], false);
        }
        gd.showDialog();
        for(int j=0; j<imageList.length; j++){

            if(gd.wasCanceled()){
                markerValue[j] = false;
            }
            else{
                markerValue[j] = gd.getNextBoolean();
            }

        }

        return markerValue;
    }

    void searchMarkerCombination(){

        //IJ.log("search for marker combination...");
        IJ.showStatus("looking for positiv combinations...");
```

```
        neighborCollection = new ArrayList<ArrayList<Integer>>();
        neighborCollectionNames = new ArrayList<ArrayList<String>>();
        //IJ.log("look for combinations...");

        for(int i=0; i<rois.length;i++){

            IJ.showProgress(i+1, rois.length);
            IJ.showStatus("looking for positiv combinations...");
            //Roi selectedRoi = rois[i];
            boolean gotNeighbors = false;
            ArrayList<Integer> foundNeighbors = new ArrayList<Integer>();
            ArrayList<String> neighborNames = new ArrayList<String>();

            for(int j=0; j<rois.length;j++){

                //IJ.log("compare with roi " + j);
                //Roi roiCandidate = rois[j];
                boolean isCandidate = false;

                if(checkCenterDist(rois[i],rois[j])){

                    for(Point p : rois[j]){

                        if(rois[i].contains(p.x, p.y)){
                            isCandidate = true;
                            //IJ.log("roi " + j + " is candidate");
                            break;
                        }
                    }

                    if(isCandidate && !countedRoi[j]){

                        gotNeighbors = true;
                        setCounted(j);
                        foundNeighbors.add(j);
                        neighborNames.add(imageNames[j]);
                        //IJ.log("roi : " + i);
                        //IJ.log("add " +imageNames[j]);

                    }
                }
            }

            if(gotNeighbors){

                foundNeighbors.add(0, i);
                neighborNames.add(0, imageNames[i]);
                neighborCollection.add(foundNeighbors);
                neighborCollectionNames.add(neighborNames);
            //  IJ.log("add neighbors from roi " + i + " to " + imageNames[i]);
            }

            resetCountedRoi();
        }

        proveMatches();

    }
```

```
    void setCounted(int roiIndex){

        countedRoi[roiIndex] = true;

    }

    void resetCountedRoi(){

        for(int i = 0; i<countedRoi.length; i++){

            countedRoi[i] = false;
        }
    }

    boolean checkCenterDist(Roi roi1, Roi roi2){

        boolean isNear = false;
        double distance = 0;
        double doubleWidth, doubleHeight;

        Rectangle rect1 = roi1.getBounds();
        Rectangle rect2 = roi2.getBounds();

        distance = dist(rect1.x, rect1.y, rect2.x, rect2.y);
        doubleWidth = (double) (rect1.width+rect2.width);
        doubleHeight = (double) (rect1.height+rect2.height);

        if(doubleWidth > distance || doubleHeight > distance){

            isNear = true;
        }

        return isNear;
    }

    double dist(int x1, int y1, int x2, int y2){

        double d = Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
        return d;
    }

    void proveMatches(){

        for(int i=neighborCollectionNames.size()-1; i>=0; i--){

            ArrayList<String> list = neighborCollectionNames.get(i);
            boolean[] checkValues = new boolean[markerValues.length];
            int c = 0;

            for(int j=0; j<markerValues.length; j++){

                if(list.contains(imageList[j])){
                    checkValues[j] = true;
                }
                else{
                    checkValues[j] = false;
                }
            }
```

```
            for(int k=0; k<markerValues.length; k++){

                if(markerValues[k] == checkValues[k]) {

                    c += 1;
                }
            }

            if(c != markerValues.length){

                neighborCollection.remove(i);
                neighborCollectionNames.remove(i);
            }
        }
        if(neighborCollection.isEmpty()){
           IJ.error("Sorry, but I have not found any of your marker combination");
        }
        else{

            showMarkerCombinations();
        }

    }

    void showMarkerCombinations(){

        Overlay rOvl = resultImage.getOverlay();
        Roi[] rRois = rOvl.toArray();

        //IJ.log("showMarkerCombination...");


        for(int i=0; i<neighborCollectionNames.size(); i++){

            ArrayList<Integer> list = neighborCollection.get(i);
          //IJ.log("got list : " + neighborCollection.get(i));
          //IJ.log("neighborCollectionNames : " + neighborCollectionNames.get(i));

            for(int e : list){

                rRois[e].setStrokeColor(Color.green);
                rOvl.add(rRois[e]);
            }

        }

        int[] totalCellCount = new int[imageList.length];
        int[] matches = new int[imageList.length];
        double[] matchRate = new double[imageList.length];

        for(int j=0; j<imageList.length; j++){

            for(int k=0; k<rRois.length; k++){

                if(rRois[k].getPosition() == (j+1)){

                    totalCellCount[j] += 1;

                    if(rRois[k].getStrokeColor() == Color.green){
```

```
                        matches[j] += 1;
                    }
                }
            }
            matchRate[j] = ((double)matches[j]/(double)totalCellCount[j])*100;
        }

        ResultsTable rt = new ResultsTable();

        for(int m=0; m<imageList.length; m++){

            rt.incrementCounter();
            rt.addValue("staining", imageList[m]);
            rt.addValue("cell count", totalCellCount[m]);
            rt.addValue("matches", matches[m]);
            rt.addValue("%",matchRate[m]);
        }
        rt.show("Marker Searching Result");

        resultImage.setTitle("Marker Searching Result");
        resultImage.setOverlay(rOvl);
        resultImage.show();

    }

}
```

8.1.4. DFT Coefficient Calculation of object outlines – ImageJ Macro

```
// getCoordinatesAndPlotDFT2
// This Macro reads ROIs and calculates the discrete fourier transformation (DFT)
// discriptors, plots and if you want save them, based on SHADE

Dialog.create("getCoordinatesAndPlotDFT2");
Dialog.addMessage("please make sure you have an opened image &\n"+
                  "at least one selected cell in ROI Manager !\n");
Dialog.addNumber("how many DFT Discriptors you need?", 20);
Dialog.addCheckbox("save your results?", false);
Dialog.addCheckbox("show intermediate results", false);
Dialog.addMessage("-------------------------------------------");
Dialog.addCheckbox("animate surface positions of ROI?", false);
Dialog.addMessage(" wanring, this choice slows your process down");
Dialog.show();

DFTCount = Dialog.getNumber();
saving = Dialog.getCheckbox();
intermediate = Dialog.getCheckbox();
animation = Dialog.getCheckbox();

name=getInfo("image.filename");
imageDir = getInfo("image.directory");
path = imageDir+"DFT_calculation_cell";

origImage = getTitle();
origHeight = getHeight();
origWidth = getWidth();

run("ROI Manager...");
cellCount = roiManager("count");

for (cell=0; cell<cellCount; cell++) {

    roiManager("select",cell);
    Roi.getCoordinates(x, y);
    roiName = Roi.getName;
    n = x.length;

    if (animation == true) {
        run("Point Tool...", "type=Dot color=Yellow size=Medium show counter=0");
        for (i=0; i<n; i++){
        IJ.log("x :" + x[i] + "; y: " + y[i]);
        makePoint(x[i], y[i]);
        wait(50);
        }

    }

    oreal = newArray(n);
    oimag = newArray(n);
    oAmpl = newArray(n);
    natemp = newArray(DFTCount);
    toreal = newArray(DFTCount);
    toimag = newArray(DFTCount);
    toAmpl = newArray(DFTCount);
    tnumber = newArray(DFTCount);
```

```
    for (k = 0; k < n; k++) {

        creal = 0;
        cimag = 0;

        for (t = 0; t < n; t++) {

            creal = creal + x[t]*cos(2*PI * t * k / n) +
                            y[t]*sin(2*PI * t * k / n);
            cimag = cimag + x[t]*sin(2*PI * t * k / n) +
                            y[t]*cos(2*PI * t * k / n);
        }
        oreal[k] = creal;
        oimag[k] = cimag;
        oAmpl[k] = sqrt(creal*creal + cimag*cimag);

    }
    for (temc=1; temc < DFTCount; temc++){
        toreal[temc]=oreal[temc];
        toimag[temc]=oimag[temc];
        toAmpl[temc]=oAmpl[temc];
        natemp[temc]=roiName+"_image_"+name;
        tnumber[temc]=temc+1;
    }

    roreal = Array.concat(roreal,toreal);
    roimag = Array.concat(roimag,toimag);
    roAmpl = Array.concat(roAmpl,toAmpl);
    rname = Array.concat(rname,natemp);
    rnumber = Array.concat(rnumber,tnumber);
    Array.show("Results (row numbers)",roreal,roimag,roAmpl,rname,rnumber);

    if (intermediate == true) {
        Plot.create(name+" cell["+cell+"]->real","Timepoint","Value",toreal);
        Plot.show();

        Plot.create(name+" cell["+cell+"]->imaginary","Timepoint","Value",toimag);
        Plot.show();

        Plot.create(name+" cell["+cell+"]->Amplitude","Timepoint","Value",toAmpl);
        Plot.show();

        waitForUser("next cell?");
    }

}

    if(saving==true){

        selectWindow("Results");
        saveAs("Results",path+"_DFT_results_of"+name+".xls");
        //IJ.log("Results of calculations are saved "+
        //"("+path+"_DFT_results_of"+name+".xls)");
        waitForUser("Results of calculations are saved\n "+
        "("+path+"_DFT_results_of"+name+".xls)");
    }

waitForUser("Process completed!");
```

8.1.5. MELC registration scripts – Python

```
# MELC_image_registration_GUI
#----------------------------
import tkinter as tk
from tkinter import filedialog
import MELC_run_reader
import registrate_MELC_run


class Main_Window:

    # constructor of the main gui
    def __init__(self):

        self.melc_dict = {}
        self.melc_run_info = MELC_run_reader

        self.master = tk.Tk()
        self.master.title("MELC run registration")
        # self.master.geometry("300x450+5+10")
        self.frame = tk.Frame(self.master)

        self.file_loc_label = tk.Label(self.master, text=
                   "directory").grid(row=0, column=0, rowspan=1, columnspan=1)
        self.path_entry = tk.Entry(self.master, width=40)
        self.path_entry.insert(0, "melkIni path")
        self.path_entry.grid(row=0, column=1, rowspan=1, columnspan=2)

        self.browse_button = tk.Button(self.master, text="browse",
                         command=self.set_ini_file).grid(row=0, column=4,
                         rowspan=1, columnspan=1)

        self.sec_label = tk.Label(self.master,
             text="marker list, select secondary markers").grid(row=1, column=2)

        self.list_box = tk.Listbox(self.master,
             width=40, height=20, selectmode='multiple')
        self.list_box.grid(row=2, column=2)

        self.run_button = tk.Button(self.master, text="RUN",
             command=self.start_melc_registration).grid(row=4,
             column=4, rowspan=2, columnspan=2)
        #print(self.master.grid_size())

        self.master.mainloop()

    def set_ini_file(self):

        ini_file_path = filedialog.askdirectory()
        self.path_entry.delete(0, len(self.path_entry.get()))
        self.path_entry.insert(0, ini_file_path)
        self.melc_run_info = self.get_MELC_run_info()

    def get_MELC_run_info(self):

        melc_run_dir = self.path_entry.get()
        melc_run_path = melc_run_dir+'/inifile/melkIni.xml'
        mri = MELC_run_reader.MELC_run_reader(melc_run_path)
        self.melc_dict = mri.MELC_dict
```

```
        count = 0
        for i in self.melc_dict:

            self.list_box.insert(count, self.melc_dict[i]["marker name"])
            count += 1

        return mri

    def start_melc_registration(self):

        secondary_markers =
             [self.list_box.get(i) for i in self.list_box.curselection()]

        print("start processing...")
        print("")

        for image_set in self.melc_run_info.MELC_dict:

            secondary = "no"

            if self.melc_run_info.MELC_dict[image_set]["marker name"] in secondary
            _markers:
                secondary = self.melc_run_info.MELC_dict[image_set]["secondary"] =
                                                                            "yes"

            f_channel = self.melc_run_info.MELC_dict[image_set]['channel']
            f_exposure = self.melc_run_info.MELC_dict[image_set]['exposure time']
            step = int(self.melc_run_info.MELC_dict[image_set]['inc step'])

            for image_set2 in self.melc_run_info.MELC_dict:

                f_channel2 =
                   self.melc_run_info.MELC_dict[image_set2]['channel']
                f_exposure2 =
                   self.melc_run_info.MELC_dict[image_set2]['exposure time']
                previous_step =
                   (int(self.melc_run_info.MELC_dict[image_set2]['inc step']))


                if f_channel == f_channel2 and
                   f_exposure == f_exposure2 and
                   (step-2) == previous_step and
                   secondary == "yes":

                    self.melc_run_info.MELC_dict[image_set]\
                   ['previous bleach image'] =
                   self.melc_run_info.MELC_dict[image_set2]['marker name'] +
                   '_' + f_exposure2 + '_' + f_channel2 + '_' +
                   self.melc_run_info.MELC_dict[image_set2]['inc step']
                    # print(step,previous_step)


        registrate_MELC_run.registrate_MELC_run(self.melc_run_info)


start_main_window = Main_Window()
```

```
# MELC_run_reader – python script to parse MELC run XML file
#-----------------
import xml.etree.ElementTree as ET
import os
'''
MELC_run_reader read the melkIni.xml file and creates a dictionary,
containing every step
'''
class MELC_run_reader:
    # some global variables
    web_prefix = ''
    MELC_dict = {}

    def __init__(self, path):

        self.path = path
        self.melc_run_dir = self.path[0:-(len('/inifile/melkIni.xml'))]

        # parsing xml to element tree
        inifile = ET.ElementTree()
        inifile = ET.parse(path)
        inifile_root = inifile.getroot()

        global web_prefix
        web_prefix = self.check_and_get_web_prefix(inifile_root.tag)

        # extract file format, stack size, max shift [x y] for registration
        self.fluorescence_prefix = inifile_root.find((web_prefix + 'runSetting/' +
 web_prefix + 'fluorescenceImagePrefix')).text
        self.bleach_prefix = inifile_root.find((web_prefix + 'runSetting/' +
web_prefix + 'bleachImagePrefix')).text
        self.phase_prefix = inifile_root.find((web_prefix + 'runSetting/' +
web_prefix + 'phaseImagePrefix')).text
        self.bleach_phase_prefix = inifile_root.find((web_prefix + 'runSetting/' +
 web_prefix + 'phaseBleachImagePrefix')).text

        self.file_format = inifile_root.find((web_prefix + 'runSetting/' +
web_prefix + 'imageFormat')).text

        self.stack_range_m = inifile_root.find((web_prefix + 'runSetting/' +
web_prefix + 'visualFieldConfig/' + web_prefix + 'stack/' + web_prefix +
'imageCountNegative')).text
        self.stack_range_p = inifile_root.find((web_prefix + 'runSetting/' +
web_prefix + 'visualFieldConfig/' + web_prefix + 'stack/' + web_prefix +
'imageCountPositive')).text

        self.max_shift_x = inifile_root.find((web_prefix + 'autofocus/' +
web_prefix + 'autofocusStep/' + web_prefix + 'function/' + web_prefix +
'correlation/' + web_prefix + 'maxShiftX')).text
        self.max_shift_y = inifile_root.find((web_prefix + 'autofocus/' +
web_prefix + 'autofocusStep/' + web_prefix + 'function/' + web_prefix +
'correlation/' + web_prefix + 'maxShiftY')).text

        self.reference_image = self.get_reference_image()
        self.reference_image_exposure_time = inifile_root.find((web_prefix +
'runSetting/' + web_prefix + 'exposureTime')).text
        self.reference_image_channel = inifile_root.find(web_prefix +
```

```
'runSetting/' + web_prefix + 'phaseFilter').attrib["name"]
        self.reference_image_size = inifile_root.find((web_prefix + 'autofocus/' +
 web_prefix + 'autofocusStep/' + web_prefix + 'roiSize')).text
        self.MELC_dict = self.get_MELC_dict(inifile_root)

        print('')
        print('melc run directory : ', self.melc_run_dir)
        print('-------------')
        print('file prefixes')
        print('-------------')
        print('fluorescence_prefix : ', self.fluorescence_prefix)
        print('bleach_prefix : ', self.bleach_prefix)
        print('phase_prefix : ', self.phase_prefix)
        print('bleach_prefix : ', self.bleach_phase_prefix)
        print('')
        print('file format : ' + self.file_format)
        print('')
        print('reference image : ', self.reference_image)
        print('reference image exposure time : ',
                                        self.reference_image_exposure_time)
        print('reference image channel : ', self.reference_image_channel)
        print('reference image size [%] : ', self.reference_image_size)
        print('')
        print('stack size : ' + str(int(self.stack_range_m) + 1 +
             int(self.stack_range_p)) + ' [' + self.stack_range_m + ' ; ' +
             self.stack_range_p + ' ]')  # adding one, because of central position
        print('')
        print('max shift [x y] : [' + self.max_shift_x + ' ; ' +
             self.max_shift_y + ']')
        print('')
        print('your MELC run in detail:')
        print('-----------------------\n')
        self.show_MELC_dict(self.MELC_dict)


    def get_reference_image(self):

        all_files = os.listdir((self.melc_run_dir) + '/source/')
        matched = [i for i in all_files if ("_RefImage" in i and
             i.endswith(self.file_format))]
        # print(matched)
        return matched


    def check_and_get_web_prefix(self, tag):
        '''
        checking some atribute naming of the nodes inside the parsed xml
        '''

        if len(tag) > 7:
            return tag[0:-7]
        else:
            return ''


    def show_MELC_dict(self, MELC_dict):

        for melc_step in MELC_dict:
            print(melc_step + ': ')
```

```
            for i in MELC_dict[melc_step]:
                print(i + ' : ' + MELC_dict[melc_step][i])
            print('-------------------------------------')


    def get_MELC_dict(self, inifile_root):
        '''
        returns dictionary containing staining step information
         melc_step_info = {image_set_1 : {inc step, channel, marker name, expsure
         time, previous bleach image, secondary?}}
        '''
        inc_step_dict = {}
        inc_step_count = 1
        image_set_count = 1

        for inc_step in inifile_root.iter((web_prefix) + 'incStep'):

            channel_step = inc_step.iter((web_prefix) + 'channelStep')
            channel_step_count = 1

            inc_step_prefix = '00'

            if inc_step_count >= 10 and inc_step_count < 100:
                inc_step_prefix = '0'

            if inc_step_count >= 100 and inc_step_count < 1000:
                inc_step_prefix = ''

            for e in channel_step:
                # print(e.tag, e.attrib)
                name = e.find((web_prefix) + 'marker')
                exposure_time = e.find((web_prefix) + 'exposureTime')
                channel = e.find((web_prefix) + 'fluorescenceFilter')

                image_set_text = 'image_set_' + str(image_set_count)
                inc_step_dict[image_set_text] = {
                    "inc step": inc_step_prefix + str(inc_step_count),
                    "channel step": str(channel_step_count),
                    "marker name": name.attrib['name'],
                    "channel": channel.attrib['name'],
                    "exposure time": exposure_time.text,
                    "previous bleach image": "",
                    "secondary": "no"
                }

                image_set_count += 1
                channel_step_count += 1

            inc_step_count += 1

        # add previous bleach image staining to each incStep

        for inc in inc_step_dict:

            f_channel = inc_step_dict[inc]['channel']
            f_exposure = inc_step_dict[inc]['exposure time']
            step = int(inc_step_dict[inc]['inc step'])
```

```
            if step == 1:
                inc_step_dict[inc]['previous bleach image'] =
inc_step_dict[inc]['marker name'] + '_' + f_exposure + '_' + f_channel +
'_' + inc_step_dict[inc]['inc step']

            else:


                for inc2 in inc_step_dict:

                    f_channel2 = inc_step_dict[inc2]['channel']
                    f_exposure2 = inc_step_dict[inc2]['exposure time']
                    previous_step = (int(inc_step_dict[inc2]['inc step']))

                    if f_channel == f_channel2 and f_exposure == f_exposure2 and
step > previous_step:
                        inc_step_dict[inc]['previous bleach image'] =
inc_step_dict[inc2]['marker name'] + '_' + f_exposure2 + '_' + f_channel2 + '_' +
inc_step_dict[inc2]['inc step']
                        # print(step,previous_step)

                if inc_step_dict[inc]['previous bleach image'] == "":
                    inc_step_dict[inc]['previous bleach image'] =
inc_step_dict[inc]['marker name'] + '_' + f_exposure + '_' + f_channel + '_' +
inc_step_dict[inc]['inc step']

        return inc_step_dict


# registrate_MELC_run - main syript with functions to registrate MELC run
#-----------------------------------------------------------------------
import os
import imageio
import numpy as np
import matplotlib.image as mpimg
from skimage.feature import register_translation
from scipy.ndimage import shift
from scipy import ndimage
from scipy import interpolate

def read_images(image_list):

    image_container = []

    for i in image_list:
        image_container.append(imageio.imread(i).astype('float'))

    loaded_images = np.array(image_container, dtype='float')

    return loaded_images


def createSuffix(m, p, file_format):
    if m < 0:
        steps = abs(m) + p

    if m > 0:
        steps = p - m
```

```
    suffix = [None] * (steps + 1)
    posCount = 0

    for s in range(m, p + 1):

        if s > 0:
            suffix[posCount] = '_p' + str(s) + file_format
            posCount += 1

        if s == 0:
            suffix[posCount] = file_format
            posCount += 1

        if s < 0:
            suffix[posCount] = '_m' + str(abs(s)) + file_format
            posCount += 1

    return suffix

def find_iamge_shift(ref_image, image):

    ref_image_center_y = int(ref_image.shape[0] / 2)
    ref_image_center_x = int(ref_image.shape[1] / 2)

    image_center_y = int(image.shape[0] / 2)
    image_center_x = int(image.shape[1] / 2)

    image = image[(image_center_y - ref_image_center_y):(image_center_y +
ref_image_center_y), (image_center_x - ref_image_center_x):(image_center_x +
ref_image_center_x)]

    shift_xy, error, diffphase = register_translation(ref_image, image, 10)
    return shift_xy

def get_interpolation_measure(input_image_b):

    dims = input_image_b.shape

    step_x = int(dims[1] / 8)
    step_y = int(dims[0] / 8)

    x_measure = []
    y_measure = []
    z_measure = []

    for sx in range(0, dims[1] + 1, step_x):
        for sy in range(0, dims[0] + 1, step_y):

            rx1 = int(sx - step_x / 2)
            ry1 = int(sy - step_y / 2)
            rx2 = int(sx + step_x / 2)
            ry2 = int(sy + step_y / 2)

            if rx1 < 0:
                rx1 = 0
            if rx2 > dims[1]:
                rx2 = dims[1]
            if ry1 < 0:
                ry1 = 0
```

```
            if ry2 > dims[0]:
                ry2 = dims[0]

            mean_value = np.min(input_image_b[ry1:ry2, rx1:rx2])

            x_measure.append(sx)
            y_measure.append(sy)
            z_measure.append(mean_value)

    return x_measure, y_measure, z_measure


def get_polynomial_interpolation(xs, ys, zs):

    tmp_A = []
    tmp_b = []

    for i in range(len(xs)):

        #under mac os float is normaly taken by default...under windows we have to
 tell explicity
        tmp_A.append([float(xs[i]*xs[i]), float(ys[i]*ys[i]), float(xs[i]*ys[i]),
float(xs[i]), float(ys[i]), 1.])
        tmp_b.append(zs[i])

    b = np.transpose(tmp_b)
    A = np.array(tmp_A)

    fit = np.linalg.inv((np.transpose(A) @ A)) @ np.transpose(A) @ b
    errors = b - A @ fit
    residual = np.linalg.norm(errors)

    print("solution:")
    print("%f x^2 + %f y^2 + %f x*y + %f x + %f y + %f = z" % (fit[0], fit[1], fit
[2], fit[3], fit[4], fit[5]))
    print("errors:")
    print("mean : " + str(np.mean(errors)))
    print("min : " + str(np.min(errors)))
    print("max : " + str(np.max(errors)))
    print("residual:")
    print(residual)

    return fit, errors, residual


def focus_measure(input_images):

    dims = input_images.shape
    global_mean = np.mean(input_images)
    step_x = int(dims[2] / 16)
    step_y = int(dims[1] / 16)

    x = []
    y = []
    z = []

    for sx in range(0, dims[2] + 1, step_x):
        for sy in range(0, dims[1] + 1, step_y):
```

```
            rx1 = int(sx - (step_x / 2))
            ry1 = int(sy - (step_y / 2))
            rx2 = int(sx + (step_x / 2))
            ry2 = int(sy + (step_y / 2))

            if rx1 < 0:
                rx1 = 0
            if rx2 > dims[2]:
                rx2 = dims[2]
            if ry1 < 0:
                ry1 = 0
            if ry2 > dims[1]:
                ry2 = dims[1]
            local_mean = np.mean(input_images[:, ry1:ry2, rx1:rx2])
            std_means = np.zeros(dims[0])

            for szx in range(dims[0]):
                std_means[szx] = np.std(input_images[szx, ry1:ry2, rx1:rx2])

            max_index_x = np.argmax(std_means)

            if local_mean >= global_mean:
                x.append(sx)
                y.append(sy)
                z.append(max_index_x)

    return x, y, z

def focus_measure2(input_images):
    dims = input_images.shape

    fm = np.zeros(dims)

    for i in range(fm.shape[0]):
        fm[i, :, :] = input_images[i, :, :] - ndimage.gaussian_filter(input_images
[i, :, :], sigma=9)
        fm[fm < 0] = 0
        fm[i, :, :] =  ndimage.gaussian_filter(fm[i, :, :], sigma=9)

    # global_mean = np.mean(input_images)
    global_mean = np.mean(fm)
    step_x = int(dims[2] / 32)
    step_y = int(dims[1] / 32)

    x = []
    y = []
    z = []

    for sx in range(0, dims[2] + 1, step_x):
        for sy in range(0, dims[1] + 1, step_y):

            rx1 = int(sx - (step_x / 2))
            ry1 = int(sy - (step_y / 2))
            rx2 = int(sx + (step_x / 2))
            ry2 = int(sy + (step_y / 2))

            if rx1 < 0:
                rx1 = 0
            if rx2 > dims[2]:
```

```
                rx2 = dims[2]
            if ry1 < 0:
                ry1 = 0
            if ry2 > dims[1]:
                ry2 = dims[1]

            # local_mean =np.mean(input_images[:, ry1:ry2, rx1:rx2])
            local_mean = np.mean(fm[:, ry1:ry2, rx1:rx2])
            std_means = np.zeros(dims[0])

            for szx in range(dims[0]):
                # std_means[szx] = np.std(input_images[szx, ry1:ry2, rx1:rx2])
                std_means[szx] = np.std(fm[szx, ry1:ry2, rx1:rx2])

            max_index_x = np.argmax(std_means)

            if local_mean >= global_mean:
                x.append(sx)
                y.append(sy)
                z.append(max_index_x)

    return x, y, z


def get_focus_map_fit(xs, ys, zs):

    print(len(xs))
    print(len(ys))
    print(len(zs))

    tmp_A = []
    tmp_b = []

    for i in range(len(xs)):

        tmp_A.append([xs[i], ys[i], 1])
        tmp_b.append(zs[i])

    b = np.transpose(tmp_b)
    A = np.array(tmp_A)
    fit = np.linalg.inv((np.transpose(A) @ A)) @ np.transpose(A) @ b
    errors = b - A @ fit
    residual = np.linalg.norm(errors)

    print("solution:")
    print("%f x + %f y + %f = z" % (fit[0], fit[1], fit[2]))
    print("errors:")
    print("mean : " + str(np.mean(errors)))
    print("min : " + str(np.min(errors)))
    print("max : " + str(np.max(errors)))
    print("residual:")
    print(residual)

    return fit, errors, residual


def gauss(x, a, b, c, d):
    return a + (b-a)*np.exp(-(x-c)**2./(2.*d**2))
```

```
def registrate_MELC_run(melc_run_info):
"""
here the whole magic happens and the raw images from melc run are processed (align
ed, subtracted, corrected and projected)
To Do !

- registrate phase and bleach_phase images -
> translate with the values fluorescent images (if possible, correct for chromatic
 shift xyz !!!)
    - subtract the registrated images
    - flat field correction ?
    - z stack projection for all in focus
 """

    print("")
    print("started...")
    print("")

    # create directory inside melc run directory

    subtracted_images_dir = melc_run_info.melc_run_dir +
'/results/subtracted_images'
    best_focus_dir = melc_run_info.melc_run_dir +
'/results/best_focus_images'
    # print("dir : ", subtracted_images_dir)

    if not os.path.exists(subtracted_images_dir):

        try:
            os.mkdir(subtracted_images_dir)
        except OSError:
            print("could not create subtracted_images directory")
        else:
            print("Error: subtracted_images folder is not created...")

    else:
        print("folder subtracted_images allready created")

    if not os.path.exists(best_focus_dir):

        try:
            os.mkdir(best_focus_dir)
        except OSError:
            print("could not create best_focus_images directory")
        else:
            print("Error: best_focus_images folder is not created...")

    else:
        print("folder best_focus_images allready created")

    # create list of all z positions
    position_suffix = createSuffix(-
int(melc_run_info.stack_range_m), int(melc_run_info.stack_range_p),
melc_run_info.file_format)
    # print(position_suffix)

    # aligning/registrating images based on reference phase contrast image
    # via cross correlation
```

```
    ph_ref_image = imageio.imread(melc_run_info.melc_run_dir + "/source/" + melc_r
un_info.reference_image[0]).astype('float')
    #ph_ref_image = np.array(ph_ref_image)
    # plt.imshow(ph_ref_image)
    # plt.show()

    for ms in melc_run_info.MELC_dict:

        image_stack = []
        inc_step = melc_run_info.MELC_dict[ms]["inc step"]
        channel_step = melc_run_info.MELC_dict[ms]["channel step"]
        marker_name = melc_run_info.MELC_dict[ms]["marker name"]
        marker_fexp_time = melc_run_info.MELC_dict[ms]["exposure time"]
        marker_f_channel = melc_run_info.MELC_dict[ms]["channel"]
        previous_bleach_image=melc_run_info.MELC_dict[ms]["previous bleach image"]
        ph_exposure_time = melc_run_info.reference_image_exposure_time
        ph_channel = melc_run_info.reference_image_channel

        if channel_step == "1":
            ph_image = imageio.imread(melc_run_info.melc_run_dir + "/source/p_" +
marker_name + '_' + ph_exposure_time + '_' + ph_channel + '_' + inc_step +
melc_run_info.file_format).astype('float')
            phb_image = imageio.imread(melc_run_info.melc_run_dir +
"/bleach/pb_" + marker_name + '_' + ph_exposure_time + '_' + ph_channel + '_' +
inc_step + melc_run_info.file_format).astype('float')

        shift_p = find_iamge_shift(ph_ref_image, ph_image)
        shift_pb = find_iamge_shift(ph_ref_image, phb_image)

        step = 0

        for z in position_suffix:

            if melc_run_info.MELC_dict[ms]["secondary"] == "no":
                f_image = imageio.imread(melc_run_info.melc_run_dir +
"/source/o_" + marker_name + '_' + marker_fexp_time + '_' + marker_f_channel +
'_' + inc_step + z).astype('float')
                b_image = imageio.imread(melc_run_info.melc_run_dir +
"/bleach/b_" + previous_bleach_image + z).astype('float')
                fb_image = imageio.imread(melc_run_info.melc_run_dir +
"/bleach/b_" + marker_name + '_' + marker_fexp_time + '_' + marker_f_channel +
'_' + inc_step + z).astype('float')
                print(marker_name + '_' + marker_fexp_time + '_' +
marker_f_channel + '_' + inc_step + z + ' - ' + previous_bleach_image + z)
            else:
                f_image = imageio.imread(melc_run_info.melc_run_dir +
"/source/o_" + marker_name + '_' + marker_fexp_time + '_' + marker_f_channel +
'_' + inc_step + z).astype('float')
                b_image = imageio.imread(melc_run_info.melc_run_dir +
"/source/o_" + previous_bleach_image + z).astype('float')
                fb_image = imageio.imread(melc_run_info.melc_run_dir +
"/bleach/b_" + marker_name + '_' + marker_fexp_time + '_' + marker_f_channel +
'_' + inc_step + z).astype('float')
                print(marker_name + '_' + marker_fexp_time + '_' +
marker_f_channel + '_' + inc_step + z + ' - ' + previous_bleach_image + z)

            # now, that we have our target images, we can at fist align them,
            # afterwards we subtract
            f_image = shift(f_image, shift_p, mode='nearest')
```

```
            b_image = shift(b_image, shift_pb, mode='nearest')
            fb_image = shift(fb_image, shift_pb, mode='nearest')

            fb_image = ndimage.gaussian_filter(fb_image, sigma=3, mode='nearest')
            subtracted = f_image - b_image
            subtracted[subtracted < 0] = 0.0

            x_measure, y_measure, z_measure = get_interpolation_measure(fb_image)

            dims = subtracted.shape

            grid_x, grid_y = np.mgrid[0:dims[1]:1, 0:dims[0]:1]
            grid_z2 = interpolate.griddata((x_measure, y_measure), z_measure,
(grid_x, grid_y), method='cubic')

            subtracted = subtracted/(grid_z2.T/np.mean(grid_z2))
            image_stack.append(subtracted_images_dir + '/o_' + marker_name + '_' +
 marker_fexp_time + '_' + marker_f_channel + '_' + inc_step + z[:-3] + 'tiff')

            subtracted[subtracted>(2**16)-1] = (2**16)-1
            imageio.imwrite(subtracted_images_dir + '/o_' + marker_name + '_' +
marker_fexp_time + '_' + marker_f_channel + '_' + inc_step +
z[:-3] + 'tiff', subtracted.astype('uint16'), format='TIFF')
            step += 1

        print(ms + " of " + str(len(melc_run_info.MELC_dict)) +
" are registered and subtracted")


        # calculating best focus image, to remove tilt in image
        images_for_best_focus = read_images(image_stack)
        dims = images_for_best_focus.shape

        xf, yf, zf = focus_measure2(images_for_best_focus)
        fit, errors, residual = get_polynomial_interpolation(xf, yf, zf)


        x_pos = np.arange(0, dims[2], 1)
        y_pos = np.arange(0, dims[1], 1)

        xx, yy = np.meshgrid(x_pos, y_pos)

        focus_map_new =xx*xx*fit[0]+yy*yy*fit[1]+xx*yy*fit[2]+
xx*fit[3]+yy*fit[4]+fit[5]
        focus_map_new[focus_map_new<0] = 0
        focus_map_new[focus_map_new>dims[0]] = dims[0]-1

        indices_array = np.zeros(dims)

        for ind in range(dims[0]):
            indices_array[ind, :, :] += ind

        sigma = 1
        gauss_along_z = gauss(indices_array, 0., 1., focus_map_new, sigma)
        fim = np.sum(images_for_best_focus * gauss_along_z, axis=0)
        best_focus_image = (fim / np.sum(gauss_along_z, axis=0))# * 2 ** 16
        best_focus_image = best_focus_image-np.min(best_focus_image)
        best_focus_image[best_focus_image>(2**16)-1] = (2**16)-1
```

```
        imageio.imwrite(best_focus_dir + '/' + marker_name + '_' +
marker_fexp_time + '_' + marker_f_channel + '_' + inc_step + '.tiff',
best_focus_image.astype('uint16'), format='TIFF')

        print("and best focus image is calculated")

    print("registration done...")
```

8.1.6. MELC data integration and unsupervised clustering – R

```
# MELC data integration and visualization
# loading the libraries
library(Seurat)
library(data.table)
library(dplyr)
library(ggplot2)
library(Rtsne)
library(uwot)
library(doParallel)
library(missForest)
library(missRanger)

# define file location and read the data, saved in variable "df"
# if dataset was loaded from saved dataset.R paths can differ...have look to
# envrionment variable
maindir <- choose.dir()
filename <- "Lungs_rawData.csv" # choose.files(default = maindir)
df <- fread(paste(maindir, filename, sep = "/"), encoding="UTF-8") # read the csv

# prepare folder for analysis results
save.path <- paste(maindir,paste(substr(filename,0,nchar(filename)-
4),"_analysis/"), sep = "/")
dir.create(save.path)

# sometimes different meta data headers inside  csv files (mainly the first two
# rows), therefore we have to to extract only the important data and convert to
# numeric values, otherwise numbers are interpreted as strings
df.values <- sapply(df[3:nrow(df),c(1:43)], as.numeric)
df.disease.state <- sapply(df[3:nrow(df),`disease state`], as.factor)
df.sampleIDs <- sapply(df[3:nrow(df),SourceID], as.factor)
df.object.ids <- 1:(nrow(df)-2)
rownames(df.values) <- df.object.ids
df.posX <- sapply(df[3:nrow(df),"Location_Center_X"], as.numeric)
df.posY <- sapply(df[3:nrow(df),"Location_Center_Y"], as.numeric)
df.CellID <- sapply(df[3:nrow(df),"ObjectNumber"], as.numeric)

# sometimes you miss markers in different experiments, so combining experiment dat
a from different MELC runs will create empty values inside the large data table.
# Imputing the data will fill the empty spaces...here we can use, e.g. missRanger
df.values.imputed <- missRanger(as.data.frame(df.values), verbose = 2)

#clipping values to be in same ratio, alternatively arcsinh transformation can be
used instead

for(sID in unique(df.sampleIDs)){
  df.sample <- df.values.imputed[df.sampleIDs == sID,]
  for(marker in colnames(df.sample)){

    q5 <- quantile(unlist(df.sample[[marker]]), probs = 0.05)
    q95 <- quantile(unlist(df.sample[[marker]]), probs = 0.95)

    df.sample[[marker]] <- scales::rescale(df.sample[[marker]],
                                           to = c(0, 1), from = c(q5,q95))
    df.sample[df.sample[[marker]] <= 0,marker] <- 0

    if(max(df.sample[[marker]]) > 0){
      df.sample[df.sample[[marker]] >= 1,marker] <- 1
    }
```

```
  }

  # here we need some back propagation or new df
  if(exists("df.sample.rescale")){
   df.sample.rescale <- rbind(df.sample.rescale,df.sample, make.row.names = FALSE)

  } else {
    df.sample.rescale <- df.sample
  }
}

# rotate the whole table and create a Seurat Object (SO)
# ...our MELC data are now interpreted as single cell seq data, so we can use all
functionality like there
df.values.T <- t(df.sample.rescale)
colnames(df.values.T) <- df.object.ids
SO = CreateSeuratObject(df.values.T)
SO$sampleID <- df.sampleIDs
SO$disease <- df.disease.state

SO.list <- SplitObject(SO, split.by = "sampleID")

for(i in 1:length(SO.list)){
  SO.list[[i]] <- FindVariableFeatures(SO.list[[i]],
                                       selection.method = "vst",
                                       nfeatures = 2000, verbose = FALSE)
}

anchors <- FindIntegrationAnchors(object.list = SO.list)
SO.integrated <- IntegrateData(anchorset = anchors)

# mean centering and stdDev scaling the expression values makes them comparable
SO.integrated <- FindVariableFeatures(SO.integrated, selection.method = "vst", mea
n.cutoff = c(0, 1), verbose = TRUE) # necessary function for dimRed
SO.integrated <- ScaleData(SO.integrated, do.center = TRUE, do.scale = TRUE, verbo
se=TRUE)
# create meta data for later grouping or splitting
Idents(SO.integrated) <- "disease"

# dimensionality reduction
SO.integrated <- RunPCA(SO.integrated, npcs = 20, verbose = FALSE) # perform PCA
SO.integrated <- RunUMAP(SO.integrated, reduction = "pca", dims = 1:20)
Idents(SO.integrated) <- SO.integrated$sampleID
UMAPPlot(SO.integrated)

# visualization of dimension reductions, saving plots
png(filename=paste(save.path, "UMAP_44PC_13dims.png"), width = 529, height = 435)
print(UMAPPlot(SO.integrated, label = FALSE))
dev.off()

df.markers <- rownames(df.values.T)

for (i in 1:length(df.markers)) {
    png(filename = paste(save.path, df.markers [i],
      "_featurePlot_44pc_16dims.png"), width = 350, height = 300)
    print(FeaturePlot(SO.integrated, features = df.markers [i], reduction =
      "umap", label = FALSE,  cols = c("yellow", "darkred")))
    dev.off()
}
```

```
# perform clustering
SO.integrated <- FindNeighbors(SO.integrated, dims = 1:16, reduction = "pca")
SO.integrated <- FindClusters(SO.integrated, resolution = c(0.18))
# plotting the clustered data
Idents(SO.integrated) <- SO.integrated@meta.data$RNA_snn_res.0.18
UMAPPlot(SO.integrated, label = TRUE)

# saving the clustered UMAP
png(filename = paste(save.path, "UMAP_clusters_res018a.png"),width=529,height=435)
print(UMAPPlot(SO.integrated, label = TRUE))
dev.off()
# saving data table
data_res018 <- df.values.imputed
data_res018$UMAP_1 <- SO@reductions$umap@cell.embeddings[, "UMAP_1"]
data_res018$UMAP_2 <- SO@reductions$umap@cell.embeddings[, "UMAP_2"]
data_res018$posX <- df.posX
data_res018$posY <- df.posY
data_res018$disease_state <- df.disease.state
data_res018$object_id <- df.object.ids
data_res018$cluster <- SO@meta.data$RNA_snn_res.0.18
data_res018$sourceID <- df.sampleIDs
data_res018$cellID <- df.CellID
write.table(data_res018, file = paste(save.path,
      "imputed_data_44PC_16dims_clustered_res_0.18.csv"),
       quote = FALSE, sep = ",", row.names = FALSE)
```

8.1.7. MELC Neibghborhood test and Randomization - Matlab

Main script reading CellProfiler output and perform neighborhood analysis

```
close all;
path='/Volumes/RALFSDATA/02_B220_ckit_kappa_prox_dist_measurement/';

%import measurements from CXCL12 somata and proximity within distance 0-
%30 pixels fromsomata edge

[ImageNumber_somata1,ObjectNumber_somata1,MeanIntensity_Bcells_prox,...
MeanIntensity_Bprogenitors_prox,MeanIntensity_CXCL12_prox,...
MeanIntensity_kappa_prox] = import_textfiles2([path,'CXCL12_proximity.txt']);

%import measurements from CXCL12 distal region at distance 31-100 pixels from
%somata edge

[ImageNumber_somata,ObjectNumber_somata,MeanIntensity_Bcells_dist,...
MeanIntensity_Bprogenitors_dist,MeanIntensity_CXCL12_dist,...
MeanIntensity_kappa_dist] = import_textfiles2([path,'CXCL12_distal.txt']);

num_images=max(ImageNumber_somata);

%plot raw intensity data
Mean_Intensities=[MeanIntensity_Bprogenitors_prox';...
MeanIntensity_Bprogenitors_dist';MeanIntensity_Bcells_prox';...
MeanIntensity_Bcells_dist';MeanIntensity_CXCL12_prox';...
MeanIntensity_CXCL12_dist';MeanIntensity_kappa_prox';MeanIntensity_kappa_dist'];

figure;
imagesc(Mean_Intensities);
title('raw data, mean intensities in CXCL12 somata surrounding','FontSize',20);
set(gca,'YTick',1:8);
set(gca,'YTickLabel',{'CXCL12 at 0-30 pix';'CXCL12 at 31-100';...
'BP at 0-30 pix';'BP at 31-100';'LpR at 0-30 pix';'LpR at 31-100';...
'VCam at 0-30 pix';'VCam at 31-100'},'FontSize',16);
xlabel('Cell ID');

diff_Bpro=MeanIntensity_Bprogenitors_prox-MeanIntensity_Bprogenitors_dist;
diff_Bcell=MeanIntensity_Bcells_prox-MeanIntensity_Bcells_dist;
diff_CXCL12=MeanIntensity_CXCL12_prox-MeanIntensity_CXCL12_dist;
diff_kappa=MeanIntensity_kappa_prox-MeanIntensity_kappa_dist;

diff_Bpro(diff_Bpro == 0) = NaN;
diff_Bcell(diff_Bcell == 0) = NaN;
diff_CXCL12(diff_CXCL12 == 0) = NaN;
diff_kappa(diff_kappa == 0) = NaN;

%all_diff=[diff_Bpro';diff_Bcell';diff_CXCL12';diff_kappa'];
%labeltext={'B Progenitor';'B cell';'CXCL12';'kappa'};
all_diff=[diff_Bpro';diff_Bcell';diff_kappa'];
labeltext={'B Progenitor';'B cell';'kappa'};
bins=-0.41:0.02:0.41;
figure; hold on;
m=nanmean(all_diff');
s=nanstd(all_diff');
subplot(615)
errorbar(m,s,'kx');
set(gca,'FontSize',16);
set(gca,'XTick',1:3);
set(gca,'XTickLabel',labeltext);
```

```
hold on;
set(gca,'FontSize',16); plot([0.5 3.5],[0 0],'k--'); ylim([-0.2 0.2]);
title('Mean and standard deviation, per stain.');

subplot(614);hold on;
set(gca,'FontSize',16);
[a,b]=hist(diff_Bpro,bins);
plot(b,a,'g-');
[a,b]=hist(diff_Bcell,bins);
plot(b,a,'r-');
%[a,b]=hist(diff_CXCL12,bins);
%plot(b,a,'c-');
[a,b]=hist(diff_kappa,bins);
plot(b,a,'b-');
title('all data pooled');
legend(labeltext,'FontSize',16);
axis([-0.31 0.31 0 130]);
ylabel('number of cells');
xlabel('distal (<0) vs proximal (>0)');
%plot intensity differences per image
for i=1:num_images

    current_data=find(ImageNumber_somata==i);
    aa=[-0.31 0.31 0 25];
    set(gca,'FontSize',16);

    %subplot(611); hold on;
    %[a,b]=hist(diff_CXCL12(current_data),bins);
    %title('CXCL12, per image');
    %plot(b,a,'c-');ylabel('number of cells');
    %axis(aa);xlabel('distal (<0) vs proximal (>0)');
    %set(gca,'FontSize',16);

    subplot(611); hold on;
    [a,b]=hist(diff_Bpro(current_data),bins);
    title('B Progenitor, per image');
    plot(b,a,'g-');ylabel('number of cells');
    axis(aa);xlabel('distal (<0) vs proximal (>0)');
    set(gca,'FontSize',16);

    subplot(612); hold on;
    [a,b]=hist(diff_Bcell(current_data),bins);
    title('B cell, per image');
    plot(b,a,'r-');ylabel('number of cells');
    axis(aa);xlabel('distal (<0) vs proximal (>0)');
    set(gca,'FontSize',16);

    subplot(613); hold on;
    [a,b]=hist(diff_kappa(current_data),bins);
    title('kappa, per image');
    plot(b,a,'b-');ylabel('number of cells');
    axis(aa);xlabel('distal (<0) vs proximal (>0)');
    set(gca,'FontSize',16);
end

%measure true and random colocalization per image pair
%pairs ordered CXCL12:BP, CXCL12:LpR, CXCL12:VCam, BP:LpR, BP:VCam, LpR:VCam
%and data ordered Observed white:Mean of random white:Std of random white,
%where white is proportion of colocalized image pixels. This means 3*6
```

```
%values. Also extracting measurements for a 'negative control' consisting
%of a pair where the first image comes from dataset 1, and the second image
%comes from dataset 2. this makes num_images*3*6=18 for rand_coloc and 18 for cont
rol_rand_coloc
rand_coloc=zeros(num_images,18);
for i=1:num_images
    if i==1%create controls
        CXCL12_1=imread(['CXCL12_cells00',num2str(1),'.tiff']);
        Bcells_1 = imread(['B_cells00',num2str(1),'.tiff']);
        Bpro_1=imread(['B_progenitor_cells00',num2str(1),'.tiff']);
        kappa_1=imread(['kappa_cells00',num2str(1),'.tiff']);
        Bcells_2 = imread(['B_cells00',num2str(2),'.tiff']);
        Bpro_2=imread(['B_progenitor_cells00',num2str(2),'.tiff']);
        kappa_2=imread(['kappa_cells00',num2str(2),'.tiff']);
        [ow1,mr1,sr1]=calc_white_pix_n_randomize(CXCL12_1,Bcells_2);
        [ow2,mr2,sr2]=calc_white_pix_n_randomize(CXCL12_1,Bpro_2);
        [ow3,mr3,sr3]=calc_white_pix_n_randomize(CXCL12_1,kappa_2);
        [ow4,mr4,sr4]=calc_white_pix_n_randomize(Bcells_1,Bpro_2);
        [ow5,mr5,sr5]=calc_white_pix_n_randomize(Bcells_1,kappa_2);
        [ow6,mr6,sr6]=calc_white_pix_n_randomize(Bpro_1,kappa_2);
        control_rand_coloc(i,:)=[ow1,mr1,sr1,ow2,mr2,sr2,ow3,mr3,sr3,ow4,mr4,sr4,o
w5,mr5,sr5,ow6,mr6,sr6];

    end

    if i<10
        CXCL12=imread(['CXCL12_cells00',num2str(i),'.tiff']);
        Bcells = imread(['B_cells00',num2str(i),'.tiff']);
        Bpro=imread(['B_progenitor_cells00',num2str(i),'.tiff']);
        kappa=imread(['kappa_cells00',num2str(i),'.tiff']);
    end
    if i>=10
        CXCL12=imread(['CXCL12_cells0',num2str(i),'.tiff']);
        Bcells = imread(['B_cells0',num2str(i),'.tiff']);
        Bpro=imread(['B_progenitor_cells0',num2str(i),'.tiff']);
        kappa=imread(['kappa_cells0',num2str(i),'.tiff']);
    end
    [ow1,mr1,sr1]=calc_white_pix_n_randomize(CXCL12,Bcells);
    [ow2,mr2,sr2]=calc_white_pix_n_randomize(CXCL12,Bpro);
    [ow3,mr3,sr3]=calc_white_pix_n_randomize(CXCL12,kappa);
    [ow4,mr4,sr4]=calc_white_pix_n_randomize(Bcells,Bpro);
    [ow5,mr5,sr5]=calc_white_pix_n_randomize(Bcells,kappa);
    [ow6,mr6,sr6]=calc_white_pix_n_randomize(Bpro,kappa);
    rand_coloc(i,:)=[ow1,mr1,sr1,ow2,mr2,sr2,ow3,mr3,sr3,ow4,mr4,sr4,ow5,mr5,sr5,o
w6,mr6,sr6];
end

figure;
aa=[-1 10 -0.05 0.2];
plot_labels={'CXCL12:B cells';'CXCL12:B progenitor cells';'CXCL12:kappa';'B cells:
B Progenitor cells';'B cells:kappa';'B progenitor cells:kappa'};
%we want a 'stair-case' distribution of plots:
plot_order=[331 334 337 335 338 339];
for p=1:6
    i=p+(p-1)*2;%to step cossectly in the results matrix
    subplot(plot_order(p));
    plot(rand_coloc(:,i),'o','markerSize',12);%CXCL12,BP true overlap
    hold on;
    errorbar(rand_coloc(:,i+1),rand_coloc(:,i+2),'k.','LineWidth',2);
```

```
    plot(0,control_rand_coloc(i),'o','markerSize',12);
    errorbar(0,control_rand_coloc(i+1),control_rand_coloc(i+2),'k.','LineWidth',2)
    set(gca,'XTick',0:2:13);
    set(gca,'XTickLabel',{'nc','2','4','6','8','10','12' },'FontSize',20);
      %negative bontrol
    title(plot_labels(p));xlabel('dataset');axis(aa);
    ylabel('colocalization'); ylim([-0.05 0.1]);
end
legend('observed colocalization',...
 'colocalization at randomization, with 2 standard deviations', 'negative control,
 measuring colocalization across datasets 1&2');
```

calc overlab function

```
function [pOL,p1,p2]=calc_overlap(I1,I2)
num_I1_pix=length(find(I1>0));
num_I2_pix=length(find(I2>0));
num_OL_pix=length(find(I1.*I2>0));

pOL=num_OL_pix/(num_I1_pix+num_I2_pix-num_OL_pix);
p1=(num_I1_pix-num_OL_pix)/(num_I1_pix+num_I2_pix-num_OL_pix);
p2=(num_I2_pix-num_OL_pix)/(num_I1_pix+num_I2_pix-num_OL_pix);
```

randomization test

```
function [ow,mr,two_times_sr]=calc_white_pix_n_randomize(I1,I2)
%NOTE; the program returns the standard deviation multiplied by 2!!!
I2orig=I2;
%here, 'randomization' is by flipping and rotating square images.
%keep I1 fixed, rotate I2, both images have to be square

%original, before rotating

[pOLl(1),p1l(1),p2l(1)]=calc_overlap(I1,I2);

%a first rotation
I2=rot90(I2);
[pOLl(2),p1l(2),p2l(2)]=calc_overlap(I1,I2);
%a second rotation
I2=rot90(I2);
[pOLl(3),p1l(3),p2l(3)]=calc_overlap(I1,I2);
%a third rotation
I2=rot90(I2);
[pOLl(4),p1l(4),p2l(4)]=calc_overlap(I1,I2);
%flip
I2=flip(I2orig);
[pOLl(5),p1l(5),p2l(5)]=calc_overlap(I1,I2);
%flip+a first rotation
I2=rot90(I2);
[pOLl(6),p1l(6),p2l(6)]=calc_overlap(I1,I2);
%flip+a second rotation
I2=rot90(I2);
[pOLl(7),p1l(7),p2l(7)]=calc_overlap(I1,I2);
%flip+a third rotation
I2=rot90(I2);
[pOLl(8),p1l(8),p2l(8)]=calc_overlap(I1,I2);

%observed white
ow=pOLl(1);
```

```
%mean white after randomizations
mr=mean(pOL1(2:8));
%standard deviation of white after randomizations
two_times_sr=2*std(pOL1(2:8));
```

import text files from CellProfiler output

```
function [ImageNumber,ObjectNumber,Intensity_MeanIntensity_B_cells_binary,...
Intensity_MeanIntensity_B_progenitors_binary,...
Intensity_MeanIntensity_CXCL12_somata_processes_binary,...
Intensity_MeanIntensity_kappa_cells_binary] =...
import_textfiles(filename, startRow, endRow)

%IMPORTFILE Import numeric data from a text file as column vectors.
%   [IMAGENUMBER,OBJECTNUMBER,INTENSITY_MEANINTENSITY_BP_CELLS_BINARY,
%   INTENSITY_MEANINTENSITY_CXCL12_SOMATA_PROCESSES_BINARY,
%   INTENSITY_MEANINTENSITY_LPR_CELLS_BINARY,
%   INTENSITY_MEANINTENSITY_VCAM_CELLS_BINARY]
%   = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
%   default selection.
%
%   [IMAGENUMBER,OBJECTNUMBER,INTENSITY_MEANINTENSITY_BP_CELLS_BINARY,
%   INTENSITY_MEANINTENSITY_CXCL12_SOMATA_PROCESSES_BINARY,
%   INTENSITY_MEANINTENSITY_LPR_CELLS_BINARY,
%   INTENSITY_MEANINTENSITY_VCAM_CELLS_BINARY]
%   = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows STARTROW
%   through ENDROW of text file FILENAME.
%
% Example:
%   [ImageNumber,ObjectNumber,Intensity_MeanIntensity_BP_cells_binary,...
%   Intensity_MeanIntensity_CXCL12_somata_processes_binary,...
%   Intensity_MeanIntensity_LpR_cells_binary,...
%   Intensity_MeanIntensity_VCam_cells_binary] = ...
%   importfile('CXCL12_proximity.txt',1, 1411);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/04/30 16:04:02

%% Initialize variables.
delimiter = '\t';
if nargin<=2
    startRow = 2;
    endRow = inf;
end

%% Read columns of data as strings:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%s%s%s%s%s%s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
```

```
dataArray = textscan(fileID, formatSpec, endRow(1)-
startRow(1)+1, 'Delimiter', delimiter, 'HeaderLines', startRow(1)-
1, 'ReturnOnError', false);

for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-...
startRow(block)+1, 'Delimiter', delimiter, 'HeaderLines', startRow(block)-...
1, 'ReturnOnError', false);

    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric strings to numbers.
% Replace non-numeric strings with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);

for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = dataArray{col};
end

numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4,5,6]
    % Converts strings in the input cell array to numbers. Replaced non-numeric
    % strings with NaN.
    rawData = dataArray{col};

    for row=1:size(rawData, 1);
      % Create a regular expression to detect and remove
       % non-numeric prefixes and suffixes.
        regexstr = '(?<prefix>.*?)(?<numbers>...
([-]*(\d+[\,]*)+[\.]{0,1}\d*[eEdD]{0,1}[-+]*\d*[i]{0,1})|...
([-]*(\d+[\,]*)*[\.]{1,1}\d+[eEdD]{0,1}[-+]*\d*[i]{0,1}))(?<suffix>.*)';
        try
            result = regexp(rawData{row}, regexstr, 'names');
            numbers = result.numbers;

            % Detected commas in non-thousand locations.
            invalidThousandsSeparator = false;
            if any(numbers==',');
                thousandsRegExp = '^\d+?(\,\d{3})*\.{0,1}\d*$';
                if isempty(regexp(numbers, thousandsRegExp, 'once'));
                    numbers = NaN;
                    invalidThousandsSeparator = true;
                end
            end
            % Convert numeric strings to numbers.
            if ~invalidThousandsSeparator;
                numbers = textscan(strrep(numbers, ',', ''), '%f');
                numericData(row, col) = numbers{1};
                raw{row, col} = numbers{1};
            end
        catch me
```

```
        end
    end
end


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw);
% Find non-numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

%% Allocate imported array to column variable names
ImageNumber = cell2mat(raw(:, 1));
ObjectNumber = cell2mat(raw(:, 2));
Intensity_MeanIntensity_B_cells_binary = cell2mat(raw(:, 3));
Intensity_MeanIntensity_B_progenitors_binary = cell2mat(raw(:, 4));
Intensity_MeanIntensity_CXCL12_somata_processes_binary = cell2mat(raw(:, 5));
Intensity_MeanIntensity_kappa_cells_binary = cell2mat(raw(:, 6));
```

8.1.8. MELC Neighborhood Analysis – R

```
# perform neighborhood analysis on MELC data

library(data.table)
library(ggplot2)
library(dplyr)

# load the data and extract meta data
path_to_file <- choose.dir()
input_data <- choose.files()

print("...read file")
#df_input <- fread(paste0(path_to_file,input_data), encoding="UTF-8")
df_input <- fread(input_data, encoding = "UTF-8")
# if data has some weird stuff in the first rows, delete !!!
num_of_rows <- nrow(df_input)
df_input <- df_input[seq(1,num_of_rows),]
marker.list <- colnames(df_input)[1:43]

#combining the different FOVs by Patient IDs
patientID_long <- df_input$sourceID
#inspect different patient IDs and build list for renaming
fovs <- unique(df_input$sourceID) # we also need fovs later in other calculations

#creating short names containing only patient ID
patientID <- fread(input_meta_data, encoding = "UTF-8")


#renaming Patient IDs
for(st in patientID) {
 df_input$sourceID[grepl(st,df_input$sourceID)] <- st
}

#updating fovs
fovs <- unique(df_input$sourceID)

#define which seed cells create the niches and where to put them in table
inspection_object <- "neutrophils"
inspection_column <- "Clusters"
print(paste("...set inspection object to", inspection_object))
#defining the radius of neighborhood
radius.px <- 32

#looking for neighbors, create extra columns for each cluster (n1, n2, n3, ....)
# take a cell, calc distance to neighbor, except target cell
# add the colum of desired neighbor by 1 if neighbor is found
# caution...go dataset vice through ....faster
# adding neighbor columns to big data frame
# (nX) := sum of neighboring cell(s) which is of cluster X
clusters.list <- unique(df_input[[inspection_column]])
colnames.neighbors <- paste0("n",clusters.list)
df_input[,colnames.neighbors] <- 0

df_with_neighbors <- data.frame()
neighbors.table <- data.frame()
df_summary <- data.frame(fovs)
colnames(df_summary) <- "FOV"
df_summary[,colnames.neighbors] <- 0
df_diseaseStates <- data.frame()
```

```
print(paste("start to look for neighbors in radius of", radius.px, "px"))

# calculating distances vectorized, monitoring via progressbar
pb <- txtProgressBar(min = 1, max = length(fovs), style = 3)


for(d in 1:length(fovs)){

  setTxtProgressBar(pb, value = d)
  a.dataset <- df_input[df_input$sourceID %in% fovs[d],]
  df_diseaseStates <- rbind(df_diseaseStates, unique(a.dataset$disease_state))

  for(i in 1:nrow(a.dataset)) {

    center.point.cluster <- a.dataset[[i,inspection_column]]

    # this if statement makes sure, only the desired cluster of cell is counted
    # to analyse all the cells and their possible neighbor clusters ommit the if
    # statement

    if(center.point.cluster != inspection_object){
      next
    }

    x1 <- as.numeric(a.dataset$posX[i])
    x2 <- as.numeric(a.dataset$posX)
    y1 <- as.numeric(a.dataset$posY[i])
    y2 <- as.numeric(a.dataset$posY)

    distances <- ((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1))

    neighbors <- a.dataset[distances <= (radius.px*radius.px),]

    for(j in 1:nrow(neighbors)){

      neighbor.cluster <- neighbors[[j,inspection_column]]
      a.dataset[[i,paste0("n",neighbor.cluster)]] <-
       a.dataset[[i,paste0("n",neighbor.cluster)]]+1
    }
    # remove seed cell count
    a.dataset[[i,paste0("n",center.point.cluster)]] <-
       a.dataset[[i,paste0("n",center.point.cluster)]]-1
    neighbors.table <- rbind(neighbors.table, neighbors)
    }

  df_with_neighbors <- rbind(df_with_neighbors,a.dataset)

  #creating summary --> counting the cell's clusters in their niche in each FOV

  for(k in colnames(df_summary)[seq(2,length(colnames(df_summary)))]) {

    cells.with.neighbors <- a.dataset[[k]]
    df_summary[[d,k]] <- length(cells.with.neighbors[cells.with.neighbors > 0])

  }
}

colnames(df_diseaseStates) <- "diseaseStates"
```

```
colnames(df_summary) <- c("FOV",clusters.list)
close(pb)
print("...just saving the neighborhood analysis output table")

write.table(df_summary, file = paste0(path_to_file,"neighbor_analysis_of_",
inspection_object, "_in_", radius.px, "_px_", input_data),
            quote = FALSE, sep = ",", row.names = FALSE)

#-------------------------------------------------------------------------------
-------------------------------------------------
#calculating the distribution of every niche in % or absulute numbers

df_freq <- subset(df_with_neighbors,
       df_with_neighbors[[inspection_column]] == inspection_object,
       select = colnames.neighbors)
df_freq_row_sum <- rowSums(df_freq)

#ignoring zero niche counts
df_freq <- df_freq[df_freq_row_sum != 0,]

df_sourc_diseaseSt_subset <- subset(df_with_neighbors,  df_with_neighbors[[inspect
ion_column]] == inspection_object, select = c("sourceID", "disease_state", "object
_id"))
df_sourc_diseaseSt_subset <- df_sourc_diseaseSt_subset[df_freq_row_sum != 0,]
df_freq <- cbind(df_sourc_diseaseSt_subset,df_freq)
colnames(df_freq) <- c("SourceID","diseaseState", "object_id", clusters.list)

write.table(df_freq, file = paste0(path_to_file,
       "single_niche_abs_neighbor_count_of_", inspection_object, "_in_", radius.px,
       "_from_", input_data), quote = FALSE, sep = ",", row.names = FALSE)

df_freq_avg <- data.frame(fovs,df_diseaseStates)
df_freq_avg[,clusters.list] <- 0

for(i in 1:length(fovs)){
  freqs_fov <- df_freq[df_freq$SourceID %in% fovs[i]]
  freq_means <- colMeans(subset(freqs_fov, select = clusters.list))

  for(j in 1:length(freq_means)){
    df_freq_avg[[i,clusters.list[j]]] <- freq_means[j]
  }

}

# writung mean frequencies of neighbors to table
colnames(df_freq_avg) <- c("SampleID", "diseaseState",clusters.list)

write.table(df_freq_avg, file = paste0(path_to_file,"niche_frequency_of_",
       inspection_object, "_in_", radius.px,
       "_from_", input_data), quote = FALSE, sep = ",", row.names = FALSE)

#-------------------------------------------------------------------------------
# calculate the mean fluorescence intensity per niche and per cell cluster
#-------------------------------------------------------------------------------
# what we would like to show is, e.g. the frequency of T cells dependent on diseas
e state from
# endothelia niches and grouped by FOV/PatientID

for (i in 3:length(colnames(df_freq_avg))) {
```

```
  print(paste("...saving plot ", "distribution_plot_of_", gsub("[[:punct:]]",
      " ", colnames(df_freq_avg)[i]), "_in_",  inspection_object,
      "_niche_within_", radius.px, "px.png"))

  df_test <- data.frame(df_freq_avg[[1]], df_freq_avg[[2]], df_freq_avg[[i]])
  colnames(df_test) <- c("SampleID", "diseaseState", colnames(df_freq_avg)[i])
  df_test$diseaseState <- factor(df_test$diseaseState, levels =
      c("control", "acute", "chronic", "prolonged"))

  colname.df_test <- colnames(df_freq_avg)[i]

  df_dotplot <- data.frame(df_freq[[1]], df_freq[[2]],df_freq[[i]])
  colnames(df_dotplot) <- c("SampleID", "diseaseState", colnames(df_freq)[i])
  df_dotplot$diseaseState <- factor(df_dotplot$diseaseState, levels = c("control",
 "acute", "chronic", "prolonged"))


  # violin plot with dots
  # dot plot calculation at different place...other loop because of df_freq ...
  # deleted , dotsize = 0.8, binwidth = 2
  data_summary <- function(x) {
    m <- mean(x)
    ymin <- m-sd(x)
    ymax <- m+sd(x)
    return(c(y=m,ymin=ymin,ymax=ymax))
  }

  # trim argument in geom_violin allows or not allows the distribution from
  # violin to be cut
  # creating the violin plot
  v <- ggplot(df_dotplot, aes(x=diseaseState, fill=factor(SampleID),
      y=df_dotplot[[3]])) + geom_violin(trim = FALSE)
  v <- v + stat_summary(fun.data = data_summary) + theme_minimal()

  # adding the labels inside the plot
  tit <- paste0("distribution plot of ", gsub("[[:punct:]]", "",
              colnames(df_freq_avg)[i]), " in ",  inspection_object,
              " niche within ", radius.px, "px")
  xtit <- "Disease State"
  ytit <- paste0("rel. Frequency of ", colname.df_test)

  v <- v + labs(title = tit , x = xtit, y = ytit)


  png(filename = paste0(path_to_file, "distribution dotPlot of ",
      gsub("[[:punct:]]", "", colnames(df_freq_avg)[i]), " in ",
      inspection_object," niche within ", radius.px, "px.png"),
      width = 10, height = 5, res = 144, units = "in")

  print(v)
  dev.off()

}

print("...completed")

#-------------------------------------------------------------------------------
# trial area to calculate the mean fluorescence intensity per niche and per
```

```
# cell cluster to have the overall average

unique_objID <- duplicated(neighbors.table$object_id)
all_niche_cells_wo_0n <- neighbors.table[!unique_objID,]

MFI.table <- as.data.frame(marker.list)
colnames(MFI.table) <- "marker"
MFI.table[["niche"]] <- rep(inspection_object,length(marker.list))
MFI.table[["disease phase"]] <- rep("",length(marker.list))
MFI.table[["sourceID"]] <- rep("",length(marker.list))
MFI.table[,clusters.list] <- 0
df_for_selection <- data.frame(patientID,df_diseaseStates)
MFI.table.final <- data.frame()
for(i in 1:dim(df_for_selection)[1]){

    MFI.table[["disease phase"]] <- rep(df_for_selection[i,2], length(marker.list)
)
    MFI.table[["sourceID"]] <- rep(df_for_selection[i,1], length(marker.list))

    ds_subset <- as.data.frame(subset(all_niche_cells_wo_0n,
             all_niche_cells_wo_0n$disease_state == df_for_selection[i,2] &
             all_niche_cells_wo_0n$sourceID == df_for_selection[i,1]))

      for(cl in clusters.list) {

        dsc_subset <- as.data.frame(subset(ds_subset, ds_subset$Clusters == cl))
        marker.means <- colMeans(dsc_subset[,marker.list])
        MFI.table[[cl]] <- marker.means
      }

    MFI.table.final <- rbind(MFI.table.final,MFI.table)

}

MFI.table.final[is.na(MFI.table.final)] <- ""

write.table(MFI.table.final, file = paste0(path_to_file,"niche_means_of_",
       inspection_object, "_in_", radius.px,
       "_from_", input_data), quote = FALSE, sep = ",", row.names = FALSE)
```

8.1.9. ST data integration and clustering analysis script – R

```
# ST data integration script
# loading necessary packages/libraries
library(Seurat)
library(SeuratData)
library(ggplot2)
library(patchwork)
library(dplyr)
library(future)

# set environment parameters to reduce calculation time
fgmS <- 6000*1024^2
plan("multiprocess", workers = 2)
options(future.globals.maxSize = fgmS)
plan()

# define data and results location
maindir <- choose.dir()
savedir <- maindir+"/analysis_results"

# some meta data to each sample
meta_data <- read.csv(file=choose.files(),sep = ";")

experiment <- meta_data$experiment
sample_ID <- meta_data$sample_ID

# filter values from meta data, requires data pre-observation
nFeaturef <- meta_data$nfeature_filter_value
percent.mtf <- meta_data$percent_mtf_filter_value

variable.features.list <- c()
barcode.list <- c()

#load ST data and create Seurat Object

for (i in 1:length(experiment)) {

  ddf <- paste0(maindir,"/single_outs/", experiment[i], "/outs")
  SO <- Load10X_Spatial(data.dir = ddf, filename = "filtered_feature_bc_matrix.h5"
)
  SO[["percent.mt"]] <- PercentageFeatureSet(SO, pattern = "^MT-")
  SO <- subset(SO, subset = nFeature_Spatial > nFeaturef[i] &
             percent.mt < percent.mtf[i] & nCount_Spatial > 1)

  SO@meta.data$experiment = rep(experiment[i],length(colnames(x = SO)))
  SO@meta.data$sample_ID = rep(sample_ID[i],length(colnames(x = SO)))
  barcode.list <- c(barcode.list,
             gsub("-1", paste0("-",i), rownames(SO@meta.data)))
  SO@meta.data$barcodes <- gsub("-1", paste0("-",i), rownames(SO@meta.data))
  if(exists("SO_all")){
    SO_all <- merge(SO_all, SO)
  } else {
    SO_all <- SO
  }
}

# first check to see if data is loaded correctly
Idents(SO_all) <- SO_all@meta.data$sample_ID
```

```
VlnPlot(SO_all, features = c("nFeature_Spatial", "nCount_Spatial", "percent.mt"),
ncol = 3)

plot1 <- FeatureScatter(SO_all, feature1 = "nCount_Spatial",
             feature2 = "percent.mt")
plot2 <- FeatureScatter(SO_all, feature1 = "nCount_Spatial",
             feature2 = "nFeature_Spatial")
plot1 + plot2


# data integration based on SCT transform
SubsetSO.list <- SplitObject(SO_all, split.by = "sample_ID")

for(i in 1:length(SubsetSO.list)){
  SubsetSO.list[[i]] <- SCTransform(SubsetSO.list[[i]],
             assay = "Spatial", verbose = TRUE)
  DefaultAssay(SubsetSO.list[[i]]) <- "SCT"
}

SO.features <- SelectIntegrationFeatures(object.list = SubsetSO.list,
       nfeatures = 3000)
SubsetSO.list <- PrepSCTIntegration(object.list = SubsetSO.list,
       anchor.features = SO.features, verbose = TRUE)
SO.anchors <- FindIntegrationAnchors(object.list = SubsetSO.list,
       normalization.method = "SCT", anchor.features = SO.features, verbose = TRUE)
SO.integrated <- IntegrateData(anchorset = SO.anchors,
       normalization.method = "SCT", verbose = TRUE)

# dimension reduction and clustering based on different resolutions
SO.integrated <- RunPCA(SO.integrated, verbose = FALSE)
SO.integrated <- FindNeighbors(SO.integrated, dims = 1:30)
res <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)
SO.integrated <- FindClusters(SO.integrated, verbose = FALSE, resolution = res)
SO.integrated <- RunUMAP(SO.integrated, dims = 1:30)
SO.integrated <- RunTSNE(SO.integrated, reduction = "pca", dims = 1:30)

# data visualization to check parameters, if necessary, rerun previous section wit
h different parameters
DimPlot(SO.integrated, reduction = "umap", group.by = c("ident", "sample_ID"))
DimPlot(SO.integrated, reduction = "tsne", group.by = c("ident", "sample_ID"))

# exporting clustered data

Out = as.data.frame(cbind(Barcode = SO.integrated@meta.data$barcodes,
                     SO.integrated@meta.data$integrated_snn_res.0.1,
                     SO.integrated@meta.data$integrated_snn_res.0.2,
                     SO.integrated@meta.data$integrated_snn_res.0.3,
                     SO.integrated@meta.data$integrated_snn_res.0.4,
                     SO.integrated@meta.data$integrated_snn_res.0.5,
                     SO.integrated@meta.data$integrated_snn_res.0.6,
                     SO.integrated@meta.data$integrated_snn_res.0.7,
                     SO.integrated@meta.data$integrated_snn_res.0.8))

colnames(Out) <- c("Barcode",paste0("ClusterRes",res))
write.table(Out,file = paste0(savedir,"/clustered_cells_allSamples1234.csv"),
             quote = F, sep = ",", row.names = F)

# dimensionality reductions export
umap.table <- cbind(SO.integrated@meta.data$barcodes,
```

```
              SO.integrated@reductions$umap@cell.embeddings[,1],
              SO.integrated@reductions$umap@cell.embeddings[,2])
colnames(umap.table) <- c("Barcode", "UMAP_1", "UMAP_2")
write.table(umap.table, file=paste0(savedir, "/umap_allSamples1234.csv"),
                                      quote = F, sep = ",", row.names = F)

tsne.table <- cbind(SO.integrated@meta.data$barcodes,
      SO.integrated@reductions$tsne@cell.embeddings[,1],
      SO.integrated@reductions$tsne@cell.embeddings[,2])

colnames(tsne.table) <- c("Barcode", "tSNE_1", "tSNE_2")
write.table(tsne.table, file=paste0(savedir,"/tsne_allSamples1234.csv"),
                           quote = F, sep = ",", row.names = F)
```

8.1.10. ST GSEA script – R

```
# perform group based Gene Set Enrichment Analysis (GSEA)
library(Seurat)
library(dplyr)
library(tidyverse)
library(presto)
library(msigdbr)
library(fgsea)
library(ggplot2)

#loading the Seurat Object / here we load the SO of the integrated data set contai
ning all rounds of ST experiment
SO <- readRDS(choose.files())
gene.sets.list <- read.csv(choose.files())
save.path <- choose.dir()
dir.create(save.path)

#add meta data column with disease phase
experiments <- unique(SO$experiment)
disease.phase <- c("chronic","chronic","prolonged","acute","control","control",
                   "chronic","control","prolonged","prolonged","acute","acute")
df.ed <- cbind.data.frame(experiments,disease.phase)
SO$disease_phase <- rep("", length(SO$experiment))

for(i in 1:length(df.ed$experiments)){
  SO$disease_phase[SO$experiment==df.ed$experiments[i]] <- df.ed$disease.phase[i]
}

#perform a fast Wilcoxon rank sum test
ST.genes <- wilcoxauc(SO,
              "disease_phase", assay = "scale.data", seurat_assay = "integrated")
head(ST.genes)

#here we see all the gene count for every group of experiment
dplyr::count(ST.genes, group)

# overview of all Molecular Signatures Database collections
msigdbr_species()

stepC <- 1
steps <- length(unique(gene.sets.list$lib))*length(unique(SO$disease_phase))*
             length(gene.sets.list$pathway)
fgseaRes_all <- data.frame()

for(l in unique(gene.sets.list$lib)){
  # we focus on human species
  m_df<- msigdbr(species = "Homo sapiens", category = l)
  fgsea_sets <- m_df %>% split(x = .$gene_symbol, f = .$gs_name)
  gs <- filter(gene.sets.list, species == "Homo sapiens", lib == l)[["pathway"]]
  for(dp in unique(SO$disease_phase)) {
      # select only the feature and auc columns for fgsea,
      # which statistics to use idepends on user specifications
    disease.genes <- ST.genes %>%
      dplyr::filter(group == dp) %>%
      #dplyr::filter(logFC >= 0.25) %>%
      dplyr::filter(padj < 0.05) %>%
      arrange(logFC) %>%
      #arrange(padj) %>%
      dplyr::select(feature, avgExpr)
```

```
    ranks<- deframe(disease.genes)
    #head(ranks)

    # run GSEA and export results
    fgseaRes<- fgseaMultilevel(fgsea_sets, stats = ranks,
                    nPermSimple = 10000, minSize = 1)
    fgseaRes$leadingEdge <- apply(fgseaRes[,"leadingEdge", drop=F],
                    1, function(x) paste(unlist(x), sep = ", ", collapse = ", "))
    fgseaRes$leadingEdge <- gsub(", ", ";", fgseaRes$leadingEdge)
    write.table(fgseaRes, file = paste0(save.path,"/fgseaResults_", l, "_",
                    dp,".csv"), quote = FALSE, sep = ",", row.names = FALSE)
    fgseaRes$diseasePhase <- rep(dp,dim(fgseaRes)[1])
    fgseaRes_all <- rbind(fgseaRes_all,fgseaRes)
  }

}

# dotplot creation with Pathway Selection
pathway.selection <- read.csv(choose.files())

fgseaRes_PSelection <- fgseaRes_all[fgseaRes_all$pathway %in%
                          pathway.selection$pathway,]
pathway.list <- annas.selection$pathway[c(2,3,4,5,15,24,37,1,6,8)]
fgseaRes_PSelection2 <- fgseaRes_PSelection[fgseaRes_PSelection$pathway %in%
                          pathway.list,]

fgseaRes_PSelection2$diseasePhase <- factor(fgseaRes_PSelection2$diseasePhase,
                          levels = c("control", "acute", "chronic","prolonged"))

#manual dotplot creation, exporting via R Studio
dplot <- ggplot(fgseaRes_AnnasSelection2, aes(x=diseasePhase, y=pathway,
      size=size,color=NES)) + geom_point(alpha=1, stroke = 2) +
      theme_classic() + theme(text = element_text(size=16))
dplot = dplot+
      scale_color_gradient(low = "grey90",  high = "darkblue", space = "Lab")
dplot
```

8.1.11. ST ssGSEA script – R

```
# perform single sample Gene Set Enrichment Analysis (ssGSEA)

library(escape)
library(dittoSeq)
library(SingleCellExperiment)
library(Seurat)
library(SeuratObject)
library(dplyr)
library(ggplot2)

#loading the data and gene sets we would like to investigate
results.path <- choose.dir()
gene.sets.list <- read.csv(choose.files()) # list of gene sets "*.csv"
SO.spatial <- readRDS(choose.files()) # saved integrated Seurat object "*.rds"

names(SO.spatial@images) <- paste0("slide_",unique(SO.spatial$experiment))

#add meta data column with disease phase
experiments <- unique(SO.spatial$experiment)
disease.phase <- c("chronic","chronic","prolonged","acute","control","control",
                   "chronic","control","prolonged","prolonged","acute","acute")
df.ed <- cbind.data.frame(experiments,disease.phase)
SO.spatial$disease_phase <- rep("", length(SO.spatial$experiment))

for(i in 1:length(df.ed$experiments)){
  SO.spatial$disease_phase[SO.spatial$experiment == df.ed$experiments[i]] <-
       df.ed$disease.phase[i]
}

#visualizing only one image in e.g. SpatialFeaturePlot to check right loading
gene_feature <- "CCL18" # gene to check on spatial plot
image_title <- "slide_" # add desired image title based on naming
SpatialFeaturePlot(SO.spatial, features = gene_feature, images = image_title)

#loop through the gene sets and perform ssGSEA on each spot
start_time <- Sys.time()

for(s in unique(gene.sets.list$species)){

  for(l in unique(gene.sets.list$lib)){

    gs <- filter(gene.sets.list, species == s, lib == l)[["pathway"]]
    #print(gs)
    GS.selected <- getGeneSets(species = s,
                               library = l,
                               gene.sets = gs)
    ES.selected <- enrichIt(obj = GetAssayData(SO.spatial, slot = "scale.data"),
                    gene.sets = GS.selected, groups = 1000, cores = 20)
    # adding the enrichment scores to Seurat Object
    for(i in colnames(ES.selected)){
      SO.spatial[[i]] <- ES.selected[[i]]
    }
  }
}

end_time <- Sys.time()
measured.time <- end_time - start_time
measured.time
```

```
# now we have all our enrichment scores, we should plot them as a feature
# plot to the single images, based on their pathway we have to make sure, that
# each image in one pathway gets the same color scale in a fixed range

steps <- length(names(SO.spatial@images))*length(gene.sets.list$pathway)
stepC <- 1

for(p in gene.sets.list$pathway){

  if(!(p %in% names(SO.spatial@meta.data))){
    next
  }

  save.path <- paste0(results.path,"/",p)
  dir.create(save.path)
  q5 <- quantile(unlist(SO.spatial[[p]]), probs = 0.05)
  q95 <- quantile(unlist(SO.spatial[[p]]), probs = 0.95)

  for(img in names(SO.spatial@images)){

    print(paste("saving image ", stepC, "of", steps))
    sfp.name <- paste0(save.path, "/enrichment_fplot_", p, "_", img,  ".svg")
    svg(sfp.name)
    print(SpatialFeaturePlot(SO.spatial, features = p,
       images = img, min.cutoff = q5, max.cutoff = q95,
       image.alpha = 0, pt.size.factor = 2)+
       ggplot2::scale_fill_gradient2(low = "white",
       mid = "lightblue",high = "darkblue"))
    dev.off()
    stepC <- stepC+1

  }
}

# saving the Seurat Object with single pathway related enrichment scores
saveRDS(SO.spatial, file =
   paste0(results.path,"/SO_spatial_1234_gsea_escape_filtered.rds"))
```

8.1.12. LSFM destriping script – Python

```
import os
import sys
import tkinter as tk
from tkinter import filedialog
import numpy as np
from scipy import ndimage
from scipy.optimize import curve_fit
from skimage import io
from skimage import img_as_float
from skimage import draw
import pylab


def get_min_lenght_of_list_elements(values_list):

    '''
    get minimum length of all lists inside list

    Parameters
    ----------
    values_list : list of lists

    Returns
    -------
    minimum length

    '''

    min_len = len(values_list[0])

    for i in values_list:

        if (len(i)<min_len):

            min_len = len(i)

    return min_len



def getEdgePoints(img_width, img_height, theta):

    '''
    calculating the endpoints (2D) of a line, starting from image center
    to image border

    Parameters
    ----------
    img_width : horizontal size of a 2D array
    img_heigth : vertical size of a 2D array
    theta : angle of line in degree, where 0° is at img_width,img_height//2...
            rotating angles anticlockwise
    Returns
    -------
    y, x coordinate of image border points

    '''
```

```
    x0 = img_width//2
    y0 = img_height//2

    x_edge = -1
    y_edge = -1

    angle_ratio = img_height/img_width

    if ((theta >= 0) and (theta <= np.degrees(np.arctan(angle_ratio)))):

        x_edge = img_width-1
        y_edge = y0-x0*np.tan(np.radians(theta))

    if ((theta > np.degrees(np.arctan(angle_ratio))) and
       (theta <= 180-(np.degrees(np.arctan(angle_ratio))))):

        t = 90-theta
        x_edge = x0+y0*np.tan(np.radians(t))
        y_edge = 0

    if ((theta > 180-(np.degrees(np.arctan(angle_ratio)))) and
       (theta <= np.degrees(np.arctan(angle_ratio))+180)):

        t = 180-theta
        x_edge = 0
        y_edge = y0-x0*np.tan(np.radians(t))

    if ((theta > np.degrees(np.arctan(angle_ratio))+180) and
       (theta <= 360-(np.degrees(np.arctan(angle_ratio))))):

        t = 270-theta
        x_edge = x0-y0*np.tan(np.radians(t))
        y_edge = img_height-1

    if ((theta > 360-(np.degrees(np.arctan(angle_ratio)))) and
       (theta <= 360)):

        t = 360-theta
        x_edge = img_width-1
        y_edge = y0+x0*np.tan(np.radians(t))


    return y_edge, x_edge
def angular_linePlot(image, angle, getValues = False, showPlot = True):

    '''
    create line selection starting and ending from image borders at a given angle

    Parameters
    ----------
    image : 2D array, created image, so working with raw values
    angle: angle of line in degree, where 0° is at img_width,img_height//2
    ...rotating angles anticlockwise
    getValues : optional, returns values of line selection measurement if True
    showPlot : optional, visualize plot on image and diagram if True

    Returns
    -------
    values from line plot if getValues=True
```

```
    '''
    h,w = image.shape
    #y0 = h//2
    #x0 = w//2
    y0, x0 = getEdgePoints(w, h, angle+180)
    y1, x1 = getEdgePoints(w, h, angle)

    lin = draw.line_nd((y0,x0),(y1,x1), endpoint=False)
    values = image[lin]

    if(showPlot):

        y, x = np.linspace(y0,y1,len(values),endpoint=False),
             np.linspace(x0,x1,len(values),endpoint=False)
        p_min = np.percentile(image,0.1)
        p_max = np.percentile(image,99.9)

        fig, axes = pylab.subplots(ncols=2)
        axes[0].imshow(image, cmap="gray", vmin=p_min, vmax=p_max)
        axes[0].plot(x,y,'r-')
        axes[0].axis('image')

        axes[1].plot(values)

        pylab.show()

    if(getValues):

        return values


def fft_angular_linePlot(fft_image, angle, getValues = False, showPlot = True):

    '''
    create line selection starting and ending from image borders at a given angle

    Parameters
    ----------
    fft_image : 2D array, created for fft image, so values are scaled by 20*log
    angle: angle of line in degree, where 0° is at img_width,img_height//2...
    rotating angles anticlockwise
    getValues : optional, returns values of line selection measurement if True
    showPlot : optional, visualize plot on image and diagram if True

    Returns
    -------
    values from line plot if getValues=True

    '''
    h,w = fft_image.shape
    #y0 = h//2
    #x0 = w//2
    y0, x0 = getEdgePoints(w, h, angle+180)
    y1, x1 = getEdgePoints(w, h, angle)

    lin = draw.line_nd((y0,x0),(y1,x1), endpoint=False)
    values = fft_image[lin]
```

```
    if(showPlot):

        y, x = np.linspace(y0,y1,len(values),endpoint=False),
             np.linspace(x0,x1,len(values),endpoint=False)
        p_min = np.percentile(20*np.log(np.abs(fft_image)+1),0.1)
        p_max = np.percentile(20*np.log(np.abs(fft_image)+1),99.9)

        fig, axes = pylab.subplots(ncols=2)
        axes[0].imshow(20*np.log(np.abs(fft_image)+1), cmap="gray",
             vmin=p_min, vmax=p_max)
        axes[0].plot(x,y,'r-')
        axes[0].axis('image')

        axes[1].plot(20*np.log(np.abs(values)+1))

        pylab.show()

    if(getValues):

        return values


def set_fft_angular_linePlot_values(fft_image, angle, values):

    '''
    set values of a line selection within fft image at a given anggle
    image do not have to be complex...other types supported as well

    Parameters
    ----------
    fft_image : 2D array, created for fft image, so values are scaled by 20*log
    angle: angle of line in degree, where 0° is at img_width,img_height//2...
    rotating angles anticlockwise
    values : values to set at single pixel line selection

    '''
    h,w = fft_image.shape
    #y0 = h//2
    #x0 = w//2
    y0, x0 = getEdgePoints(w, h, angle+180)
    y1, x1 = getEdgePoints(w, h, angle)

    lin = draw.line_nd((y0,x0),(y1,x1), endpoint=False)
    fft_image[lin] = values

    return fft_image

def rescale_freqlist_to_same_shape(freq_list):

    '''

    bring single angle measurements to one rectangular array by interpolation
    of different lengths of measurements

    Parameters
    ----------
    freq_list : list of single angle measurements

    Returns
```

```
    -------
    rectangular shaped array with interpolated measurements

    '''

    list_len = len(freq_list)
    max_len = 0

    for i in freq_list:
        length = len(i)

        if length>max_len:
            max_len = length

    freq_img = np.zeros((list_len,max_len))

    for j in range(list_len):

        freq_img[j,:] = np.interp(np.linspace(0,len(freq_list[j])-1,
             num=max_len),np.arange(len(freq_list[j])),freq_list[j])

    return freq_img


def get_doG_bandpass(image_shape, sd1, sd2, normalize = True):

    '''
    create band pass filter mask by difference of two gaussians

    Parameters
    ----------
    image_shape : size of image mask as tuple
    sd1 : standard deviation of 1st gaussian
    sd2 : standard deviation of 2nd gaussian
    normalize : True/False for normalization by devision of mask's max value

    Returns
    -------
    band pass filter mask image

    '''

    iy, ix = image_shape
    cy, cx = iy//2, ix//2
    x = np.linspace(0, ix, ix)
    y = np.linspace(0, iy, iy)
    X, Y = np.meshgrid(x, y)
    if (sd1 == 0):

        g_mask_sd2 = np.exp(-((X-cx)**2+(Y-cy)**2)/(2*sd2**2))
        g_mask_sd2 = g_mask_sd2-g_mask_sd2.min()
        g_mask_sd2 = g_mask_sd2/g_mask_sd2.max()

        g_mask_diff = g_mask_sd2

    else:
        g_mask_sd1 = np.exp(-((X-cx)**2+(Y-cy)**2)/(2*sd1**2))
        g_mask_sd2 = np.exp(-((X-cx)**2+(Y-cy)**2)/(2*sd2**2))
        g_mask_sd1 = g_mask_sd1-g_mask_sd1.min()
```

```
        g_mask_sd2 = g_mask_sd2-g_mask_sd2.min()
        g_mask_sd1 = g_mask_sd1/g_mask_sd1.max()
        g_mask_sd2 = g_mask_sd2/g_mask_sd2.max()

        g_mask_diff = g_mask_sd2-g_mask_sd1

    if (normalize):
        g_mask_diff /= g_mask_diff.max()
        #g_mask_diff[g_mask_diff < 0] = 0

    return g_mask_diff


def get_MSE(img1,img2):

    '''
    calculate mean squared error of two different images

    Parameters
    ----------
    img1, img2 : images

    Returns
    -------
    mean squared error value

    '''
    h,w = img1.shape
    squared_diff = (img2-img1)**2
    return (1/(h*w))*np.sum(squared_diff)


def make_spreaded_gaussian_line(img_shape, theta, sigma, sigma_tol=0.01):

    '''
    creates line with gaussian gradient at given angle

    Parameters
    ----------
    img_shape : tuple of image mask's size
    theta : angle of line
    sigma : standard deviation offset value
    sigma_tol : spreading of sigma along angled axis

    Returns
    -------
    image filter mask with gaussian line function
    '''
    vs,us = img_shape
    vc, uc = vs//2, us//2
    x = np.linspace(-uc, uc, us)
    y = np.linspace(-vc, vc, vs)
    V,U = np.meshgrid(x, y)

    u_ = U*np.cos(np.radians(theta))+V*np.sin(np.radians(theta))
    w = sigma_tol*(np.sqrt(U**2+V**2))+sigma

    mask = 1-np.exp((-0.5*u_**2)/(w**2))
```

```
    return mask


def gauss(data, offset, amplitude, mean, sigma):

    '''
    gaussian line function

    Parameters
    ----------
    data : one dimensional array with values
    offset : minimum value of gaussian function
    amplitude : maximum value of gaussian function
    mean : location of maximum value
    sigma : standard deviation, defining width of gaussian peak

    Returns
    -------
    gaussian function based on given parameters

    '''

    return offset+(amplitude-offset)*np.exp(-(data-mean)**2/(2*sigma**2))


def get_gaussian_fit_of_line(line_values, show_plot=False):

    '''
    calculate fit function of gaussian estimation

    Parameters
    ----------
    line values : values along a line selection to fit
    show_plot : optional parameter to visualize result of fit

    Returns
    -------
    fitting parameters

    '''

    x_data = np.arange(line_values.shape[0])
    offset_guess = 0
    amplitude_guess = np.max(line_values)
    mean_guess = np.mean(x_data)
    sigma_v = line_values > amplitude_guess/2
    sigma_guess = 2*np.sum(sigma_v)

    popt, pcov = curve_fit(gauss, x_data, line_values,
             p0=[offset_guess,amplitude_guess,mean_guess,sigma_guess])

    if(show_plot):

        pylab.plot(x_data, line_values, label="data")
        pylab.plot(x_data, gauss(x_data,*popt),label="fit")
        pylab.legend()
        pylab.show()

    return popt, pcov
```

```
def readjust_intensity_range(image,new_min,new_max):

    '''
    image intensity range transformation

    Parameters
    ----------
    image : array containing the image intensity values
    new_min : minimum value of new intensity range
    new_max : maximum value of new intensity range

    Returns
    -------
    transformed image with new intensity range

    '''

    alow = image.min()
    ahigh = image.max()

    return (new_min + (image-alow)*(new_max-new_min)/(ahigh-alow))


def get_padding_size(image):

    '''
    calculate required pixel distances for padding image, so that image can
    be placed in squre with size of power by two...avoiding ringing artifacts

    Parameters
    ----------
    image : image data to extract shape information

    Returns
    -------
    width and height related pixel (half-) distances

    '''

    h, w = image.shape

    v = 0

    if w > h:
        v = w
    else:
        v = h

    pv = np.ceil(np.log2(v))

    return (int(((2**pv)-h)//2), int(((2**pv)-w)//2))


# define image location
# asking user for directory - where are the images?
root = tk.Tk()
root.withdraw()
```

```
dir_path = filedialog.askdirectory()

if (dir_path == "" or dir_path == ()):
    print("no selection...exiting script!")
    sys.exit()

print("selected directory")
print(dir_path)
image_list = []

for (dirpath, dirnames, filenames) in os.walk(dir_path):

    image_list.extend(filenames)
    break

# # create destriping folder
results_folder_path = os.path.join(dir_path,"destriped/")
os.mkdir(results_folder_path)

for selected_img in image_list:

    # load image and display the image cliped to the first and last percentile
    img = img_as_float(io.imread(os.path.join(dir_path,selected_img),key=0))
    print(selected_img)
    #img = img_as_float(io.imread(img_path+img_name))

    #check if power of two
    ph_offset, pw_offset = get_padding_size(img)
    img = np.pad(img,((ph_offset,ph_offset),(pw_offset,pw_offset)),'linear_ramp')
    img_min = img.min()
    img_max = img.max()
    img_mean = img.mean()
    img_h, img_w = img.shape
    img_cy, img_cx = img_h//2, img_w//2

    # # image plot for development
    # p_min = np.percentile(img,0.5)
    # p_max = np.percentile(img,99.5)
    # pylab.imshow(img, cmap="gray", vmin=p_min, vmax=p_max)
    # pylab.colorbar()
    # pylab.show()

    # Take the 2-dimensional DFT and centre the frequencies
    fftimage = np.fft.fft2(img)
    fftimage = np.fft.fftshift(fftimage)
    p_min = np.percentile(np.log(np.abs(fftimage)+1),0.1)
    p_max = np.percentile(np.log(np.abs(fftimage)+1),99.9)
    # image plot for development
    # #pylab.imshow(np.log(np.abs(ftimage[img_cy-512:img_cy+512,
    # img_cx-512:img_cx+512])+1), cmap="gray", vmin=p_min, vmax=p_max)
    # pylab.imshow(np.log(np.abs(fftimage)+1))
    # pylab.colorbar()
    # pylab.show()

    #io.imsave('fft_from_stripy_image.tif', np.log(np.abs(fftimage)+1))

    # calc center region fit, which should be uneffected of destriping

    central_line_values = fft_angular_linePlot(fftimage, 0,
```

```
            getValues=True, showPlot=False)
    popt_fft, pcov_fft =
            get_gaussian_fit_of_line(np.log(np.abs(central_line_values)+1))

    angles = [99.5, 80.5, 90] # device related angles

    fftimage_orig = np.copy(fftimage)
    img_dupl = np.copy(img)
    img_dupl2 = np.copy(img)

    for a in angles:

        fftimage = np.fft.fft2(img_dupl)
        fftimage = np.fft.fftshift(fftimage)
        filter_mask = abs(make_spreaded_gaussian_line(fftimage.shape, a,
             2*3.4, 0.048)-1) # 0.048 is total angular difference in px

        central_line_values2 = fft_angular_linePlot(fftimage_orig, a,
                                getValues=True, showPlot=False)
        popt_fft2, pcov_fft2 =
             get_gaussian_fit_of_line(np.log(np.abs(central_line_values2)+1))

        filter_mask *= get_doG_bandpass(fftimage.shape, round(popt_fft[3])/2,
                                2*popt_fft2[3])

        # testing different weights...mean association not found so far
        #weight = popt_fft2[1]/popt_fft[1]
        weight = 1
        filter_mask = np.abs((filter_mask*weight)-1)
        #io.imsave("filter_mask_"+str(int(a))+".tif",filter_mask)
        #print(popt_fft)
        #print(popt_fft2)

        fftimage *= filter_mask
        img_dupl = np.abs(np.fft.ifft2(np.fft.ifftshift(fftimage)))

    # #image plot after destriping for development
    # pylab.imshow(img_dupl[ph_offset:img_h-ph_offset,pw_offset:img_w-pw_offset])
    # pylab.colorbar()
    # pylab.show()

    #mean squared (difference) error for development
    #print(get_MSE(img*2**16-1, img_dupl*2**16-1))
    # save image in results folder
    #io.imsave('results_test/destriped_'+img_name, (img_dupl[ph_offset:img_h-
ph_offset,pw_offset:img_w-pw_offset])*2**16-1)
    io.imsave(os.path.join(results_folder_path,selected_img),
                    np.uint16((img_dupl[ph_offset:img_h-ph_offset,
                    pw_offset:img_w-pw_offset])*2**16-1))
```

8.1.13. SNR calculation of stripy and destriped LSFM images

```
 """
Calculate SNR values based on ilastik's probability maps
"""
import pandas as pd
import numpy as np
from skimage import io
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

def show_prob_maps(prob_maps):

    colors_R = [(0, 0, 0), (1, 0, 0)] # first color is black, last is red
    cm_r = LinearSegmentedColormap.from_list("Custom", colors_R, N=256)

    colors_G = [(0, 0, 0), (0, 1, 0)] # first color is black, last is green
    cm_g = LinearSegmentedColormap.from_list("Custom", colors_G, N=256)

    colors_B = [(0, 0, 0), (0, 0, 1)] # first color is black, last is blue
    cm_b = LinearSegmentedColormap.from_list("Custom", colors_B, N=256)

    plt.imshow(prob_maps[:,:,0], cmap=cm_r, alpha=0.33)
    plt.imshow(prob_maps[:,:,1], cmap=cm_g, alpha=0.33)
    plt.imshow(prob_maps[:,:,2], cmap=cm_b, alpha=0.33)
    plt.title("probability input maps")
    plt.show()

def show_image(image, img_title="", color="gray", p_low=1, p_high=99):

    p_l = np.percentile(image,p_low)
    p_h = np.percentile(image,p_high)

    plt.imshow(image, cmap=color, vmin=p_l, vmax=p_h)
    plt.colorbar()
    plt.title(img_title)
    plt.show()

def get_mean_std_of_probmap(image,prob_map,threshold=0.66):

    prob_th_ids = prob_map > threshold
    obj_mean = (image[prob_th_ids]).mean()
    obj_std = (image[prob_th_ids]).std()

    return obj_mean, obj_std

# load image data
img_path = "H:/LightSheet/160101_depicted_examples_results/"
img_path2 = "H:/LightSheet/160101_depicted_examples/"

# define microscope related image naming
date = "16-01-01"
device = "UltraII"
channel = "C01"
stage = "xyz-Table "
file_suffix = ".ome.tif"

# select image layers to calculate SNRs
layers = ["Z0250","Z0300","Z0350","Z0400","Z0435","Z0450","Z0500"]
```

```
empty_array = np.zeros(len(layers))
# prepare results table
idx = 0
snr_data = pd.DataFrame({"layer" : layers,
                         "PSNR_orig" : empty_array,
                         "PSNR_destr" : empty_array,
                         "SNR_orig" : empty_array,
                         "SNR_destr" : empty_array,
                         "SBR_orig" : empty_array,
                         "SBR_destr" : empty_array})

for z in layers:

    # read the image data
    filename_orig = img_path2+date+"_"+device+"_"+channel+"_"+stage+str(z)+
       file_suffix
    filename_destriped = img_path+"destriped_"+date+"_"+device+"_"+channel+
       "_"+stage+str(z)+file_suffix
    filename_prob = img_path+"destriped_"+date+"_"+device+"_"+channel+"_"+
       stage+str(z)+".ome_Probabilities.tif"

    orig_img = io.imread(filename_orig)
    destriped_img = io.imread(filename_destriped)
    prob_map = io.imread(filename_prob)

    mus = []
    stds = []

    # calculate features
    for i in range(prob_map.shape[2]):

        mu_orig, stdDev_orig =
             get_mean_std_of_probmap(orig_img, prob_map[:,:,i])
        mu_destr, stdDev_destr =
             get_mean_std_of_probmap(destriped_img, prob_map[:,:,i])
        mus.append(mu_orig)
        stds.append(stdDev_orig)
        mus.append(mu_destr)
        stds.append(stdDev_destr)

    # set values in table
    snr_data["PSNR_orig"][idx] = mus[0]/stds[4]
    snr_data["PSNR_destr"][idx] = mus[1]/stds[5]

    snr_data["SNR_orig"][idx] = mus[0]/stds[2]
    snr_data["SNR_destr"][idx] = mus[1]/stds[3]

    snr_data["SBR_orig"][idx] = mus[0]/mus[2]
    snr_data["SBR_destr"][idx] = mus[1]/mus[3]

    idx += 1

print(snr_data)

# save the results table
snr_data.to_csv(img_path+"SNR_results.csv")
```

9. List of publications and author contribution

9.1. Publications as part of this thesis

Radbruch, H., Mothes, R., Bremer, D., Seifert, S., Köhler, R., Pohlan, J., Ostendorf, L., Günther, R., Leben, R., Stenzel, W., Niesner, R. A., & Hauser, A. E. (2017). Analyzing nicotinamide adenine dinucleotide phosphate oxidase activation in aging and vascular amyloid pathology. *Frontiers in Immunology*, *8*(JUL), 1–12. https://doi.org/10.3389/fimmu.2017.00844

Kriegel, F. L., Köhler, R., Bayat-Sarmadi, J., Bayerl, S., Hauser, A. E., Niesner, R., Luch, A., & Cseresnyes, Z. (2018). Morphology-Based Distinction Between Healthy and Pathological Cells Utilizing Fourier Transforms and Self-Organizing Maps. *Journal of Visualized Experiments : JoVE*, *140*. https://doi.org/10.3791/58543

Kriegel, F. L., Köhler, R., Bayat-Sarmadi, J., Bayerl, S., Hauser, A. E., Niesner, R., Luch, A., & Cseresnyes, Z. (2018). Cell shape characterization and classification with discrete Fourier transforms and self-organizing maps. *Cytometry Part A*, *93*(3), 323–333. https://doi.org/10.1002/cyto.a.23279

Holzwarth, K., Köhler, R., Philipsen, L., Tokoyoda, K., Ladyhina, V., Wählby, C., Niesner, R. A., & Hauser, A. E. (2018). Multiplexed fluorescence microscopy reveals heterogeneity among stromal cells in mouse bone marrow sections. *Cytometry Part A*, *93*(9), 876–888. https://doi.org/10.1002/cyto.a.23526

Pascual-Reguant, A., Köhler, R., Mothes, R., Bauherr, S., Hernández, D. C., Uecker, R., Holzwarth, K., Kotsch, K., Seidl, M., Philipsen, L., Müller, W., Romagnani, C., Niesner, R., & Hauser, A. E. (2021). Multiplexed histology analyses for the phenotypic and spatial characterization of human innate lymphoid cells. *Nature Communications*, *12*(1), 1–15. https://doi.org/10.1038/s41467-021-21994-8

Mothes, R., Pascual-Reguant, A., Koehler, R., Liebeskind, J., Liebheit, A., Bauherr, S., Philipsen, L., Dittmayer, C., Laue, M., von Manitius, R., Elezkurtaj, S., Durek, P., Heinrich, F., Heinz, G. A., Guerra, G. M., Obermayer, B., Meinhardt, J., Ihlow, J., Radke, J., … Hauser, A. E. (2023). Distinct tissue niches direct lung immunopathology via CCL18 and CCL21 in severe COVID-19. *Nature Communications*, *14*(1), 791. https://doi.org/10.1038/s41467-023-36333-2

9.2. Additional publication contribution

Riedel, R., Addo, R., Ferreira-Gomes, M., Heinz, G. A., Heinrich, F., Kummer, J., Greiff, V., Schulz, D., Klaeden, C., Cornelis, R., Menzel, U., Kröger, S., Stervbo, U., Köhler, R., Haftmann, C., Kühnel, S., Lehmann, K., Maschmeyer, P., McGrath, M., … Radbruch, A. (2020). Discrete populations of isotype-switched memory B lymphocytes are maintained in murine spleen and bone marrow. *Nature Communications*, *11*(1), 1–14. https://doi.org/10.1038/s41467-020-16464-6

Du, W., Lenz, D., Köhler, R., Zhang, E., Cendon, C., Li, J., Massoud, M., Wachtlin, J., Bodo, J., Hauser, A. E., Radbruch, A., & Dong, J. (2021). Rapid Isolation of Functional ex vivo Human Skin Tissue-Resident Memory T Lymphocytes. *Frontiers in Immunology*, *12*(March), 1–12. https://doi.org/10.3389/fimmu.2021.624013

Dornieden, T., Sattler, A., Pascual-Reguant, A., Ruhm, A. H., Thiel, L. G., Bergmann, Y. S., Thole, L. M. L., Köhler, R., Kuhl, A. A., Hauser, A. E., Boral, S., Friedersdorff, F., & Kotsch, K. (2021). Signatures and specificity of tissue-resident lymphocytes identified in human renal peritumor and tumor tissue. *Journal of the American Society of Nephrology*, *32*(9), 2223–2241. https://doi.org/10.1681/ASN.2020101528

Treimer, W., & Köhler, R. (2021). Determination of the spatial resolution in the case of imaging magnetic fields by polarized neutrons. *Applied Sciences (Switzerland)*, *11*(15). https://doi.org/10.3390/app11156973

9.3. List of conference contributions

Köhler, R. , Holzwarth, K., Philipsen, L., Niesner, R.A., Hauser, A.E., “Histo Cytometry using Multi Epitope Ligand Cartography (MELC) in the bone marrow”, CYTO 2018 33rd Congress of the international society for advancement of cytometry, Prague, Czech Republic, oral presentation

Köhler, R. , Holzwarth, K., Philipsen, L., Wählby, C., Niesner, R.A., Hauser, A.E., “Histo Cytometry using Multi Epitope Ligand Cartography (MELC)”, 28th Annual Conference of the German Society for Cytometry 2018, Jena, Germany, oral presentation

Köhler, R., Holzwarth, K., Pascual-Reguant, A., Philipsen, L., Wählby, C., Niesner, R.A., Hauser, A.E., “Image preprocessing of multiplexed data for better and easier (2D-) segmentation”, 3rd NEUBIAS Conference – Luxembourg 2019, Luxembourg, oral presentation

Köhler, R., Holzwarth, K., Pascual-Reguant, A., Bauherr, S., Philipsen, L., Wählby, C., Niesner, R.A., Hauser, A.E., “Histo Cytometry using Multi Epitope Ligand Cartography (MELC)”, 3rd NEUBIAS Conference – Luxembourg 2019, Luxembourg, poster presentation

Koehler, R., Mothes, R. , Pascual-Reguant, A., Liebeskind, J., Liebheit, A., Bauherr, S., Philipsen, L., Niesner, R.A., Radbruch, H., Hauser, A.E., , “Spatial analysis of tissues at the protein and transcriptional level”, ZIBI Retreat 2022, Berlin, Germany, oral presentation

10. Acknowledgements

When I was offered the opportunity to do my PhD as a fresh graduate of my master's degree in applied physics / medical engineering, there was actually no reason to hesitate... man, what a great idea. Looking back, it was an exciting, interesting, funny and sometimes hard way up to this point. As my doctor mother once said to me: "In the doctoral thesis you sometimes have to learn how to deal with frustration." And what can I say, yes, of course she was right about something, but the positive moments definitely outweighed the negative. I learned a lot, got to know great people and I would like to take this opportunity to thank everyone.

First and foremost I would like to thank my supervisors Prof. Dr. Raluca A. Niesner (Biophysical Analytics) and Prof. Dr. Anja E. Hauser (Immune Dynamics) for excellent, motivating and analytical support during my PhD. The door to their office was always open to me and they were able to advise me on any question, no matter how strange. They were also open to new ideas and accordingly gave me a lot of freedom to establish them. The familiar working atmosphere within the working groups invited people to continue and contributed to the further scientific development of everyone.

As part of the NieHau working group family, I would of course also like to thank all my colleagues, students and interns for the incredibly great time with you at the German Rheumatism Research Center in Berlin. Together we made our everyday work as pleasant as possible, even if one or the other project drove us into the evenings. Together we could learn from each other and broaden our horizons. Just like my meanwhile two multiplex microscopes Jürgen and Jutta, some of which I had to disassemble down to their individual parts on my table.

Natürlich möchte ich mich zu guter Letzt auch noch bei meiner Familie, zukünftigen Ehefrau und meinen Freunden bedanken. Ihr habt mich durch alle Phasen begleitet, aufgebaut und mir immer wieder neue Kraft gegeben um durchzuhalten. Wie auch schon im Studium, hat mich innerhalb der Promotion ein weiterer Familienteil verlassen. Papa, ich bin mir sicher, dass du trotzdem von oben runterguckst, dich freust und alle möglichen bösen Dinge von mir fernhältst.

I THANK YOU ALL

11. Finanzierungsquellen, Interessenskonflikte und Selbständigkeitserklärung

Finanzierungsquellen

Die Arbeit wurde finanziell im Rahmen der Anstellung als Mikroskop Operateur des Deutschen Rheuma Forschungszentrums der Arbeitsgruppe Hauser - Immundynamik unterstützt.

Interessenkonflikte

Im Rahmen dieser Arbeit bestehen keine Interessenkonflikte durch Zuwendung Dritter.

Selbständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Ich versichere, dass ich ausschließlich die angegebenen Quellen und Hilfen in Anspruch genommen habe.

Berlin, den 28.11.2023 Ralf Köhler

mbvberlin mensch und buch verlag

49,90 Euro | ISBN: 978-3-96729-236-7