Aus dem Zentrum für Kinder- und Jugendmedizin der Universität Heidelberg
(Geschäftsführender Direktor: Univ. - Prof. Dr. med., Prof. h. c. (RCH) Georg F. Hoffmann)
Abteilung für Pädiatrische Nephrologie

Computer vision and data-analysis solutions for phenotypic screening of small model organisms

Inauguraldissertation
zur Erlangung des Doctor scientiarum humanarum (Dr. sc. hum.)
an der
Medizinischen Fakultät Heidelberg
der
Ruprecht-Karls-Universität

vorgelegt von
Laurent Thomas

aus
Saint-Brieuc, Frankreich

2020

Dekan: Prof. Dr. Wolfgang Herzog

Doktorvater: Prof. Dr. med. Franz Schaefer

Table of Contents

List of abbreviations

API: Application Programming Interface

CNN: Convolutional Neural Network

CPU: Central Processing Unit (Processor)

GUI: Graphical User-Interface

GPU: Graphics Processing Unit (Graphic card)

MTM: Multi-Template-Matching

NMS: Non-Maxima Suppression

PCA: Principal Component Analysis

ROI: Region Of Interest

SVM: Support Vector Machine

TP: True Positive

FP: False Positive

TN: True Negative

FN: False Negative

List of figures

1. Introduction

1.1. Microscopy for life-science research

1.1.1. A valuable tool, a diversity of techniques

As the idiom "seeing is believing" reads, microscopy offers an incredibly powerful tool to shed light on complex microscopic structures and phenomena. With resolutions in the range of few nanometres for electron microscopes and around half a micrometre for light microscopes, these technologies are well adapted to the observation of biological objects such as individual cells. Light-microscopy is particularly popular in life-science research, thanks to its flexibility and accessibility. It allows the observation of living or fixed cells, tissues or microscopic organisms at various scales, or following dynamic processes over time, in a single image plane or in 3D. Especially using fluorescence labelling, specific molecular targets can be highlighted, even within living organisms. While the resolution with such imaging systems is still limited by the diffraction of the light (Abbe 1873), recent techniques relying on image-reconstruction or stimulated emission are able to generate super-resolved images, beyond the diffraction limit.

One implication to keep in mind with light-microscopy is the trade-off between the different aspects of the acquisition: the spatial resolution, the temporal resolution (or speed), the signal sensitivity, and the phototoxicity (i.e. sample health) often represented as a pyramid (Figure 1) (Scherf and Huisken 2015). For a given imaging system, this trade-off is fixed such that emphasizing one aspect will systematically impact the others. Widefield and confocal microscopy are widely used in life-science research for the imaging of cells or small organisms. These systems are compatible with high throughput and time-lapse imaging, while being affordable compared to more sophisticated setups. However, for fluorescence imaging, widefield microscopy suffers from the contribution of out-of-focus light, which is partly rejected in confocal microscopy thanks to the pinhole (Minsky 1988). Alternatively, deconvolution of the acquired images can improve the signal to noise ratio after acquisition (Shaw 1995; Swedlow and Platani 2002). Widefield and confocal microscopy have limited penetration depth, due to the diffusion of light in the tissues, and thus are not adapted for the imaging of thick samples (Dudko and Weiss 2005). 2-photons microscopy offers a higher penetration depth thanks to the longer wavelengths used for the excitation, as well as an increased axial resolution, but requires powerful lasers in the near-infrared range, which might be detrimental for the sample (Denk et al. 1990). Single-plane illumination microscopy (SPIM also called light-sheet fluorescence microscopy - LSFM) relies on a thin plane of excitation light, orthogonal to the optical axis (Santi 2011). It achieves enhanced optical sectioning, ideal for the three-dimensional high-resolution imaging of large transparent samples such as fish or drosophila embryos (Huisken 2004). This technique is compatible with the time-lapse imaging of live samples and has low photo-toxicity but generates large amount of data and is limited in sample throughput.

All methods previously mentioned have a spatial resolution limited by the diffraction limit to about half the excitation wavelength in the lateral direction and about one wavelength along the axial direction (Abbe 1873; Stelzer 2002). Novel methods relying on image-reconstruction were proposed to find an alternative to the optical resolution limit. With structured illumination microscopy (SIM), a sequence of images illuminated with patterned excitation light translated and rotated is acquired in widefield, which allows the reconstruction of super-resolved images of about twice the diffraction-limited resolution, with low phototoxicity (Dougherty and Goodwin 2011; Gustafsson 2000). Using single-molecule imaging by photo-activation (PALM (Betzig et al. 2006)) or stochastic fluorescence emission (STORM (Rust et al. 2006)) coupled to temporal acquisition, even more resolved images down to few dozen nanometres resolution can be reconstructed. Due to the reconstruction process, these methods are however limited in their temporal resolution, and thus not optimal for dynamic time-lapse imaging. Finally, stimulated fluorescence emission (STED (Hell and Wichmann 1994)), also achieves super-resolved images in the range of 50 nanometres resolution with biological samples, but requires high-intensity lasers for the stimulation-emission-depletion.

Every microscopy technique offers a different disposition of the acquisition trade-off (Figure 1). Yet, with such diversity of imaging modalities, complementary experiments can be carried out to explore different problematics. For phenotypic screening for instance, high-throughput imaging at low to medium resolution is typically carried out first to identify interesting conditions, which can be further explored at higher-resolution or in time-lapse experiments, to better characterise the phenotypic response (Neumann et al. 2010).

FIGURE REMOVED FOR ONLINE ARCHIVING TO COMPLY WITH FIGURE´s COPYRIGHT

Figure 1: Trade-off in light-microscopy represented by the summits of a pyramid

For a given imaging technique, this trade-off is fixed such that emphasizing one aspect will systematically impact the others. Reproduced by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, Nature Biotechnology*, The smart and gentle microscope*, Sherf Nico et al (2015).

1.1.2. Model systems for microscopy

In biomedical research studies, simplified model systems are typically used as they offer a controlled biological environment, compared to complex organisms with multiple tissues, organs, and variable genetic backgrounds. 2D cell cultures are widely used as a starting point for *in-vitro* studies, and a historically well-established model system. They offer multiple experimental advantages including the simplicity of handling (Harrison and Rae 1997), the diversity of biomolecular tools (Davis 2012), and the availability of multiple immortalized cell lines for various species, including humans. Patient-derived primary cell lines can also be used for a maximum of fidelity with a particular pathology or tissue (Stacey 2001). However, the drawback of in-vitro cell culture as a model system resides in its simplicity, which does not reflect the complex micro-environment of the original tissue or organism, with e.g. its extra-cellular matrix, and various signalling pathways (Figure 2). These limitations can compromise the extrapolation of results from cell cultures to higher organisms like mammals, and thus may impact the outcome of biomedical research studies with possible applications to humans e.g. for the identification of novel therapeutic compounds. More complex *in-vitro* systems are thus being developed to access higher physiological relevance, such as 3D cell culture systems (Antoni et al. 2015), human organoids (Kim et al. 2020; Lehmann et al. 2019) or micro-fluidic devices (Meyvantsson and Beebe 2008). While these systems are likely to show higher physiological relevance than flat 2D cell cultures, they still lack the complexity of a living organism. Therefore, small organisms are also used as *in-vivo* models, including the nematode *Caenorhabditis elegans* (Strange 2006)*,* and the fruit fly *Drosophila melanogaster* (Dahmann 2008). Those organisms allow the conduction of more sophisticated and realistic experiments (including behavioural studies) than cell cultures while still being relatively easy to handle and inexpensive. However, these models are not vertebrates and thus are relatively distant from humans (Figure 2). The mouse *Mus musculus* is the vertebrate model organism closest to human (Bronson et al. 1966; Hedrich and Bullock 2004), but it is relatively expensive, complex to image using microscopy and limited in terms of experimental throughput. Alternatively, small vertebrate organisms such as the zebrafish *Danio rerio* have emerged as novel model systems that are easy to breed, inexpensive and well suited for microscopy imaging (Lieschke and Currie 2007). A major advantage of small model organisms is the possibility to observe complex phenotypic responses to genetic or chemical perturbation (Brady et al. 2016; Kaufman et al. 2009), providing a better overview of the range of effects on a living organism. Example of phenotypic readouts classically reported in phenotypic studies include survival rate, overall or organ-specific morphological profiling, cellular and tissue-specific phenotypes, reporter gene expression patterns or behavioural response (Cornet et al. 2017; Gehrig et al. 2018; Kokel et al. 2010).

FIGURE REMOVED FOR ONLINE ARCHIVING TO COMPLY WITH FIGURE´s COPYRIGHT

Figure 2: A comparison of model systems compatible with microscopy imaging

Different model systems are available for biomedical research, with various levels of physiological relevance to humans, and complexity of handling (PDX: Patient-derived Xenografts). Relative scores for a set of selected criterium related to the handling of the specimen and the physiological relevance are represented as being the best (dark green tick), good (light green tick), partly suitable (yellow tick) and not suitable (red cross). In particular, the zebrafish (*D.rerio*) offers a good compromise between complexity of handling and physiological relevance. Reproduced by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, Nature Reviews Molecular Cell Biology, *Human organoids: model systems for human biology and medicine*, Jihoon Kim et al (2020).

1.1.3. Microscopy-based screening: a tool for biomedical research

With the advent of automation, modern microscopy systems can generate tremendous amounts of high-quality images in a relatively short time. Together with multiplexed sample mounting systems e.g. microtiter plates, the imaging of several samples and experimental conditions can be streamlined. Moreover, thanks to the diversity of fluorescent reporters and transgenic constructs, microscopy-based screening enables complex phenotypic studies for the characterisation of chemical or genetic perturbation in biological systems (Bray et al. 2016; Caicedo et al. 2016; Futamura et al. 2012; Wagner and Schreiber 2016). Image-based screening thus offers a powerful tool for the preliminary testing of research hypothesis e.g. for drug-discovery (Esner et al. 2018; MacRae and Peterson 2015; Simm et al. 2018; Zheng et al. 2013) or disease modelling (Bradford et al. 2017; Kim et al. 2020; Lieschke and Currie 2007), complementing other screening technologies such as sequencing or biochemical assays.

Classical screening studies include chemical screens with libraries of small molecules for the identification of therapeutic compounds, toxic molecules or modulators of signalling pathways (Eggert 2013; Rennekamp and Peterson 2015; Richter et al. 2017; Saydmohammed et al. 2011; Westhoff et al. 2020); and genetic screens for the identification of genes involved in specific biological processes such as mitosis (Neumann et al. 2010; Rines et al. 2008) or embryogenesis (Driever et al. 1996; Sönnichsen et al. 2005). For these genetic studies, the perturbations can be introduced using RNA interference, genome-editing tools or mutagenic agents (Kaufman et al. 2009; Neumann et al. 2006; Quwailid et al. 2004; Sönnichsen et al. 2005). Among the diversity of microscopy techniques, widefield and confocal microscopy are the most common modalities for high-throughput imaging of biological objects. They provide rapid imaging, with sufficient spatial resolution and signal sensitivity for classical phenotyping of cells or small model organisms (Boutros et al. 2015; Perlman 2004; Rennekamp and Peterson 2015). Microscopy-based screening has been widely used to image fluorescently-labelled cell cultures with various cell lines and assays (Boutros et al. 2015; Bray et al. 2016; Esner et al. 2018; Futamura et al. 2012; Neumann et al. 2006). High-throughput imaging also has application for histology and digital pathology with whole slide imaging (Al-Janabi et al. 2012; Pantanowitz et al. 2011). It is for instance used for the identification of intra-cellular parasites (e.g. Malaria) in chemically stained blood samples (Li et al. 2019; Loddo et al. 2019). More recently, multi-cellular organoids (Lukonin et al. 2020) as well as small organisms such as zebrafish larvae (Pandey et al. 2019; Rennekamp and Peterson 2015; Richter et al. 2017; Shah et al. 2020; Westhoff et al. 2020) or medaka embryos (Gierten et al. 2020) have also been imaged at high-throughput with both widefield and confocal systems.

1.1.4. Challenges of microscopy-based screening with small organisms

While the imaging of 2D cell culture in petri dish or well-plate is historically well established, alternative biological systems such as organoids and small model organisms introduce new experimental challenges, in particular for high throughput imaging (Gehrig et al. 2018). First, specimen handling should follow legal animal husbandry regulations (Aleström et al. 2020). Then, the experimental procedure involves a variety of operations including e.g. for zebrafishes, specimen crossing, collection and dechorionation of embryos, experimental treatment (anaesthesia, microinjection) and sample mounting (Kaufman et al. 2009; Nusslein-Volhard and Dahm 2002). These are time-consuming procedures even for trained experimenters. Automated systems exist for some of these operations (Letamendia et al. 2012; Mandrell et al. 2012; Pardo-Martin et al. 2010), still in most laboratories, most steps are performed manually.

At the scale of a microscope, small organisms are large three-dimensional objects, and thus more complicated to image than flat cell cultures. In particular, without adapted mounting strategies, these specimens will adopt random positions and orientations in wells of microtiter plates, which complicates the standardized imaging of tissues or organs of interests at high-throughput (Burns et al. 2005; Gehrig et al. 2009; Peravali et al. 2011; Spomer et al. 2012). Dedicated sample mounting strategies compatible with high-throughput imaging are thus necessary to assure the reliable orientation and positioning of specimens (Alessandri et al. 2017; Chang et al. 2012; Fuad et al. 2018; Pardo-Martin et al. 2010; Popova et al. 2018; Wittbrodt et al. 2014). Yet, even with optimized specimen mounting, the imaging of deeper structure within the organism might still be hampered by the natural pigmentation of tissues or by optical aberration resulting from the diffusion of the illumination within the tissues and mounting medium (Dudko and Weiss 2005). The pigmentation can sometimes be suppressed in pigmentation-mutant transgenic lines, with genetic manipulation or chemical treatment (Karlsson et al. 2001; Lischik et al. 2019; Wakamatsu et al. 2001). The optical aberrations can also be limited by reducing the refractive index mismatch between the mounting medium and the sample (Boothe et al. 2017; Fadero and Maddox 2017) or by actively correcting the aberrations with adaptive optics (Booth 2007; Wang et al. 2014). The latter solution requires sophisticated instrumentation though, which is usually not compatible with high throughput imaging.

Another challenge for the imaging of specific tissues or organ in small organisms, is that despite adapted mounting systems, the uncertainty in the positioning usually still ranges in the order of few hundred micrometres. At the scale of a high magnification objective, this can represent a major deviation of the position by the equivalent of several fields of view. Therefore, most small organism screening studies to date are limited to low resolution images encompassing the full specimen

(Peterson and Fishman 2011; Rennekamp and Peterson 2015), and thus have limited insight into finer phenotypes at the tissue or cellular scale. This limitation could be overcome though, using feedback microscopy for the targeted high-resolution imaging of regions of interest (ROI) previously localized from low-resolution images (see section 1.4).

With small model organisms, achieving standardized imaging conditions compatible with the automated analysis by image-processing workflows involves the optimization of multiple conditions: sample mounting, choice of the staining/fluorescent reporter, acquisition parameters, anaesthesia, etc. (Lischik et al. 2019). Moreover, microscopy-based screening heavily relies on automated acquisition (e.g. autofocus), therefore the scaling-up of an experiment from a few samples, which acquisitions were typically optimized on a sample-by-sample basis, to a fully automated protocol requires major adaptation efforts. In addition to the challenge of automation, the multi-dimensional imaging of numerous samples (3D volume, x channels, y time-points) results in large datasets comprising from several dozens of gigabytes to few terabytes depending on the scale of the study. This tremendous quantity raises serious challenges for researchers, first to deal with the storage and general handling of the data, but also for the visualization and analysis of those datasets. Those experimental considerations ultimately limit the complexity of image-based screening studies. While these burdens may be overcome by adapted hardware architecture and software tools, most software solutions are dedicated to cell-based systems, and often not broadly applicable to alternative biological models. Novel solutions dedicated to alternative microscopy specimens such as small model organisms and organoids are thus necessary to put off these limitations, and to enable complex screening workflows empowering biomedical research in these promising model systems.

1.2. Zebrafish as a model organism for microscopy-based screening

1.2.1. An advantageous model for biomedical research

The zebrafish *Danio rerio* offers a number of advantages as a vertebrate model organism. First it shares high genetic similarities with humans, with about 70% of its genome containing orthologs to human genes, and 86% of known drug targets (Gunnarsson et al. 2008; Howe et al. 2013). Besides, thousands of transgenic and mutant lines are available, including models of human diseases (Bradford et al. 2017; Geisler et al. 2016; Lieschke and Currie 2007), while classical biomolecular tools (CRISPR, RNA interference) can be used to further study genetic profiles (Koster and Sassen 2015; Marques et al. 2019). These characteristics enable complex *in-vivo* studies for biomedical research with applications for human disease modelling, drug discovery, safety applications, environmental and toxicological studies (Brady et al. 2016; MacRae and Peterson 2015). Besides the genetic background, zebrafish embryos present additional properties that are ideal for microscopy and high-content screening: rapid reproduction and large number of offspring, small sizes compatible with wells of microtiter plates, optical transparency, *ex-utero* development, and rapid organogenesis (Kaufman et al. 2009). Thanks to their external embryonic development and optical transparency in the first stages, zebrafishes are a model of choice for developmental research (Cha and Weinstein 2007; Hocking et al. 2013; Kimmel et al. 1995; Nüsslein-Volhard 2012; Rieger et al. 2011; Zhou et al. 2010) (Figure 3.B). With the availability of model diseases and transgenic lines, zebrafish larvae have also been extensively used to study multiple tissues, organs and related pathologies (Figure 3.A) such as cardiovascular development and diseases (Bakkers 2011; Nguyen et al. 2008), liver cancer and diseases (Jang et al. 2012; Lam and Gong 2006), kidney development and diseases (Gehrig et al. 2018; Hostetter et al. 2003; Pandey et al. 2019), infectious diseases (Masud et al. 2017), oncology (Langenau et al. 2003; White et al. 2013; White 2015) but also regeneration (Marques et al. 2019; Petrie et al. 2014; Zhou et al. 2010). Finally, under the current legislation, zebrafish embryos comply with the 3R principle (Replacement, Reduction and Refinement) for the more ethical use of animals in research testing (Ball et al. 2014; Kirk 2018; Russell 1995)

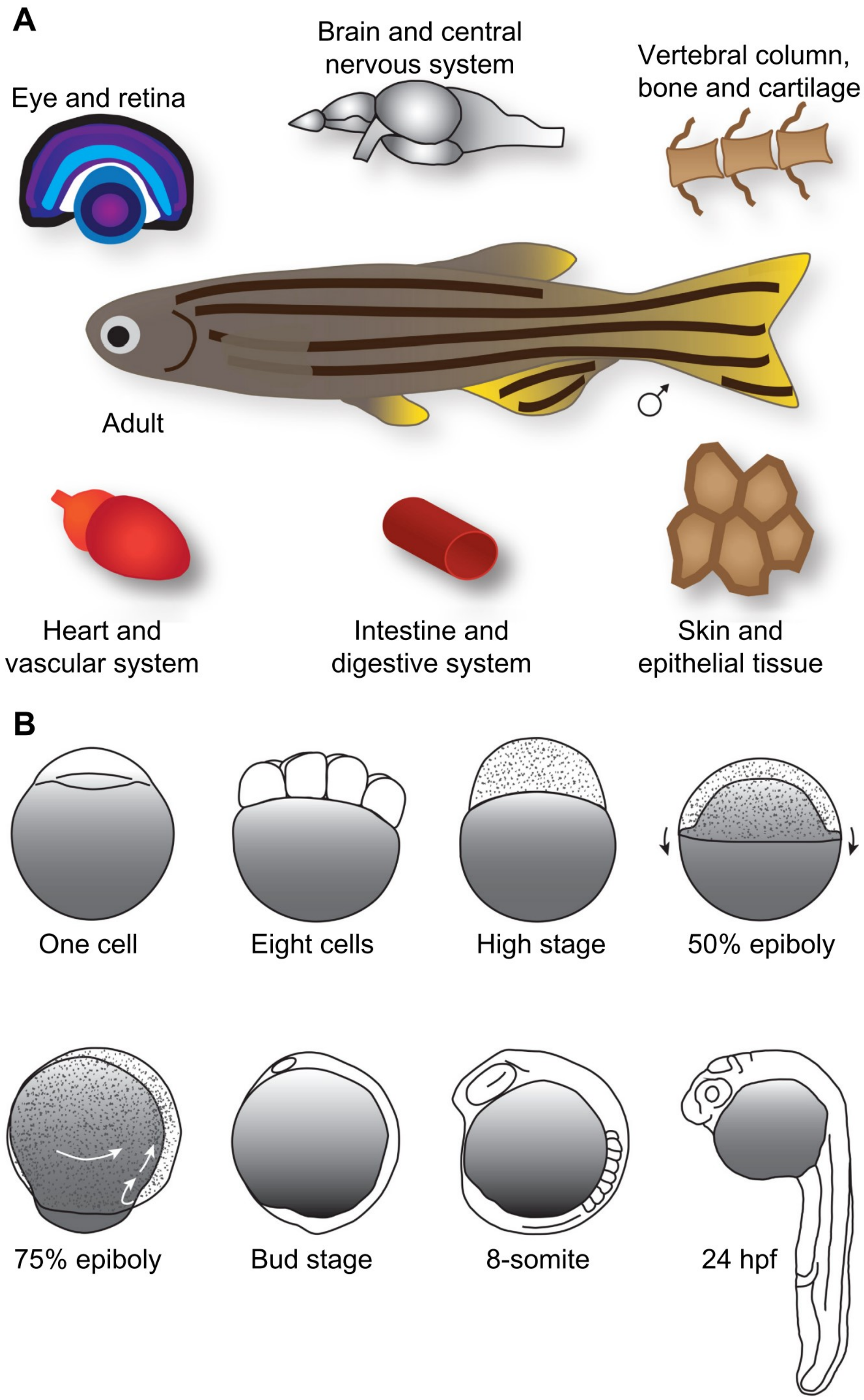

A
Eye and retina
Brain and central nervous system
Vertebral column, bone and cartilage
Adult
♂
Heart and vascular system
Intestine and digestive system
Skin and epithelial tissue
B
One cell
Eight cells
High stage
50% epiboly
75% epiboly
Bud stage
8-somite
24 hpf

Figure 3: Zebrafish as a model organism for vertebrate development and human diseases

A) Example of tissues and organs for which zebrafish is used as a model. B) Ex-utero developmental stages of zebrafish embryos during the first 24 hours post-fertilization. Reproduced from (Koster and Sassen 2015) under the terms of the Creative Commons Attribution - Non Commercial (unported, v3.0) License, licenced by Dove Medical Press Limited. The full terms of this license are available at https://www.dovepress.com/terms.php.

1.2.2. Zebrafish for microscopy-based screening

Thanks to their advantageous genetic and physical properties, zebrafish larvae and embryos have become a model of choice for chemical and genetic microscopy-based screening (Brady et al. 2016; Kaufman et al. 2009; Rennekamp and Peterson 2015). Previous chemical screening with libraries of small molecules in zebrafish lead to the identification of regulators of vertebrate development (Peterson et al. 2000; Richter et al. 2017), modulators of heart rate (Burns et al. 2005), neuro-active molecules (Copmans et al. 2016; Jenkins and Urban 2010; Kokel et al. 2010), compounds impairing kidney development (Westhoff et al. 2020) or affecting cyst formation in the developing kidney (Pandey et al. 2019). Genetic screens in zebrafish similarly identified genes involved in embryogenesis (Driever et al. 1996), in the integrity of the glomerular filtration barrier (Hanke et al. 2013; Hentschel et al. 2007) or in the formation of cysts in the kidney (Sun et al. 2004). To tackle the experimental challenges associated with the high-throughput imaging of zebrafish larvae (see section 1.1.4), previous researches reported the development of dedicated mounting systems for the reliable positioning and orientation of zebrafish larvae or embryos, using micro-fabrication (Alessandri et al. 2017; Popova et al. 2018; Wittbrodt et al. 2014) or sophisticated microfluidic systems (Chang et al. 2012; Fuad et al. 2018; Pardo-Martin et al. 2010). A number of custom software solutions were also described to automate specific phenotypic readouts (Mikut et al. 2013; Xia et al. 2013) including the quantification of heartbeat (Fink et al. 2009; Gierten et al. 2020; Pylatiuk et al. 2014; Sampurna et al. 2018; Spomer et al. 2012), the morphological profiling of bone and cartilage (Stern et al. 2011b; Stern et al. 2011a; Tarasco et al. 2020), the quantification of organ pigmentation (Schutera et al. 2016; Tarasco et al. 2020), the segmentation and morphological profiling of tissues and organs (Annila et al. 2013; Gehrig et al. 2009; Teixidó et al. 2018), the quantification of specimen motion (Kato et al. 2004; Lischik et al. 2019; Marcato et al. 2015) and the identification of fertilization state (Shang et al. 2019). By combining such image-processing solutions with automated microscopes, feedback microscopy was also applied for the automated imaging of the heart (Spomer et al. 2012) and kidney (Pandey et al. 2019) in zebrafish larvae. Finally, standard assays have been proposed as references to study specific tissues and associated signalling pathways, such as the tail-fin amputation and related wound assay to study regeneration and the immune response (Marques et al. 2019; Renshaw et al. 2006), the “eye-assays” to identify disruption of the glomerular filtration barrier by monitoring the level of a heavy molecular weight fluorescent reporter in the blood circulation (Hanke et al. 2015), the photomotor response (PMR) assay, applicable to identify neuroactive agents (Kokel et al. 2010) or the combination of organ-specific assays to quantify overall compounds toxicity (Cornet et al. 2017).

1.3. Image-analysis for microscopy (Bioimage Analysis)

1.3.1. Digital image-analysis

Image-analysis, or image-processing share similarities with signal processing, with common operations such as the application of filters, signal deconvolution or arithmetic operations. However, with image-processing, one must deal with more than 1 dimension, typically the 2 dimensions of an image-frame but potentially more when working with time-lapse or volumetric data. In digital images, the original signal from a sensor is stored as a discrete 2-dimensional array of values, the pixels, arranged following a grid, the image-matrix. For many applications including microscopy, the sensor is a camera collecting photons and the pixels are related to the individual detectors composing the camera sensor. For other applications like medical imaging, the image is not obtained by the collection of photons but by another physical process (ultrasound, X-rays). The sensor is therefore not a conventional camera, but the result is still a digital image with similar characteristic than a microscopy image. Digital image-analysis has thus applications in multiple fields, from so-called computer vision for daily-life scenarios such as face recognition or autonomous driving, medical solutions with image-based diagnostic to various research fields such as microscopy and astronomy. While every field has its own specific practices, there is often some overlap and thus common tools between disciplines.

In practice, image-analysis is typically built as workflows or pipelines (Michael R. Berthold et al. 2009; Carpenter et al. 2006) which are a succession of simple operations such as image pre-processing with special filters, binarization, morphological operations, etc. Image-analysis workflows are ultimately designed to derive some information about the content of an image. It can be some qualitative information such as reporting the presence of an object or structure, an indicator of image-quality or assigning an image to a category, or quantitative such as the position or count of objects, their morphological profiling or the measurement of a signal intensity. Designing robust image-analysis workflows requires some technical skills but has the potential to drastically reduce the need for manual inspection of the images, thus saving time and improving reproducibility, especially for large datasets.

1.3.2. Bioimage analysis: when image-analysis meets microscopy

Modern biomedical microscopy studies typically involve image-analysis to derive information about the biological system. Published workflows span from fully automated pipelines to interactive solutions requesting input from the user at multiple steps of the process. Classical applications for biological studies are the segmentation of objects and the measurement of an intensity (fluorescence, pigmentation), a count or morphological profiling of objects or structures (Aung et al. 2019; Carpenter et al. 2006; Legland et al. 2016; Mi-Sun Kang et al. 2016; Wagner and Lipinski 2013), the tracking of object-positions over time (Aaron et al. 2019; Amat et al. 2014; Gallois 2018; Tinevez et al. 2017) or the quantification of colocalization between different objects (Cordelières and Bolte 2014; Costes et al. 2004; Lagache et al. 2015). Often biological samples are labelled using chemical dyes or fluorescent reporters, therefore intensity-based segmentation (i.e. thresholding) is usually employed to isolate the objects and ROIs in the images. This approach is routinely applied for the segmentation of fluorescent cells and nuclei in microscopy images (Bray et al. 2015; Caicedo et al. 2017; Carpenter et al. 2006; Ljosa and Carpenter 2009; Neumann et al. 2006). However, those segmentation workflows typically involve several intermediate steps for the pre-processing of the images (background subtraction (Caicedo et al. 2017; Model and Burkhardt 2001; Singh et al. 2014), denoising (Meiniel et al. 2018)), and the post-processing of the segmentation mask (e.g. morphological operations (Aaron et al. 2019; Legland et al. 2016)). This results in a multiplicity of parameters which render those workflows very application-specific, ultimately preventing their broad applicability to new sets of images. Besides, for other types of samples like small model organisms or organoids, these approaches might not be applicable, in particular for the identification of specific tissues or organs due to a lack of contrast with the background or with the rest of the organism (Figure 5.B).

Computer vision, meanwhile, benefited from years of research on these problematics of pattern and object-recognition. Recently even, some deep learning solutions have been successfully re-purposed from the detection of objects in real-life scenes to microscopic objects such as macromolecular crystals (Bruno et al. 2018) and cell cultures (Waithe et al. 2020). Beyond deep-learning, computer vision could benefit bioimage analysis in many more ways. However, those methods were originally addressed to computer vision experts and targeted to completely different imaging modalities. Therefore, major efforts are required for their benchmarking with concrete biomedical imaging applications, and for their adaptation to a broader audience of non-expert users, such as life science researchers.

1.3.3. Image-based profiling

Images are large multidimensional grid of pixels, which represent a huge amount of numbers. Typical image-frame may range from 200 x 200 to 2000 x 2000 pixels respectively corresponding to 40 thousand and 4 million pixel-values. Therefore, most image-analysis workflow first condense the raw pixel information from images to a fewer number of descriptive figures, the features, providing a more succinct and informative image-profile, compatible with automated analysis (Caicedo et al. 2017). Such features may include morphological descriptors describing the shape of individual objects e.g. after segmentation (Futamura et al. 2012; Legland et al. 2016; Wagner and Lipinski 2013), intensity features related to the intensity distribution of an image or image-region (Papageorgiou et al. 1998; Viola and Jones 2004) or more complex intensity features such as image-moments (Khotanzad and Hong 1990;

Ming-Kuei Hu 1962; Teague 1980) or texture (Haralick and Shanmugam 1973; Heilbronner 1992). By combining different feature types, a very descriptive feature set can be generated. For their CellPainting assay, (Bray et al. 2016) thus reported 1000 features related to the morphology, intensity, texture and adjacency of the cell body, nuclei and cytoplasm of each segmented cell. In comparison, (Caie et al. 2010) used only 150 features to describe the phenotype of cancer cells after drug treatment. While it still seems like a large number, it is much smaller than the original number of pixels. Pre-trained deep learning models can also be used to extract image-features, by using only the “base” of the network containing the convolutional layers (Kumar et al. 2015; Pawlowski et al. 2016; Spanhol et al. 2017). Often only a fraction of the computed features is relevant for a given classification problem, therefore reducing the number of features to the most discriminative ones can help the subsequent classification steps. This can be achieved by filtering redundant features showing high correlation, using an iterative search to identify the most discriminative features or eliminating uninformative features (Caicedo et al. 2017; Loo et al. 2007; Neumann et al. 2006). Dimensionality reduction using e.g. PCA can also be used to reduce the number of features but has the drawback to lose the interpretability of the features and to introduce some approximations.

In rare cases, typically with small images, raw pixel intensities can be directly used as features by flattening the 2D images to 1D vectors. This was demonstrated for the classification of 28x28 pixels grayscale images of hand-written digits (Y. Lecun et al. 1998) and of 20x20 pixels image-patches of cell nuclei (Han et al. 2012) by directly passing the 1D vector of pixel values to a classifier.

1.3.4. Supervised and unsupervised classification

Given a large dataset of images or samples, a classical application is the identification of samples with similar profiles or phenotypic responses. This is used for instance in pharmacological studies to identify compounds with similar molecular targets or modes of action, based on the phenotypic profiles of the treated cells or specimens (Caicedo et al. 2016; Caie et al. 2010; Futamura et al. 2012; Ljosa et al. 2013; Loo et al. 2007; Perlman 2004; Young et al. 2008). There are 2 solutions to classify samples: the supervised and unsupervised approaches, both relying on descriptive features for each sample. With the supervised approach, the samples are classified into predefined categories, whose characteristics are known (e.g. wild-type phenotype vs a known mutant phenotype). A simple approach to classify an unknown sample is to look for the most similar sample(s) of known category (k-nearest neighbour search (Dhanabal and Chandramathi 2011)). Another popular approach is to rely on a classifier, previously trained on a set of example samples for which the target categories are known (Kotsiantis et al. 2007; Lakhmi Prasanna et al. 2017). Widespread classifier algorithms are decision trees and random forests (Breiman et al. 1984; Ho 1995; Liaw and Wiener 2002; Safavian and Landgrebe 1991), support vector machines (SVM) (Cortes and Vapnik 1995; Noble 2006) and artificial neural networks (Lippmann 1989; Ou and Murphey 2007). SVM were previously used to identify mitotic cells and other cell states (Conrad et al. 2011; Neumann et al. 2006) or for the localization of cell nuclei by classifying image-patches (Han et al. 2012). The supervised approach is a rather top-down approach starting from known categories down to the samples. The unsupervised way (also called clustering) is inversely bottom-up, starting from the samples and forming groups (or clusters) of similar samples *de-novo* by aggregating similar samples one after the other, according to a measure of distance between the growing clusters and the remaining samples (Jain et al. 1999; Olaode et al. 2014; Saxena et al. 2017). This distance metric is again based on the sample features. With clustering, neither the categories nor the typical characteristics of the final clusters are known beforehand. The clusters are thus often simply denoted by integer indexes (cluster 1 to N). Classical clustering algorithms includes the k-mean algorithm (Teknomo 2006; VanderPlas 2016), the hierarchical clustering and density-based clustering (DBSCAN (Ester et al. 1996)).

In both the supervised and unsupervised algorithms, all samples of a dataset end up in one of the predefined categories, or in one of the clusters. While the unsupervised way can be applied right away to a dataset without previously labelling example samples and training a classifier, in practice it requires additional efforts to characterize the clusters and to verify that the partitioning of the samples is coherent with the scientific problematic.

1.3.5. Comprehensive visualization with dimensionality reduction

When exploring datasets composed of multiple samples each described by several numerical features, it is always informative to visualize the distribution of the data, before experimenting with supervised or unsupervised classification. While it is relatively straightforward to observe the distribution of the values for one specific feature across the dataset (e.g. with a histogram of values), this solution only provides a limited view of the available information. Therefore, one classical way to have a quick overview of a large quantitative dataset is to use dimensionality reduction to further condense the information from multiple sample features (morphological descriptors, expression levels for a set of genes) to fewer pseudo-coordinates that can be used for 2D or 3D visualization. Dimensionality reduction has thus been extensively used in life science to represent gene expression profiles of different cell populations (Butler et al. 2018; Shah et al. 2020; Stuart et al. 2019) or to visualize image-based phenotypic features (Caicedo et al. 2017; Caie et al. 2010; Lukonin et al. 2020). There are multiple dimensionality reduction methods, among the most popular are the t-distributed Stochastic Neighbour Embedding (t-SNE)(Maaten and Hinton 2008; Wattenberg et al. 2016), Principal Component Analysis (PCA) (Abdi and Williams 2010; Wold et al. 1987) and Uniform Manifold Approximation and

Projection (UMAP) (McInnes et al. 2020). PCA is historically widely used, along other applications than dimensionality reduction, for instance to derive pseudo-faces (eigenfaces) from databases of human faces (Turk and Pentland 1991). Compared to the other methods, PCA has the advantage to be parameter-free. Besides, it is available for many software platforms and programming libraries including KNIME (Michael R. Berthold et al. 2009), the python package scikit-learn (Pedregosa et al. 2011), OpenCV (OpenCV contributors 2000), Orange (Demšar et al. 2013) as well as web applications such as the embedding projector of Tensorflow (Martín Abadi et al. 2015) or SleepWalk (Ovchinnikova and Anders 2019). More technical information about PCA and dimensionality reduction is reported in section 2.9.3 (see also the chapter about PCA in the python data-science handbook (VanderPlas 2016)).

A simple example of dimensionality reduction is shown in Figure 4 using UMAP for the visualization of colour encoding from an array of 4 values (R, G, B, Transparency) reduced to 2 XY-coordinates suitable for visualization as a scatter plot. Each dot in this scatter plot corresponds to a different combination of these 4 components, i.e. to a different colour (used to render the individual dots). As can be seen from the colour coding, similar colours tend to localize in similar area of the plot (i.e. similar X,Y values), meaning that the dimensionality reduction efficiently condensed the information while preserving most of the dataset characteristics.

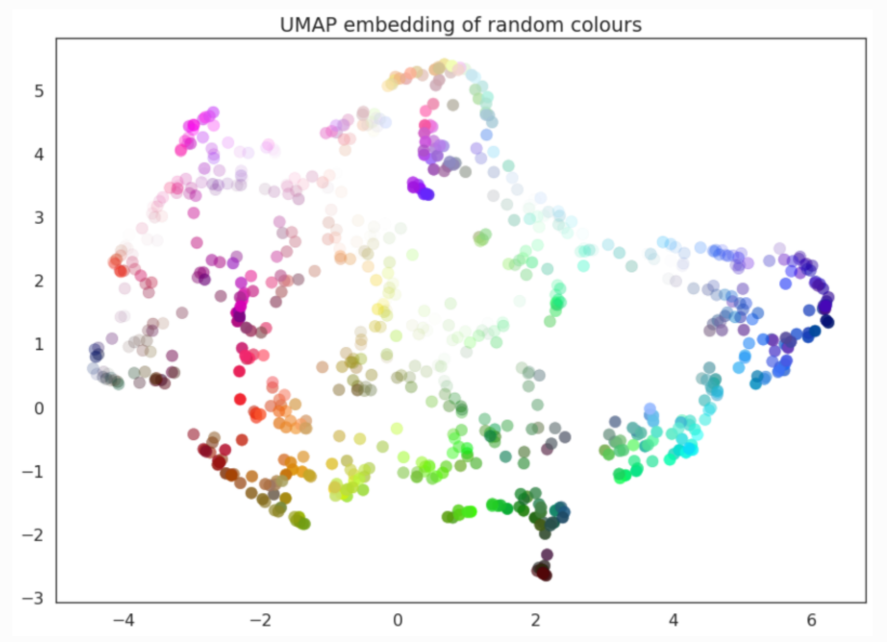


Figure 4: Example of 2D visualization after dimensionality reduction using UMAP

Each point in this 2D point cloud corresponds to a different colour originally encoded as a vector of 4 values (R, G, B, Alpha) corresponding to the 3 RGB colour components and Alpha the transparency. Using UMAP for dimensionality reduction, the colour encoding was reduced from 4 values to 2 components, which are used as X,Y coordinates for the data points. As it can be seen from the plot, similar colours tend to localize in similar areas of the plot, meaning that most of the colour information was conserved after the transformation. Reproduced based on the UMAP online documentation (McInnes 2018).

1.4. Using image-analysis for automated microscopy

1.4.1. Principle of feedback microscopy

The principle of feedback microscopy, also called smart imaging, computer-aided microscopy or adaptive imaging (Scherf and Huisken 2015; Tischer et al. 2014), is to couple the acquisition with image-analysis "on the fly" to perform complex acquisitions, e.g. for the targeted imaging of automatically localized regions of interest, the tracking of moving objects, or the detection of rare events triggering a specific acquisition sequence. Typically, a feedback microscopy protocol involves 3 steps (Conrad et al. 2011; Stuurman 2012). The first step is the acquisition of an overview image following a relatively simple pre-scanning protocol (low resolution overview image, coarse Z-stack, short time lapse). The overview image is then quantified by a dedicated image-analysis routine that will look for some ROIs (e.g. cells, tissues) or evaluate the quality of the image. Finally depending on the outcome of the analysis (the feedback), a new sequence of events can be executed (displacement of the objective, high-resolution acquisition, adjustment of the illumination). This procedure can then be repeated for multiple samples e.g. for the automated imaging of a full well plate (Pandey et al. 2019; Peravali et al. 2011; Spomer et al. 2012), or iteratively for the same specimen to adapt the acquisition over time (Wang et al. 2014). A widespread example of feedback microscopy is the autofocus, traditionally available on most commercial microscopes.

Feedback microscopy has been previously used in life science for the targeted imaging of ROIs at high-resolution, previously localized from low-resolution images. This approach was applied for the targeted imaging of HeLa cells, individually localized by intensity-thresholding of the overview images, and subsequently imaged at higher-resolution (Gunkel et al. 2017) or by super-resolution imaging using single-molecule localization microscopy (Eberle et al. 2017). (Conrad et al. 2011) used a similar approach completed by image-classification for the identification and exclusive imaging of HeLa cells in a specific mitotic stage, which represents a small fraction of the cell population. Besides cell cultures, organoids (Lukonin et al. 2020) and small organisms such as zebrafish larvae have also been imaged at high-resolution with high-throughput, thanks to feedback microscopy with such a prescan/rescan strategy. (Peravali et al. 2011) thus demonstrated the imaging of the zebrafish head region detected by template matching, (Pandey et al. 2019) described the high-resolution acquisition of the fluorescently-labelled kidney in the *Tg(wt1b:EGFP)* transgenic line, localized by centre of mass, while (Spomer et al. 2012) used intensity-based thresholding to identify and image the heart-region.

The targeted imaging of previously localized regions of interest is one application of feedback microscopy. Another promising application is adaptive microscopy, for which the feedback provides information about the sample in order to optimize the acquisition parameters (e.g. exposure time, illumination intensity) or to perform a specific acquisition sequence. This approach was used to correct optical aberrations in real time using adaptive optics (Wang et al. 2014), while (Rabut and Ellenberg 2004) kept migrating cells in focus and in the field of view of the objective for time-lapse acquisition, by tracking cells and correcting the microscope stage position accordingly. Finally, in addition to imaging, feedback microscopy has also been demonstrated for active photomanipulation such as optogenetics, laser ablation, photobleaching or photoactivation. (Leifer et al. 2011; Stirman et al. 2011) thus reported the targeted activation of neurons by optogenetic in freely moving *Caenorhabditis elegans*, while (Conrad et al. 2011) performed FRAP (Fluorescence Recovery After Photobleaching) experiments in the nuclei of CBX1-EGFP cells previously identified as being in interphase or prophase by the feedback routine.

While feedback microscopy has the potential to automate complex acquisition protocols, which would otherwise require time-consuming manual interventions, the analysis routine used to evaluate the images and to return the feedback is a critical component for the success of the procedure. It must be robust to the variability of the sample, while still being specific to the expected phenotype. Besides, in addition to a robust image-analysis pipeline, feedback microscopy requires a controllable microscope setup, which typically involves hardware interfacing and custom script development to allow the communication between the image-analysis and the microscope control. Implementing feedback microscopy on a setup thus requires technical skills which are not at the core of life-science research. Although commercial and open source solutions exist to enable smart microscopy applications (ACQUIFER 2017a; Conrad et al. 2011; Edelstein et al. 2010; PYME contributors 2020; Rhode 2015; Schorb and Sieckmann 2017), no standard protocols currently exist.

1.4.2. Applications to small-organisms microscopy screening

One practical application of feedback microscopy is the targeted high-magnification imaging of ROIs, identified from lower magnification images. While the resolution increases with the magnification of the objective, the size of the field of view is inversely reduced. For samples like dense cell cultures which are usually homogeneously spread over the field of view, this is not an issue, except that a fewer number of cells can be imaged within the smaller field of view of a high-magnification objective. In this case, the high-magnification imaging is usually performed by selecting random image patches within the low-magnification image. Thus, feedback microscopy is usually not necessary with dense cell cultures, unless a specific cellular phenotype or event is researched, e.g. a specific mitotic stage (Conrad et al. 2011).

For more “localized” samples (e.g. low-density cell cultures, organoids, microscopic and larger organisms), which only occupy a small fraction of the field of view at low magnification, automated high-magnification imaging without feedback microscopy is more challenging. To some extent, it might still be possible when the positions of the samples are known a priori. This can be achieved thanks to adapted sample mounting, that helps positioning the sample in a specific position, like the centre of a well in a 96-well plate (Alessandri et al. 2017; Popova et al. 2018; Wittbrodt et al. 2014). For a large majority of these mounting systems though, the precision of the positioning will still be in the order of a few hundred micrometres. At the scale of a high-magnification objective, this can represent a major deviation of the position by the equivalent of several fields of view. Therefore, even with regular sample positioning, high-resolution imaging following the expected sample coordinates provided by the mounting system will likely result in missing the ROI in most cases (Figure 5). Using feedback microscopy instead, the positions of the ROIs are extracted from low-magnification images thanks to a dedicated pattern recognition algorithm. The algorithm yields the positions of the ROI with a precision of few low-resolution pixels, which allows more accurate positioning for high-magnification imaging. It can also improve the reproducibility of the imaging as the detected regions are likely to occupy a similar position in the field of view for the successive samples, which effectively spares the need for additional alignment of the imaged regions in post-processing.

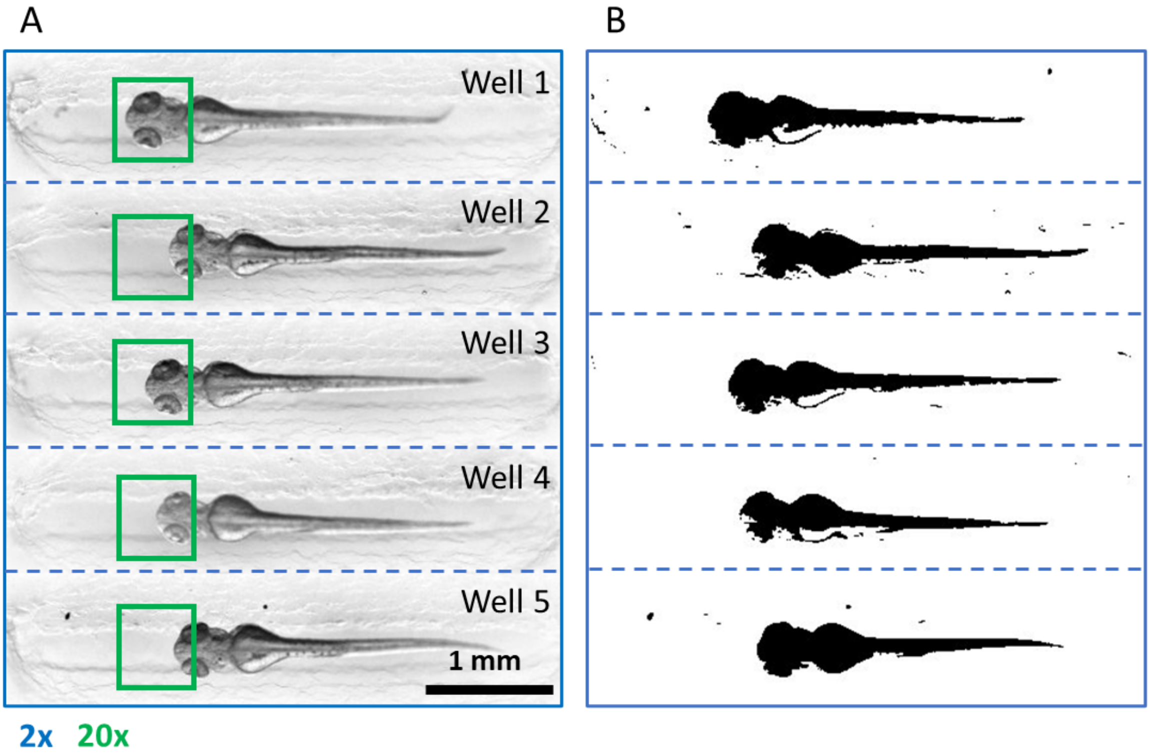


Figure 5: Slight positional shifts prevent the automated imaging of regions of interest

A) View of zebrafish larvae mounted in wells of a 96-well plate, using pre-formed agarose cavities for the reliable dorsal orientation of the specimen (Wittbrodt et al. 2014). Despite the consistent positioning of the specimen in the wells, a slight positional difference of less than a millimetre prevents the automated high-resolution imaging of the region of interest when identical coordinates are used. The images outlined in blue are cropped from the original full well images, according to a rectangular region positioned in the centre of the well. The green rectangle corresponds to the expected size of the field of view for a 20X high-magnification objective, and is positioned at fixed coordinates relative to the centre of the wells. **B)** Using intensity-thresholding allows isolating the full specimens but remains challenging to localize a specific organ-region (e.g. the head region).

1.4.3. “On-the-fly” processing

Image-processing can also be coupled to the imaging to perform analysis “on the fly”: while the acquisition is still running. This saves time especially for computationally expensive workflows like deconvolution, or if some image-reconstruction is needed (structured illumination, super-resolution etc.). It can also be used to reduce the size of the dataset, by automatically identifying the interesting fractions of the dataset e.g. by cropping ROI or isolating interesting time points (Pandey et al. 2019; Scherf and Huisken 2015).

1.5. Introduction to object-detection

1.5.1. Applications of object-detectors

Object-detectors originate mostly from the field of computer vision for use-cases such as face and pedestrian recognition (Dalal and Triggs 2005; Rowley et al. 1998), character recognition (Y. Lecun et al. 1998) or autonomous driving (Grigorescu et al. 2020). With the availability of large annotated datasets representing various object-classes and associated challenges for object-detection and classification (Deng et al. 2009; Everingham et al. 2010; Yann Lecun et al. 1998; Russakovsky et al. 2015), the field gained major interest and keeps drawing attention to develop better and faster algorithms. Some of those algorithms have been re-purposed for life science applications, such as the classification of macromolecular crystals (Bruno et al. 2018), cells and nuclei detection or segmentation (Han et al. 2012; Ronneberger et al. 2015; Waithe et al. 2020), plankton detection and classification (Luo et al. 2018) or identification of breast cancer or microcalcification in mammography images (Dembrower et al. 2020; El-Naqa et al. 2002). Object detectors can be applied for object-tracking too, by running the detection on the successive frames of a video. Many of these algorithms were optimised for this purpose to run the detection in real-time, which is a critical requirement for dynamic applications e.g. autonomous driving. The latest innovations in the field now aim at analysing not only a single image but short video sequences, to recognize specific behaviour or actions (Mobahi et al. 2009; Wu et al. 2015).

1.5.2. Distinction between localization and recognition

By object-detection, one usually implies both the localization of objects within an image and the recognition of the objects' categories. Therefore, object-detectors typically combine 2 relatively independent mechanisms: one for the proposal of possible object-regions, a second for their consecutive recognition. Classical strategies for the generation of candidate image-regions include intensity-based thresholding (Neumann et al. 2006; Spomer et al. 2012), brute-force search with rectangular sliding windows (Gallego et al. 2018; Han et al. 2012; Vaillant 1994; Viola and Jones 2004), dedicated region proposal algorithms (Girshick et al. 2016; Uijlings et al. 2013; Zitnick and Dollár 2014) or bounding boxes (anchors) randomly distributed within the image (Liu et al. 2016; Redmon et al. 2016). To identify genuine image-regions from the pool of candidates, the recognition process typically relies on a set of descriptive image-features for each region. The features for a given image-region are then evaluated by a decision mechanism, responsible for the distinction between actual vs erroneous object-regions or the assignment to a specific object-category. Classifier algorithms are routinely used for the decision task, as they can handle multiple input features and output object-classes. Object-locations are ultimately given by the position of the image-regions classified as containing the object.

1.5.3. Classifiers in computer vision

Classifiers are mathematical models used for the prediction of sample categories, provided a set of descriptive sample features (see the classification of *iris* flowers species based on morphological measurements (Fisher 1936)). A well-trained classifier should assign samples with similar features to the same category. Classifiers have multiple applications in computer vision such as image-classification (Conrad et al. 2011; Kumar et al. 2015; Neumann et al. 2006; Shang et al. 2019) and pixel-classification. Pixel-classifiers are mainly used for segmentation (Arganda-Carreras et al. 2017; Berg et al. 2019) and key points detection (Stern et al. 2011a). For image-classification, a sample is an individual image, or for object-detection a subregion of an image (patch), while for pixel-classification a sample is a single pixel of an image. In both cases, there are at least 2 classification outputs, e.g. object vs background for object-detection, foreground vs background for binary pixel classification. With more than 2 categories, multiple image classes can be predicted, while for pixel-classifiers segmentation of an image in various domains is possible.

A classical workflow for image-classification with 2 categories (binary) is illustrated in Figure 6. First, image-instances are turned into a more succinct set of descriptive features. The features are then evaluated by a pre-trained classifier that will return a probability for the image-instance to belong to the expected category (positive). Finally, depending on a user-defined threshold on the probability, the image-instances are separated between positive and negative. The probability threshold allows tuning the detection selectivity for a given classifier: a high threshold yields a selective classification, with almost no false positive detection but with a higher risk to miss actual positive instances (false negative). Inversely decreasing the threshold allows a more permissive detection process (Figure 7A, (Chou 2019)). In the context of object-detection, a false positive detection is any image-region erroneously reported as containing the object of interest. Similarly, a false negative is an actual object not detected. A true positive is an object correctly detected and a true negative is any background region correctly undetected (Figure 8). With more than 2 classification outputs i.e. not only positive/negative, the principle is identical except that the classifier will return for each image-instance, the probability associated with each category. The probabilities over the different categories sum up to 1, such that a given image-instance is normally assigned to a single category, which obtained the highest probability. For microscopy applications, binary classification was previously used for the localization of cells and nuclei by classifying image-patches as belonging to the object-class or to the background, based on edge features (Han et al. 2012) or similarly for the identification of glomerulus in kidney tissue segments using features extracted by a convolutional neural network (Gallego et al. 2018). Multi-class classifiers have been used to identify the mitotic state of cells, based on intensity and shape features after segmentation (Conrad et al. 2011; Neumann et al. 2006).

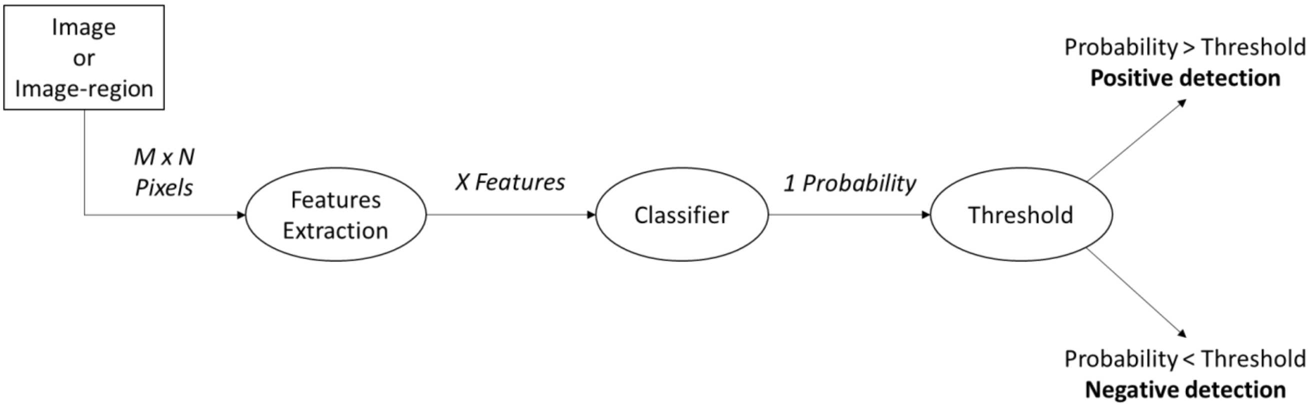
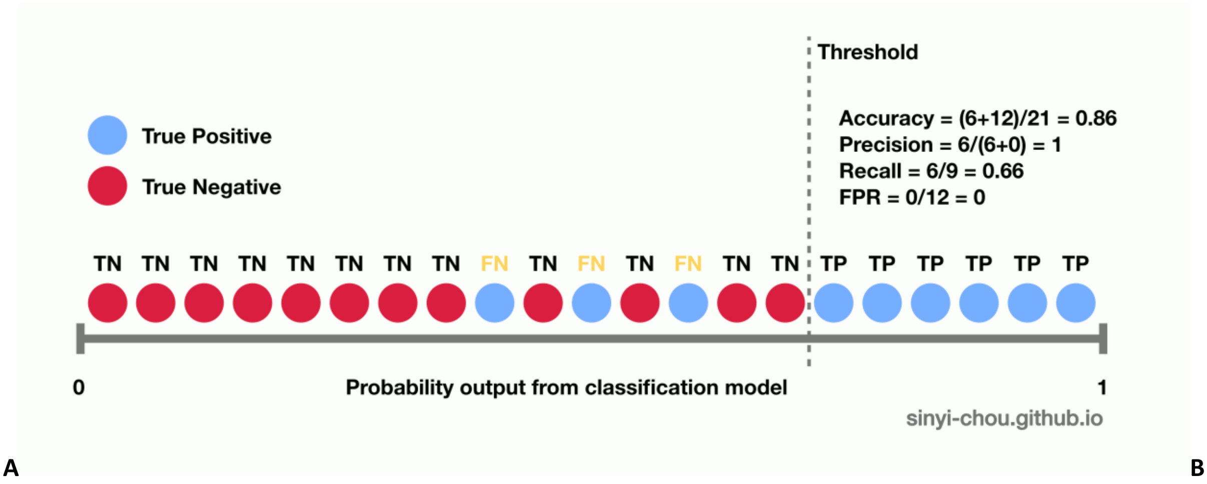


Figure 6: General scheme for binary image-classification

First the image to classify is described with a few key figures, the image-features. The number of features (X) describing the image is typically much smaller than the initial image size (N x M) in order to condense the information and facilitate the classification. The features are then passed to a pre-trained classifier, calculating the probability for the image to belong to the expected category (positive), based on the extracted features. In the last step, the dataset is split between positive and negative images according to a user-defined threshold on the probability. For a multi-class classifier (more than 2 outputs), the scheme is identical except that the classifier will return a set of probabilities distributed over the different output categories, such that they sum up to 1. (Own contribution, reproduction allowed under a CC-BY license).


A
Threshold
Accuracy = (6+12)/21 = 0.86
Precision = 6/(6+0) = 1
Recall = 6/9 = 0.66
FPR = 0/12 = 0
True Positive
True Negative
TN TN TN TN TN TN TN TN FN TN FN TN FN TN TN TP TP TP TP TP TP
0
Probability output from classification model
1
sinyi-chou.github.io

B

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

Figure 7: Tuning the classification threshold for a binary classifier

A) Each dot represents a sample belonging to the positive group (blue) or negative group (red). The samples are ordered by probability to be positive, as returned by a binary-classifier discriminating between positive/negative (blue/red) samples. Because classifiers are not perfect, the predicted probability for some positive samples is lower than for some negative and vice-versa. Given this classifier, the selectivity of the classification can be tuned by adjusting the threshold on the probability (dashed line). Samples on the right side of the threshold are classified as positive, and as negative on the left side. Reproduced with permission from (Chou 2019). **B)** Expression of the True Positive Rate (TPR), which is the fraction of positive samples (P) correctly classified (the True Positive - TP), and False Positive Rate (FPR) which is the fraction of negative samples (N) erroneously classified as Positive (False Positive - FP). FN: False Negative, TN: True Negative.

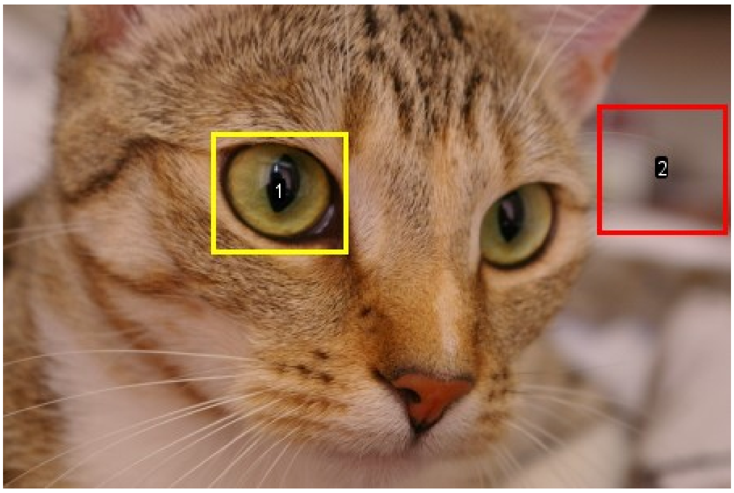


Figure 8: False positive and false negative with object-detectors

Above are some hypothetical localizations from a detector supposed to identify the eyes of the cat. The eye on the left is correctly detected, it is a true positive detection (bounding box 1 in yellow). The eye on the right is not detected, it is a false negative detection. Finally, the bounding box 2 in red is incorrectly reported by the detector, it is a false positive detection. All other image regions are true-negative, they belong to the background class and were indeed not reported by the detector. Example image “Chelsea the cat” available via the scikit-image library (skimage.data.chelsea), public domain (CC0 license).

1.5.4. Non-Maxima Suppression (NMS)

A given object is likely to be predicted multiple times by overlapping bounding-boxes or similar shapes used to outline the detected objects. To keep a single optimal detection per object, non-maxima suppression (NMS) is typically performed after detection. The degree of overlap between neighbouring detections is often used as a criterium for this procedure, and is calculated as the ratio of the intersection and union area (intersection over union - IoU) between the rectangular bounding-boxes (Alexe et al. 2012; Felzenszwalb et al. 2010; NG 2018; Rosebrock 2014b; Rothe et al. 2015) or circles (Yang et al. 2020) predicting the object-locations. Highly overlapping detections most likely depict the location of the same object, while bounding boxes with no or small overlap correspond respectively to distant objects or to neighbouring objects. This ratio is equal to 1 if the bounding boxes fully overlap and 0 if they do not overlap at all. Overlap-based NMS works by setting a threshold on the IoU, which is equivalent to choosing a maximum degree of overlap allowed between pairs of bounding-boxes.

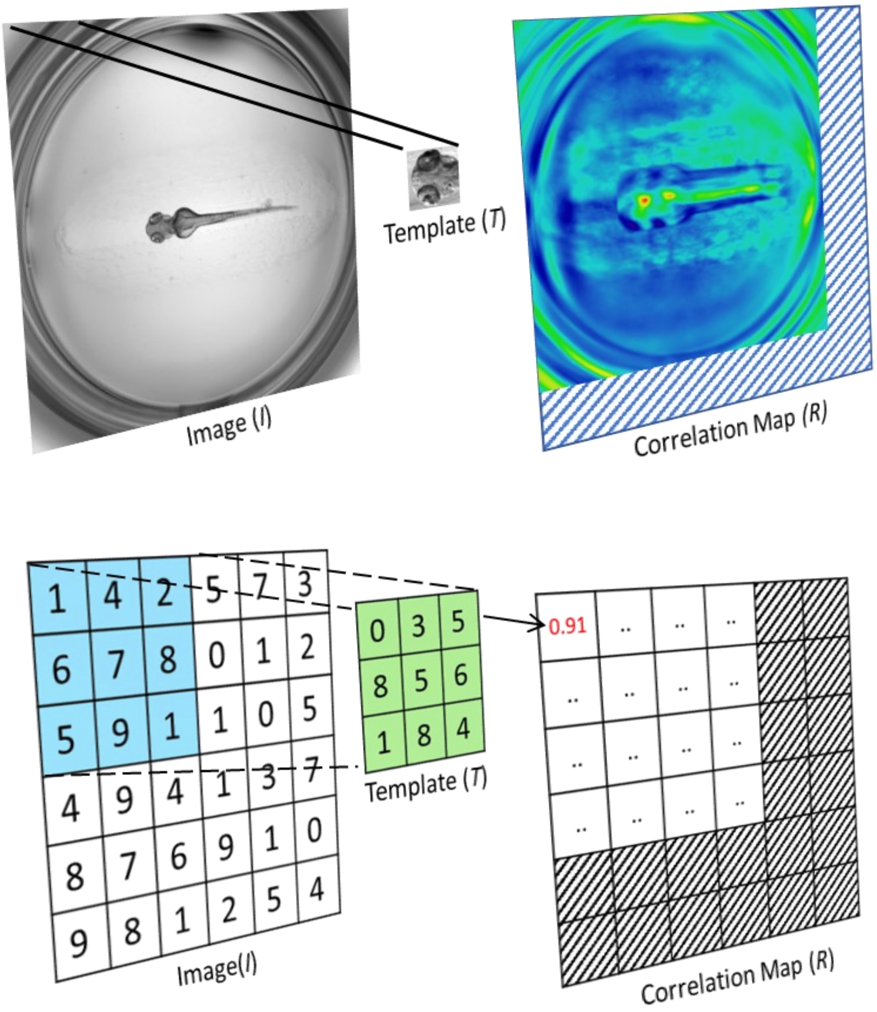
1.6. Object-detection with template matching

1.6.1. Principle

Template matching is a relatively intuitive algorithm, relying on the previously mentioned sliding-window search (see section 1.5.2). In this case, a small template image representing the object of interest is used as a sliding window, translated over the image in which the objects should be localized. For successive positions of the sliding-window (X,Y corresponding to the position of its top left corner in the coordinates of the image), the algorithm compares the pixel values between corresponding pixels of the current image patch and template image. The resulting correlation (or difference) score for this X,Y-position of the sliding window is stored in the pixel at position X,Y in a new image matrix called score map. The computation of the score map is illustrated in Figure 9, for the localization of the head-region in zebrafish larvae. Once the sliding window search completed, the most probable object-positions are given by the coordinates of the extrema in the score map (local minima if a difference score is used, local maxima if a correlation score is used). The associated object-locations are predicted for each extremum as a bounding-box of dimensions identical to the template image, with its top left corner positioned at the coordinates of the extrema. The detection score is provided by the value of the extrema. In a way, template matching is thus a binary object-detector (object vs background) relying on a sliding-window with a single feature: the similarity of pixel values with the template image. A threshold on this similarity score can be used to tune the detection selectivity.

1.6.2. Previous applications

Template matching has been previously used for the automated inspection of manufactured parts (e.g. printed circuit boards (Crispin and Rankov 2007)), for the detection of fruits and vegetables (Bao et al. 2016; Oke et al. 1995). It has also been used in life-sciences for cell detection (including the use of artificial template images (Young et al. 1998)), for the localization of specific geometric shapes in gels (Tseng et al. 2011), for head detection in zebrafish larvae (Peravali et al. 2011) or more recently for the localization of unhatched zebrafish embryos in well plates (Shang et al. 2019). (Cole et al. 2004) also proposed a complex object-recognition pipeline using multiple template images, illustrated with the recognition of various LEGO blocks.



$$CorrMap(x,y) = \frac{Template\ .ImagePatch\ at\ (x,y)}{\sqrt{\sum_{x',y'} Template^2 . \sum_{x',y'} ImagePatch\ at\ (x,y)^2}}$$

$$CorrMap(x,y) \ = \frac{\sum_{x'y'} T(x',y').I(x+x',y+y')}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}}$$

$$CorrMap(0{,}0) = \frac{1*0\ +\ 4*3\ +\ 2*5+6*8\ +\ 7*5\ +\ 8*6\ +5*1\ +\ 9*8\ +\ 1*4}{\sqrt{(1^2+4^2+2^2+6^2+7^2+8^2+5^2+9^2+1^2)*(0^2+3^2+5^2+8^2+5^2+6^2+1^2+8^2+4^2)}} = 0.91$$

Figure 9: Computation of the score map using template matching

A small template image (T) is used as a sliding search window translated over the target image (I). For every position (x, y) of the template image (position of the top left pixel), the intensity correlation between the overlapping pixels of the template (green) and the image (blue) is computed. This yields a score between 0 and 1, stored in the corresponding pixel at position (x,y) of the score map (R). The sum over x’,y’ corresponds to the sum over the pixels of the template. T(x’, y’) and I(x’, y’) correspond to the pixel value of respectively the template and image-patch at position x’, y’. Own contribution – Redistribution allowed under a CC-BY license.

1.6.3. Advantages and limitations of single-template matching

Template matching is a relatively simple object-detection solution, easy to understand and to experiment with. It has almost no parameters (the score threshold) and does not require any image pre-processing. Besides, using a normalised score for the computation of the score map, the search is robust to variations of illumination between the template and the image. Despite this simple principle and a history of successful applications, template matching is not widely used in the bioimage analysis community. One reason might be the lack of flexible implementation in popular image-analysis software. The implementations in image-analysis libraries such as OpenCV (OpenCV contributors 2000) or Scikit-image (van der Walt et al. 2014) are limited to the computation of the score map, while the rest of the detection pipeline is left to the user. Other limiting factors might be the intrinsic limitations of a template matching search with a single template. The first limitation, which is common to other sliding window approaches, is that a given object might be detected multiple times at slightly shifted locations. This can be explained by the probability to have multiple local extrema in a neighbourhood of the score map. Non-maxima suppression (NMS) could be applied to yield a single detection per object but is typically not available with existing implementations of template matching. The second limitation, specific to template matching, is that the sliding window search uses only the fixed intensity pattern of the provided template image. Therefore, if the objects of interest show some variability in the images (change of perspective, rotation, morphological deformation), single-template matching will likely fail to localize all objects. A solution is to repeat the search with multiple template images representing the expected object states (Bao et al. 2016; Cole et al. 2004; Young et al. 1998). Again, a given object might be detected multiple times, but this time also by different templates, and thus NMS should also be applied here to assure a single optimal detection per object. The resulting object-detections are thus not a simple sum of the individual template searches. Although such a multi-template search is promising, no widely available implementation was previously available.

1.7. Object-detection with cascade detectors

1.7.1. Introduction

Cascade detectors (also called cascade classifiers) were first reported by (Viola and Jones 2001) as fast object-detectors for real-time face detection in grayscale images. Cascade detectors belong to the machine-learning detectors using a sliding window search and recognizing a single object-category (e.g. human faces). There are thus binary detectors (object vs background). While mostly applied to face and pedestrian detection, cascade detectors were also previously used for the localization of biological objects in tissues, including cell nuclei (Han et al. 2008), Chagas parasites (Uc-Cetina et al. 2015) or white blood cells (Maulana Budiman et al. 2016). Nowadays, cascade detectors tend to be overcome by deep-learning detectors, however they are still a widely available alternative for object-detection.

1.7.2. Principle for the detection

There are several components in cascade detectors, which were explained individually in more details in section 1.5.2. First those detectors rely on a sliding window search for the localization of objects. Then to describe image patches at every location of the sliding window, specific image-features are used: the so-called Haar-like features, representing patterns of intensity as occurring within an image-patch as in Figure 10.A. Finally, the discrimination between image-patches containing the object and other non-object patches is performed by a succession of binary decision-tree classifiers. The decision trees are arranged successively (hence the name cascade) such that the next tree in the cascade evaluates the positive detections from the previous tree but with more features, to refine the classification by eliminating false positive detections from the previous cascade stages (Figure 11).

Cascade detectors showed improved detection speed compared to previous sliding-window object detectors thanks to two novel subtleties. The first one is the use of Haar-like patterns for the recognition of objects and the use of integral images for their efficient computation. The second innovation is the cascading scheme which limits the computation of features, and thus the computation time for obvious non-object regions, which are typically a majority of all the image-regions the sliding-window search will evaluate. Those regions get quickly discarded by the first stages of the cascade, while only image-regions with a high probability to contain an object manage to pass the cascade of classifiers, and to eventually get classified as an object-region.

In a cascade, every stage is supposed to virtually return all object-regions (high True Positive rate), while still discarding a good amount of non-object regions (True Negatives), using only few features to assure a rapid classification. Therefore, a low probability threshold is typically used for the binary classification at every stage, which often translates to a high False Positive rate (see section 1.5.3). Yet, the cascading scheme genuinely compensates for this apparent weakness. Indeed, the overall cascade false positive rate is equivalent to the product of the individual classifiers' false positive rate (Viola and Jones 2001). Therefore, the more stages in the cascade, the lower the overall false positive rate of the detector. On the other hand, the rule is also valid for the overall true positive rate[1]: it decreases with the number of stages, as with every classification round a few object-regions might get erroneously discarded (False Negative). Therefore, the individual classifiers should have a high true positive rate. Typical values for the individual classifiers are 0.99 for the True Positive rate and 0.5 for the False Positive rate. This means that for every stage, almost all genuine object-regions should be reported as positive (TP), while about a half of the non-object regions passing that stage will erroneously be classified as containing an object (FP), and the other half correctly discarded (TN).

1.7.3. Haar-like features

Haar-like features represent greyscale intensity patterns occurring within an image-patch as in Figure 10.A. Using those features, (Viola and Jones 2001) reported the robust detection of faces for white individuals with only 2 features as shown in Figure 10.B, reflecting natural intensity patterns for white faces. The major advantage of such features is their rapid computation using integral images, which further contributes to the rapidity of the detection process. Although the patterns apparently provide coarse image-representations, evaluating several of them for a given image-patch allows the representation of complex intensity distributions representing objects, at a still low computational cost. For a given image-patch, there are various Haar-features corresponding to different patterns, but also to different locations or scales of a pattern within the detection window. The subset of patterns relevant for the detection of a particular object is selected during the training of the cascade detector. Haar-like features provide an additional advantage for the detections of objects at multiple scales. Indeed, the patterns can be rescaled to cover the detection of objects of various sizes, which is more efficient than generating an image-pyramid and running a fixed-size detector for each level of the pyramid. Nowadays, the default solution to train and use cascade detectors is via the OpenCV library (OpenCV contributors 2000; OpenCV contributors 2018). However, contrary to the original publication this implementation still relies on image-pyramids for multiscale detection.

[1] Denoted as Detection rate in the original publication

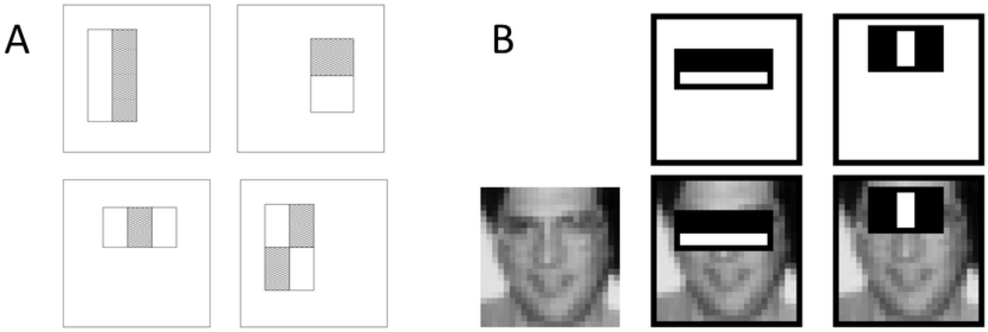


Figure 10: Example of Haar-like features for a given detection window

A) Haar-like features are depicted as patterns of grey and white rectangles, illustrating possible intensity patterns occurring within a detection window (represented by the outer square). For each pattern, the feature value is calculated in 2-step. First the sum of the pixel intensities for the image-region underlying the grey and white regions are calculated separately. Then the feature value is given by the difference between the sum for the grey and white region. B) First 2 features retained during the training of a cascade classifier for face-detection, overlaid on an example face-image at the resolution used for detection. The features reflect the natural intensity pattern occurring for white faces: the eye-region is darker than the nose-region both along the horizontal and vertical. Figures from (Viola and Jones 2001).

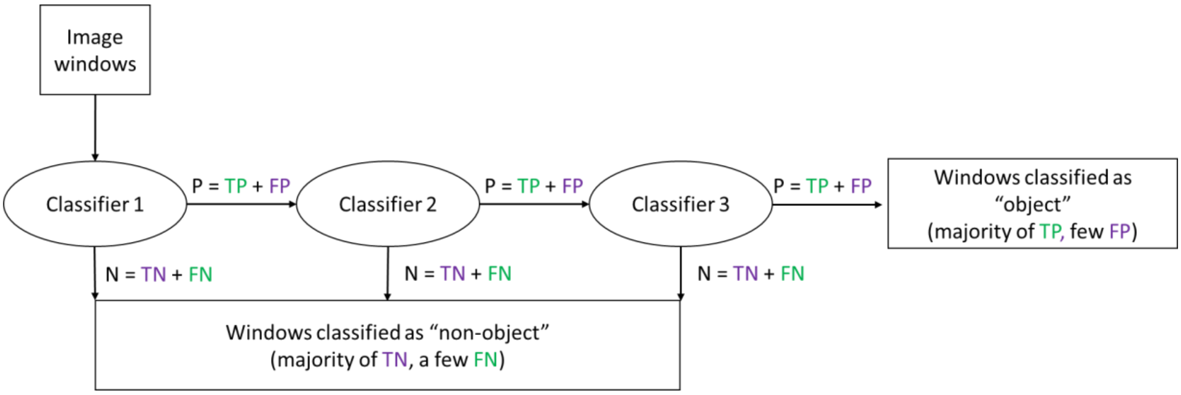


Image windows
Classifier 1
P = TP + FP
Classifier 2
P = TP + FP
Classifier 3
P = TP + FP
Windows classified as "object" (majority of TP, few FP)
N = TN + FN
N = TN + FN
N = TN + FN
Windows classified as "non-object" (majority of TN, a few FN)

Figure 11: Classification with a trained cascade classifier

Here the cascade contains 3 classifiers (or stages). A candidate image-region as generated by the sliding-window enters the cascade on the left. For every stage, Haar-like features are computed and evaluated by the corresponding classifier. If the image-region is classified as potentially containing the object of interest, it is passed to the next classifier for finer evaluation, otherwise it is discarded, and the next image-region enters the cascade. Abbreviations: P: classified as Positive - N: classified as Negative – TP: True Positive – FP: False Positive, TN: True Negative, FN: False Negative. (Own contribution).

1.7.4. Training a cascade detector

Like any machine-learning method, cascade detectors are trained using a collection of labelled image-examples (training set). With cascade detectors, the localization is assured by a sliding-window and thus is not part of the model prediction, only the class (object/non-object) of the image-patches is predicted. Therefore, the training set is ultimately composed of cropped images representing the object for the positive instances, and cropped images representative of a range of non-object image-regions for the negative instances. Before the actual training of the model, the training image-instances are downscaled to a small size defined as a parameter of the training and corresponding to the dimensions of the detection window. The size of the detection window determines the total number of Haar-features available, for instance a classical 24 x 24 pixels window as in Figure 10.B, offers 162 336 Haar-like features corresponding to different intensity-patterns, and multiple shifted positions of a given pattern within the detection window. Like for the training images, the bounding-boxes predicting object-regions always have an aspect-ratio identical to the detection window used for training, however the bounding-boxes dimensions can cover a range of scales. For example, a cascade trained with a detection window of 30 x 10 pixels (width x height) will always predict object-location with bounding-boxes of 3:1 aspect-ratio. The window-size can be adapted for any aspect-ratio or to account for objects which would not be distinguishable if downscaled to a few dozen pixels. It is however advised to keep a small detection window to limit the number of features available and thus facilitate the training process.

For cascade classifiers, the training always generates a new model architecture *de-novo*, such that the number of decision trees forming the cascade, their structure and the features used for the classification depend on the training set, and on the parameters used for the training. The training of a cascade is an iterative process in which the stage classifiers are trained consecutively, until each of them reaches the minimum requirements defined as parameters of the training (minimum True Positive rate and maximum False Positive rate[2]). To improve these rates with minimal computational cost for the detection, optimal features selected using a variant of the AdaBoost learning strategy (Viola and Jones 2001) are progressively added for each stage. Once a stage classifier reaches the minimal requirement, its training stops, and a new classification stage is added on top of the current one, hence extending the cascade. The only difference for classifiers further in the cascade is that the available training set is exclusively composed of image-instances classified as positive by the classifiers of the previous stages. This “reduced” training set contains both ground-truth positives image-

[2] Respectively denoted as minHitRate and maxFalseAlarmRate in OpenCV

instances (true positive detections from the previous classifiers) and ground-truth negative image-instances (the false positive detections from the previous classifiers). Meanwhile, all image-instances previously predicted as negative are never evaluated by the following classifiers. Therefore, as more classifiers are appended to the cascade, the training set reduces in size, in particularly for the negative ground-truth instances as every new classifier tends to eliminate a large fraction of them. The number of positive ground-truth instances in the training set also decreases along the training as some positive instances get misclassified as negative, but more slowly than the negative fraction. Therefore, the number of negative ground-truth images in the training set is usually higher than the number of positive images (e.g. 2 to 5 folds).

The training and thus the growth of the cascade stops either when the “available” training set runs out of ground-truth images (most likely negative ones), when the overall false positive rate for the cascade reaches the minimum rate defined in the training parameters, or when the overall true positive rate for the cascade drops below the minimum rate, meaning that adding more stages will be detrimental to the detection of actual objects.

Finally, provided the individual false positive and true positive rates for the individual classifiers as well as the number of stages, the overall cascade rate, and the number of ground-truth positive and negative images available at every stage can be estimated. This can be used to evaluate if the size of the training set is sufficient. An excel sheet performing those estimations is provided as supplementary data to this thesis (Thomas 2020a).

1.8. Object-detection with deep-learning detectors

1.8.1. Deep-learning models

Deep-learning belongs to machine-learning methods and have recently gained a new interest for its powerful capacities when applied to speech-recognition, image classification and other data-science tasks. Deep learning designates mostly 2 types of models: fully connected neural networks (also called dense neural networks or multi-layer perceptrons) and convolutional neural networks (CNN).

Fully connected neural networks have been used as other machine-learning methods to perform classification or regression tasks, provided 1-dimensional feature vectors (Lippmann 1989; Ou and Murphey 2007). A fully-connected network is made of stacked layers (hence the name deep) each containing multiple units called neurons, receiving as inputs, the outputs of some or all neurons of the previous layer (Figure 12). Each neuron performs a simple linear combination of its input values, with different weights (Wn in Figure 13) and a global offset. The result of this linear combination then goes through a non-linearity function (also called activation function) which finally yields the output of the neuron (Figure 13). The weights and offset for each neuron are adapted as part of the overall model training. With multiple layers and multiple neurons in each layers, such fully connected networks have multiple parameters and connections that give them the capacity to account for complex mathematical relations between the input of the network (typically a list of feature numbers describing a sample) and the output of the network (a category index or quantitative value). Artificial neural networks are thus particularly powerful for tasks such as classification or mathematical regression (Lakhmi Prasanna et al. 2017).

With images, the pixel values are first turned into descriptive features to be classified by fully connected neural networks (unless the images are sufficiently small – see (Y. Lecun et al. 1998)). This is not the case anymore for the second type of deep-learning models, the Convolutional Neural Network (CNN), which directly take 2D images as input. The novelty and strength of those modern models is the introduction of convolutional layers, replacing individual neurons, which have the capacity to derive optimal recognition features directly from the images. These layers perform convolution operations, between an input array of pixels, and a layer-specific kernel whose coefficients are adjusted during the training (Bouvrie 2006; Moen et al. 2019)(Figure 14). The various kernels composing the model are the actual features derived from the training data, and correspond to visual patterns, used to describe image-content. The input array that is convolved with those kernels is either the input image for the first layer, or the so-called feature map resulting from the convolution operation of the previous layer (Gu et al. 2017). Again, by stacking multiple layers, the model can distinguish multiple patterns and combinations of those, which in turns allow the recognition of various object classes. Finally, while the convolutional layers are used to extract features, the last layers of such CNN models are almost always fully connected layers like the fully connected models introduced before. Those layers take as input a long 1-dimensional feature vector, resulting from the consecutive convolution operations and an ultimate flattening operation of the final feature volume (Figure 14). The fully connected layers process this long feature vector to yield the output of the network, typically a class index or some quantitative values (probabilities, etc.). In those models, the succession of convolution operations is used to condensate the original image information into a feature vector, suitable for classification or regression tasks. These models were initially demonstrated for character recognition (Y. Lecun et al. 1998) and later were expanded to classification of images from real-life scenes in 1000 different classes (Simonyan and Zisserman 2015).

Other CNN architectures exist for instance to regenerate an image from the feature vector, such that the output of the network is not a class or a set of values but another image (Moen et al. 2019). Those models have applications for segmentation (Caicedo et al. 2019; Falk et al. 2018; Ronneberger et al. 2015), image-restoration (Weigert et al. 2018), denoising (Krull et al. 2019) or artistic style transfer (Gatys et al. 2016). The model architecture is often depicted by a characteristic U-shape or hour-glass shape (Newell et al. 2016; Ronneberger et al. 2015), representing the encoding of the original image into a more succinct array of features, then decoded to a new image (also called encoder-decoder architectures).

A major advantage of CNN models compared to previous classifiers is that they can learn optimal features for a given prediction task directly from the training dataset. They thus have a high degree of adaptability, contrary to manually selected feature sets which might not perform equally well on new datasets or different prediction tasks. For any machine-learning task related to images or sounds, CNN models have thus often become the default solution. However, training such models from scratch require a large amount of ground-truth annotated data, powerful hardware like graphic cards, some programming knowledge and time. Using transfer learning, it is possible to limit this burden by retraining a pre-trained network, to perform a task similar to the one it was originally trained for (e.g. classification) but for a different dataset (Moen et al. 2019). By doing so, the features learned on the original dataset are reused and eventually adapted to the new training set. This effectively reduces both the training time and the number of ground-truth data required to train a model.

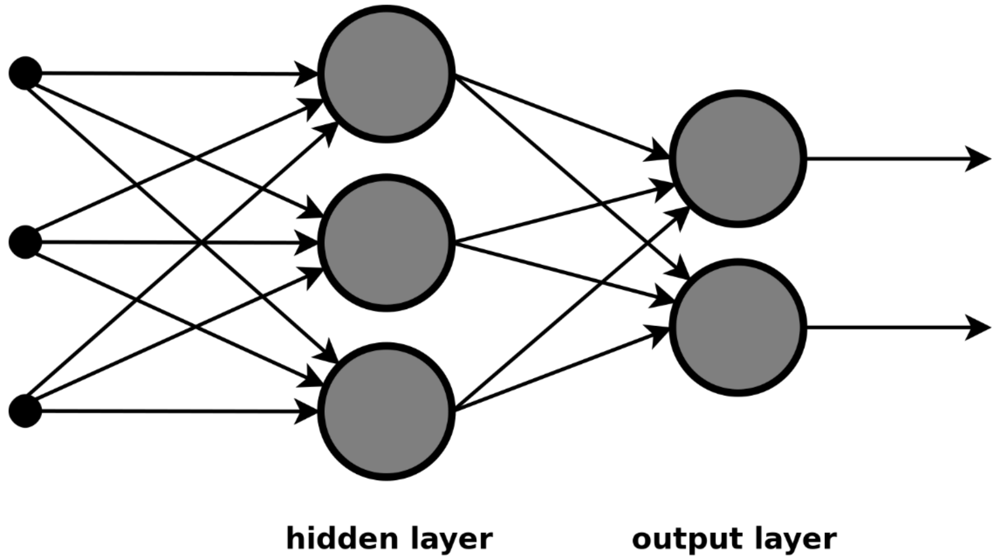


Figure 12: Scheme of a simple fully connected neural network

On the left are 3 inputs features. The hidden layer contains 3 neurons and the output layer 2 neurons. The network has thus 2 outputs, which can correspond for a classification task to the probabilities for 2 classes. With a fully connected network, all neurons of a layer receive inputs from all units of the previous layer. Figure under CC-BY-SA license from (Burgmer 2018).

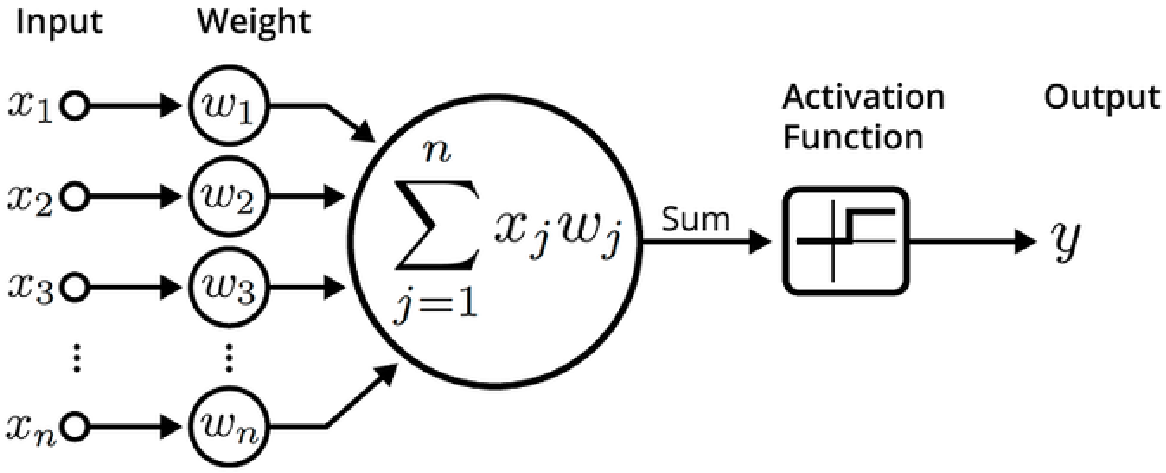


Figure 13: Scheme of single neuron, also called perceptron

The neuron performs a linear combination (weighted sum) of the inputs (x1,…xn) using variable weights (Wn) and a bias (not represented here). The result of the sum then goes through a so-called activation function for normalisation, which yields the output of the neuron y. The weights Wn and bias are adapted during the training to find the optimal combination, that would best predict the output provided the input features x. Figure from (Pirvu 2019) - public domain.

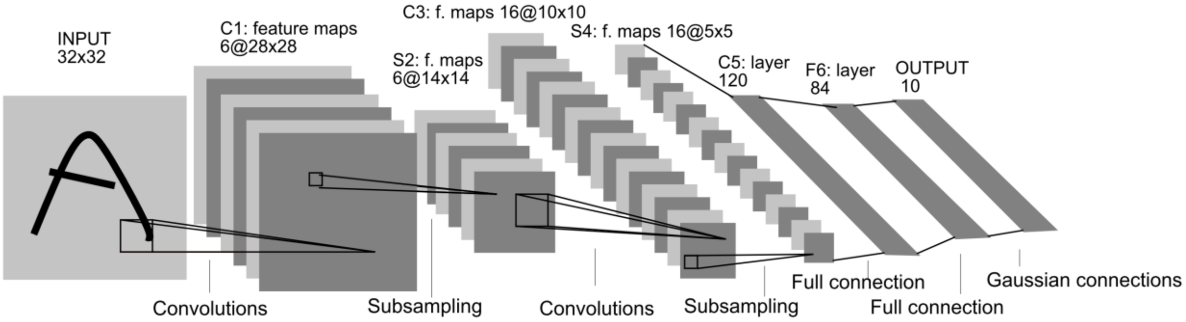


Figure 14: Architecture of LeNet-5, one of the first CNN model published for hand-written character recognition

Reproduced from (Y. Lecun et al. 1998). The network takes as input a 32x32 pixel image, which goes through multiple convolutions to extract features. In the first convolutional layer (C1), 6 convolutions are performed, using 6 different kernels. Each convolution of the image with a kernel results in an output image called feature map. By the process of convolution, one pixel of the feature map depicts the response to a specific pattern for a neighbourhood of the original image, corresponding to the size of the kernel (typically 3x3 or 5x5 pixels, depicted by black squares on the figure). In this model, subsampling layers are used to further condensate the information. Deeper in the network, the convolutions are done between a 3D volume made of multiple feature maps and 3D kernels (ex: 3 feature maps of dimensions NxN with a 5x5x3 kernel). Towards the end of the network (right side), the fully connected layers process large 1D feature vectors resulting from the successive convolutions (from 120 features to a final 10 features output).

1.8.2. Object-detection with Convolutional Neural Networks

The previous section describes classical architectures for deep-learning models, with a particular focus on convolutional neural networks (CNN) for image-classification. CNN have also been used for object-detection, originally coupled to a sliding-window approach like previous classifier-based object-detectors. This method was demonstrated for application such as face-detection (Rowley et al. 1998), more recently for mitosis detection in histology images of breast cancer (Cireşan et al. 2013) or for glomerulus detection in kidney tissue-section (Gallego et al. 2018). Modern CNN object-detectors abandoned the sliding-window search for more efficient localization strategies. Those methods include region-proposal as popularized by the Region-based CNN (R-CNN) (Girshick et al. 2016), or regression of bounding-boxes positions and dimensions (Szegedy et al. 2013) as in YOLO (Redmon et al. 2016), SSD (Liu et al. 2016) and RetinaNet (Lin et al. 2018). The principle remains identical, the image-patches generated from the region-proposal are passed to the CNN for feature extraction and classification. For the models performing bounding-box regression, the positions and dimensions of the bounding boxes are also part of the model prediction, in addition to the object-classes.

1.9. Aim of the work

Zebrafish is increasingly used in microscopy-based screening due to its various experimental advantages. As a vertebrate organism, it offers a physiological complexity that is highly relevant for translational studies in human. This is complemented by the availability of a wide repertoire of genetic and biochemical tools. Zebrafish are ideal for microscopic imaging applications, including large-scale studies using automated imaging systems, for e.g. toxicology or drug discovery research. However, their complex three-dimensional nature requires novel methodologies tackling the handling, imaging and analysis of thousands of zebrafish embryos. Furthermore, as most imaging and software solutions target cell-based assays, there is a lack of dedicated tools to fully exploit the potential of the zebrafish embryo in biomedical screening scenarios.

Therefore, the aim of this PhD project was to establish a flexible toolbox addressing the specific requirements of whole-organism screening assays. Using techniques from computer vision and data-science, novel scientific software and protocols for automated acquisition and analysis of large image datasets were developed. This also involved exploration, benchmarking, adaptation, and documentation of these methods for application on microscopy datasets.

The developed tools cover various steps of the image-based screening workflow: smart microscopy, (semi-)automated image acquisition, manual and automated annotations of regions of interest, image-classification, and data-exploration. Object-detection methods with various complexity and performance were implemented for the detection of specific tissues and organs in small organisms. For the scoring of specific phenotypes, supervised classification methods were implemented based on state of the art deep-learning models. Manual annotations tools were also developed to facilitate the systematic manual annotations of regions of interest in large image datasets, or to generate ground-truth annotations required for the training of supervised machine-learning algorithms. Finally, dimensionality reduction and adapted data-visualization platforms were illustrated for the exploration of large quantitative screening datasets.

In addition to the tools, extensive documentation has been established in the form of online wiki guides and video tutorials. The tools were designed to be of general applicability and are particularly adapted to phenotypic screening studies in small model organisms. They were illustrated for the acquisition and analysis of microscopy images of zebrafish and medaka embryos, both using brightfield and fluorescent channels. They have the potential to streamline the experimental workflow by drastically reducing the burden associated with manual annotation, exploration, and analysis of large datasets. Finally, the tools are open-source and can be adapted for the development of custom solutions.

2. Material and Methods

2.1. Hardware and software requirements

2.1.1. General remarks

All software solutions presented in this thesis rely on free open-source projects running identically on major operating systems (Windows, Mac, Linux). No special hardware requirements are necessary, except for the training of deep-learning models for which a NVIDIA CUDA-capable GPU is advised. From our experience, a simple consumer graphic card (e.g. 2 GB RAM) is sufficient for the task of training a model for image-classification, however for object-detection a more powerful GPU was necessary (5 GB or more).

2.1.2. Hardware configuration

Although no specific hardware requirement is necessary to run the presented software, a list of the hardware configurations used for this project is included below.

- **Laptop**

- Operating System: Windows 10 64-bit
- Model: Lenovo ThinkPad T570
- Motherboard: LENOVO 20H90017GE
- CPU: Intel Core I7-7500U 2.7 GHz
- RAM: 16 GB

- **Workstation1**

- Operating System: Windows 10 Professional 64-bit
- Motherboard: SuperMicro X8DT3
- CPU: Intel Xeon X5660 2.8 GHz (2x)
- RAM: 64 GB
- GPU: Nvidia RTX 2080 Ti

- **Workstation2 (ACQUIFER HIVE)**

- Operating System: Windows Server 2016 64-bit
- Motherboard: SuperMicro X9SRA/X9SRA-3
- CPU: Intel Xeon E5 1650 3.2 GHz
- RAM: 128 GB
- GPU: NVIDIA GTX760

2.1.3. Fiji

This research project resulted in novel plugins and functionalities for ImageJ/Fiji (Rueden et al. 2017: 2; Schindelin et al. 2012; Schneider et al. 2012), a user-oriented cross-platform software for scientific image analysis written in Java. Fiji is built on top of ImageJ and provides additional functionalities (script editor, updater…) and libraries. Most plugins reported here require some of these libraries and thus are not compatible with ImageJ unless specifically indicated in the dedicated method section. The plugins make extensive use of functions from the ImageJ API (Rasband and ImageJ contributors n.d.).

The plugins were tested on multiple versions of Fiji with ImageJ at its core. The latest tested version being Fiji 2.1.0 with ImageJ 1.53f. The latest version of Fiji can be downloaded as a zip from https://fiji.sc/ . The ImageJ core can be update or rolled backed to older versions within Fiji via the menu *Help > Update ImageJ*.

The plugins presented here can be installed by activating the dedicated update sites (Menu *Help > Update…* then *Manage update sites* and ticking the corresponding sites). Fiji should be restarted after activating the update sites.

2.1.4. KNIME

The Konstanz Information Miner (KNIME analytics platform) (Michael R. Berthold et al. 2009; Michael R Berthold et al. 2009) is an open-source graphical data-analysis environment written in Java and built on top of Eclipse (eclipse.org). It allows building complex data-processing workflows by linking single processing units called nodes. KNIME is particularly well adapted to process large tabular data or collections of images. Workflows can be saved to a .knwf file or shared publicly via the online KNIME Hub (hub.knime.com). The latest version of KNIME can be downloaded for the most common operating systems at knime.com/downloads. The workflow reported in this dissertation were last tested with KNIME 4.2.2.

2.1.5. Python

The solution presented here were tested on Python 3.6 but should work with any Python 3.x version unless mentioned differently in the concerned chapter. Python was installed with the free Anaconda individual python distribution available at https://www.anaconda.com/products/individual. For the installation of packages, either pip or anaconda was used (see dedicated sections).

2.1.6. OpenCV

This project relies heavily on the programming library OpenCV (OpenCV contributors 2000), which provides functions for classical image-processing operations (reading image files, applying filters, etc.). OpenCV has the advantage to be available for multiple programming languages, which facilitates porting code from one software to the other (e.g. from python to Fiji). In this project, the python and java bindings were used. While some of the presented software were also tested functional with later versions of OpenCV, the reader can take version 3.4.2 as reference version for this project. In python, the OpenCV library can be installed via pip, using the following command in a terminal:

pip install opencv-python to install the latest version

pip install opencv-python==3.4.2 to install this specific version

The opencv-python package was briefly tested on Unix systems via Binder (Forde et al. 2018) running a docker image (Forde et al. 2018). On Unix systems, the opencv-python package might trigger some errors du to incompatible GUI framework, which prevents the execution of the code. The headless version of the package can be installed instead.

pip install opencv-python-headless

In Fiji, functions from the OpenCV library can be accessed by any scripting language after activating the *IJ-OpenCV* update site (Domínguez et al. 2017). This update site also provides converter classes between the ImageJ and OpenCV objects, including image and ROI converters. The plugins reported here were tested with IJ-OpenCV 1.2.1. Source codes for the IJ-OpenCV project is available on GitHub: https://github.com/joheras/IJ-OpenCV.

Finally, for the training of cascade detectors, the program executables can be downloaded with OpenCV releases (OpenCV team 2020) up to version 3.4. The command line executables are then located in the subfolder *\opencv\build\x64\vc14\bin* and can be called by opening a command line in this directory and typing the name of the executable file.

2.1.7. GluonCV/MXNET

For the training of a deep-learning object-detector (see section 2.6 and 3.5), the python libraries GluonCV (Guo et al. 2020) and MXNET (Chen et al. 2015) were installed with GPU support using pip. Additionally, a system wide CUDA installation is necessary for GPU support (see section 2.1.9). It is important to use the correct version of CUDA according to the MXNET package, see the MXNET documentation at https://mxnet.apache.org/versions/1.7.0/get_started.

In the present case, we used the following configuration and hardware:

- Graphic card NVIDIA GeForce RTX 2080 Ti
- CUDA 9.2
- cuDNN 7.6.2
- mxnet-cu92mkl 1.5.0 (via pip)
- gluoncv 0.4.0 (via pip)

2.1.8. TensorFlow and Keras

The TensorFlow (Martín Abadi et al. 2015) and Keras (Chollet and others 2015) python libraries were used for image-classification via the integration in KNIME (see sections 2.8.2, 2.8.3). GPU support for these libraries relies on CUDA which is automatically installed with the GPU-version of the python packages. The following package versions were used:

- python 3.6.9 or 3.6.10
- tensorflow (or tensorfow-gpu) 1.12.0 (not above!)
- keras (or keras-gpu) 2.2.4
- pandas 0.24.5

2.1.9. CUDA

CUDA is a proprietary software library for hardware acceleration on CUDA-capable NVIDIA graphic cards (GPU) and is necessary for some deep learning workflows presented in this thesis (refer to the dedicated method sections for the CUDA version). For the section 3.6.2 involving the Keras and TensorFlow python packages, CUDA can be easily installed like a regular python package via the Anaconda python distribution. Installing tensorflow-gpu will automatically install the required versions of the CUDA toolkit and the cuDNN libraries, using this command in an anaconda prompt

conda install tensorflow-gpu

For the project involving object-detection with MXNET, a system-wide installation of CUDA and cuDNN is necessary in addition to the MXNET python package. Installers for CUDA and the complementary files for cuDNN can be downloaded from the NVIDIA website. In this case, it is important to download the right version of CUDA and cuDNN depending on the version of the MXNET package that will be used (see section about MXNET). Once installed for the full system using an installer, the version of the CUDA compiler can be verified by calling the following command in a terminal.

```
nvcc –version
```

2.2. Image-datasets

Benchmarks were carried out on previously acquired datasets, imaged using a widefield high-content screening microscope (IM04) from ACQUIFER using brightfield or fluorescent channels.

2.2.1. Medaka embryos

Randomly positioned and oriented unhatched medaka embryos in wells of 96-welll plate, imaged at 2X magnification in brightfield. The dataset originates from (Gierten et al. 2020) and is hosted on Zenodo (Gierten and Gehrig 2019).

2.2.2. Zebrafish larvae

Zebrafish larvae dorsally oriented within agarose cavities in wells of 96-well plate, imaged at 2X magnification in brightfield. Dataset available on Zenodo (Gehrig 2019).

2.2.3. Zebrafish pronephroi

Transgenic *Tg(wt1b:EGFP)* zebrafish larvae expressing GFP in the pronephroi, dorsally oriented in agarose cavities of wells of a 96-well plate. Images of the GFP channel at 4X and 10X magnification from previous screening studies were used (Pandey et al. 2019; Westhoff et al. 2020). A set of 10X magnification images is available on Zenodo (Pandey et al. 2020).

2.3. Interactive annotation tools

2.3.1. Roi 1-click tools

The source code for the 1-click toolset is contained in a single ImageJ macro-language file (.ijm) compatible with both ImageJ and Fiji. In Fiji, the toolset can be installed by activating the *Roi 1-click-tools* update site. In ImageJ, the toolset can be installed by copying the ijm file to the directory *ImageJ\macros\toolsets.* The toolbar can then be activated by clicking the >> on the right side of the ImageJ/Fiji window and selecting the Roi 1-click tool entry. Double-clicking the tool icons displays the configuration window.

Source code and documentations are hosted on GitHub (Thomas 2020k). Video tutorials are available on YouTube (Thomas 2020l).

2.3.2. Qualitative annotation plugins

The plugins are contained in a set of jython codes for Fiji, they are not compatible with a sole ImageJ installation. In particular, the *addButton* method from the *GenericDialogPlus* class of Fiji is needed. The plugins are installed in Fiji by activating the *Qualitative annotations* update site. Source code and documentations are hosted on GitHub (Thomas 2020j). Video tutorials are available on YouTube (Thomas 2020d).

2.4. Object-detection with Multi-Template Matching (MTM)

2.4.1. Installation in Fiji

The multi-template matching plugins can be installed in Fiji by activating the *IJ-OpenCV* and *Multi-Template-Matching* update sites. The plugins are listed in the entry *Multi-Template-Matching* at the bottom of the Plugin menu list. Preconfigured Fiji bundles for windows with dependencies and pre-installed plugins are archived on Zenodo at http://doi.org/10.5281/zenodo.3994247. Source code and documentation are available at https://github.com/multi-template-matching/MultiTemplateMatching-Fiji. Video tutorials are available on YouTube (Thomas 2019b).

2.4.2. Installation in python

The Multi-Template-Matching python package can be installed using pip. See https://pypi.org/project/Multi-Template-Matching/ for a list of available versions.

pip install Multi-Template-Matching will install the latest version (the name is case-sensitive)

pip install Multi-Template-Matching==1.5.3 will install a specific version, here 1.5.3

pip install Multi-Template-Matching --no-deps will install the package without its dependencies.

The pip installer verifies the dependencies versions to make sure the configuration is compatible if some dependencies are already installed. Currently Multi-Template-Matching does not require any specific version of its dependencies. As a reference, here is a list of dependencies and associated versions for a working configuration.

- Multi-Template-Matching == 1.5.1
- numpy == 1.16.4 (Harris et al. 2020)
- opencv-python-headless == 4.1.0.25
- scikit-image == 0.15.0 (van der Walt et al. 2014)
- scipy == 1.3.0 (SciPy 1.0 Contributors et al. 2020)
- pandas == 0.25.0 (McKinney 2010)

The package was tested on Windows and Unix system via Binder (Forde et al. 2018). Source codes, documentation and example jupyter notebooks are available on the following GitHub repository: https://github.com/multi-template-matching/MultiTemplateMatching-Python.

Since the publication of this dissertation, an alternative python implementation of multi-template matching relying on the packages scikit-image and shapely (but not OpenCV) was published, see https://pypi.org/project/mtm/ . It is more object-oriented than the original implementation.

2.4.3. Installation in KNIME

To install Multi-Template-Matching in KNIME, a KNIME and python installation with Multi-Template-Matching 1.5.1 is necessary. The python installation should be set up in KNIME as explained in the KNIME documentation (KNIME n.d.). The workflow file can be downloaded from the KNIME Hub (Thomas 2020i). Opening the workflow will automatically download the necessary KNIME extensions (KNIME Image Processing etc.). The workflow was last tested with KNIME 4.2.0.

2.4.4. Generating a template image

A template image can be generated by cropping a rectangular image region outlining an object of interest. In ImageJ/Fiji, this can be achieved by drawing a bounding box with the rectangle ROI tool and using one of the command *Image > Crop* or *Image > Duplicate*, or respectively the keyboard shortcuts *shift + x* or *shift + d*. The resulting image can be saved to a file.

For python and other programming languages, a 2D image array representing the template can be "cropped" from an image by slicing the image array along columns and rows. In pseudo-code, with *w* and *h* the width and height of the template and *i, j* the row and column index of the top left pixel of the bounding box outlining the template region, the corresponding template slice from an original image is *template = image[i + h - 1 ; j + w - 1].*

2.4.5. Template pre-processing

In Fiji, the plugins propose the generation of additional template images by rotation or flipping of the provided template(s). This is achieved thanks to the rotation and flipping functions of the *ImageProcessor* class of the ImageJ1 API.

In python, the template pre-processing is not part of the package and is left at the choice of the user, using any image-processing python library.

2.4.6. Computation of the score map and local extrema detection

Provided a template and an image to search, the function *matchTemplate* from OpenCV is used to compute the score map, using one of the normalised score methods. (1/*TM_SQDIFF_NORMED*: normalised-square difference, 3/*TM_CCORR_NORMED*: normalised cross-correlation, 5/*TM_CCOEFF_NORMED:* 0-mean normalised cross-correlation). Once the score map computed, candidate object-locations are localized as maxima of the score map when a correlation-score is used, and score minima when a difference-score is used.

If a single object is expected in the image, the global score maxima or minima is localised in the score map using the *minMaxLoc* function from OpenCV. If multiple objects are expected in the image, local score extrema are localised as follow. For the Fiji plugin, the local score maxima in the resulting score map are localised using the *MaximumFinder* class of the ImageJ1 API (ij.plugin.filter) with a tolerance of 0 and a user-defined threshold between 0 and 1.

For the python implementation, the local score maxima are localised using the p*eak_local_max* function from the scikit-image package (van der Walt et al. 2014).). The local extrema detection can be adjusted by a user-defined score threshold to tune the detection sensitivity (as explained in section 1.5.3), here with a score threshold instead of the probability score. Local score minima are localised similarly, as the local maxima of the inverted score map (invert = 1-score_map) and the score threshold is also inverted accordingly (max_threshold = 1 – min_threshold). For every local extremum, a bounding-box of dimensions identical to the template used for the computation of the score map is associated, with its top left corner localised at the position of the extremum. The score for this candidate detection is the value of the extrema.

2.4.7. Non-Maxima Suppression (NMS)

To yield a set *S’* of *N* non-redundant detections from the initial set of candidate object-locations *S*, a custom NMS procedure was implemented. The procedure is initialized by sorting the extrema in *S* by descending score (from most to least confident detections), and by adding the most confident detection (first one in the sorted set) to *S'*. Then an iterative process loops over the remaining extrema in *S*. For a given iteration, with the selected extremum, the intersection over union (IoU) between its associated bounding-box and the bounding-boxes of the previously collected extrema in *S'* are computed. If the list of IoU are all below the overlap threshold, which means that the current maximum is not substantially overlapping with any extrema of higher score in *S'*, then the tested extremum is added to *S'* otherwise it is discarded and the iteration continues with the next extremum in S, until we have collected *N* extrema in S' or until there are no more extrema available to test in *S*. Once the iteration terminated, the final object locations are provided by the extrema in *S´* (Figure 29). In practice *N* is the expected number of objects in the image, defined by the user, and the final set S' might contain *N* or less final detections. Initially, a custom implementation of NMS was written in python, but in the latest versions of the Multi-Template-Matching python package, it was replaced with the OpenCV equivalent (*cv2.dnn.NMSBoxes*).

2.4.8. Eye and lens detection in zebrafish

A 2-step template matching approach was used for the detection and segmentation of eye and lens in laterally oriented zebrafish larvae (see section 3.3.8). The images were acquired at the 4X magnification with the brightfield channel on an ACQUIFER IM04. The detection pipeline is implemented as a KNIME workflow. The workflow starts with a first template matching search with a single template, for the localization of the head region. The result was visualized as a mask image in KNIME, which is used to crop the head region using the *segment cropper* node. Within the cropped head region, a second template matching with a single eye template was carried out to localize and crop the eye region. For both these template searches, a normalised cross-correlation score was used for the detection of a single object per image.

For the segmentation of the eye-lens, a Canny edge detection filter (low threshold: 30 – high threshold: 50 – sigma: 3 pixels) was first applied on the image of the eye region via the scikit image python library. A circle enclosing the lens was finally localized from this edge map by applying the Hough circle transform of the scikit image library (minimum radius: 30 pixels, maximum radius: 40 pixels, 1 expected circle, step size: 1 pixel).

For the segmentation of the eye, background subtraction was first performed on the image of the eye region, by subtracting a blurred version of the image (*gaussian convolution* node – sigma: 30 pixels). The background-subtracted image was then thresholded (*global thresholder* node using Otsu method). The resulting binary image was then eroded 3 times and dilated back 3 times to remove smaller element (*morphological image operation* node). The binary image was converted to a KNIME labeling using the *connected component analysis* node. Segment smaller than 2000 pixels were removed using the *labeling filter* node. Finally, in a custom script embedded in a python node, the contour of the labeling was extracted using the OpenCV function *findCountour* and an ellipse was fit on the resulting contours using the OpencCV function *fitEllipse*. Scikit-image version 0.15.0 and OpenCV version 3.4.2 were used.

2.5. Object-detection with cascade detectors

The OpenCV command line executables to train cascades and visualize the features can be downloaded with OpenCV releases up to version 3.4, from the official website (OpenCV team 2020). The command line executables are then located in the subfolder *\opencv\build\x64\vc14\bin.* The executables can be called by opening a command line in this directory.

2.5.1. Fiji plugins for training set generation

The plugins (unreleased) are a collection of .py scripts which can be dragged and dropped on the main Fiji window to open them in the script editor, then clicking the *run* button will execute the plugin. To use the plugin, the object-regions should be first outlined as rectangular regions of interest in Fiji, stored in the ROI manager. This can be done for a collection of images opened as a stack. While outlining object-regions manually with the mouse in ImageJ/Fiji, pressing "t" on the keyboard will store the current ROI in the ROI manager. It is important that the outlined object-regions have an aspect-ratio close to the dimensions chosen for the cascade training, to prevent any deformation of the object as the training images are downscaled. For this purpose, pressing "alt" while resizing a rectangular ROI will maintain the aspect-ratio. For ROIs already stored in the ROI Manager, the dimensions (and position) can be manually set via the menu *More>Specify*… When annotating objects of similar sizes, using a fixed-sized reference ROI allows to facilitate the annotation process. This can be achieved by dragging a reference ROI and pressing "t" successively for every new location. Alternatively, the ROI 1-click tools can be used to quickly generate ROIs of predefined dimensions (see section 3.2.2).

2.5.2. Cascade training

Cascade detectors can be trained using the graphical user interface by (Ahmadi 2017), which can be downloaded from his personal website (see URL in reference). The parameters are identical to the ones available via the OpenCV command line utility. For the training images, one can simply put cropped images of the objects, and full-size background images in separate directories respectively named *p* and *n* (for positive and negative).

As an alternative to the GUI program, the executable *opencv_traincascade.exe* can be used to train a cascade via the command line. This command takes as arguments the location of the training set (*info.dat* or *train.vec* for the positive instances, *bg.txt* for the negative instances) and a number of arguments for the cascade training, including the maximal number of stages in the cascade, etc. See the online documentation (OpenCV contributors 2018) for details about the parameters and training.

For the positive training instances (the objects) the image file paths and object-location should be reported in a text file *info.dat*, which is then converted to a *.vec* file, containing both the training images and the object-locations. The structure of the info.dat is described in the OpenCV documentation (OpenCV contributors 2018).

The *.vec* file is then generated from the file *info.dat* with the following command

```
opencv_createsamples.exe -info info.dat -vec output.vec -h 24 -w 24
```

w and h are the target dimensions in pixels for the downscaling of the training images. They should correspond to the dimensions used for the training of the cascade, and thus impact the total number of Haar-features available for training (the larger the window, the higher the possible number of features, but the longer the training).

For the negative training instances, a simple text file (bg.txt) containing the file-paths to the full-size background images is sufficient.

2.5.3. Visualisation of the cascade features

After the training, the Haar-features can be visualized overlaid on a training image, using the executable *opencv_visualisation.exe* (see Figure 37). The overlay images can be saved to disk by precising the argument *d* (output directory).

```
opencv_visualisation.exe -m="yourPath\trainedCascade.xml" -i="yourPath\image.png"
-d="SomeOutPath\filename”
```

2.5.4. Cascade detection plugin for Fiji

The detection plugin is contained in a single jython script (.py, unpublished). The *IJ-OpenCV* update site should be activated in Fiji to provide the OpenCV dependencies and converter functions. For the detection with the cascade, the images should be opened in Fiji as a single image window or as a stack. Upon execution of the plugin, the images are first internally converted to 8-bit, using the *imageProcessor.convertToByte(True)* method of the ImageJ1 API and then converted to OpenCV Mat images using IJ-OpenCV. A *CascadeClassifier* instance is initialized from the xml file of the trained cascade provided in the plugin’s GUI. The detection is then run on the image using the *detectMultiScale3* method signature with the parameters *minNeighbors=1, outputRejectLevels=True* and the *scaleFactor, min/maxSize* as chosen in the plugin’s GUI. The set of detections returned as then passed to the function *cv2.dnn.NMSBoxes* with the chosen *score threshold* and *maxIoU* specified in the GUI. If a specific number of object (Nobject) is expected, then only up to Nobject are returned. Otherwise, all the detections passing the NMS are returned and rendered as rectangular ROIs in Fiji. For the benchmarks, the time reported corresponds for each image to the fraction of the code from the detection with *detectMultiScale3* until the NMS with *cv2.dnn.NMSBoxes*.

2.5.5. Training of a zebrafish larvae detector

A custom cascade detector was trained for the detection of zebrafish larvae, dorsally oriented within wells of a 96-well plate, as imaged with the Acquifer imaging machine at the lowest magnification (2X) (see section 2.2.2). Using the plugin described in section 3.4.1, zebrafish larvae from 60 training images were outlined as positive training instances and downscaled to the chosen detector dimensions (width x height: 113 x 23 pixels). The corresponding 60 artificial background images were generated, and 29 additional negative regions were also manually outlined and downscaled to the detector's dimensions (Figure 36). The training was done via the GUI software of (Ahmadi 2017) with the laptop configuration (see section 2.1.2). From the pool of negative images, the training utility was configured to generate 300 negative images. For the positive instances, the training was set to use 90% of the initial training instances, the 10% remaining being used to replace misclassified positive instances, which are "lost" over the training. The training was set to reach a maximum of 5 stages, with a maximal false positive rate per stage of 0.35 (maxFalseAlarmRate) and a minimum true positive rate per stage of 0.99 (minHitRate). The detection window was set to sampleWidth x sampleHeight: 113 x 23 pixels, corresponding to a pool of 3 275 676 possible Haar-features. The memory buffer for the precalculated feature values and indexes were both set to 4 GB. All other parameters were left to default (acceptanceRatioBreakValue: -1, stageType: BOOST, featureType: HAAR, boostType: GAB, weightTrimRate: 0.95, maxDepth: 1, maxWeakCount: 100, mode: BASIC). The training terminated after completing 2 stages in 5 min on the laptop configuration, as the number of negative training images was exhausted.

2.5.6. Detection with the Zebrafish larvae detector

For the detection of zebrafish larvae with the trained cascade, a separate test set of 33 full resolution images were used (2048 x 2048 pixels, see section 2.2.2). The detection was done in Fiji with the custom Fiji detection script (see section 2.5.4) with the following parameters:

Detection
scale factor: 1.1
min/max width: 1000 – 1500
min/max height: 200 – 350

NMS
minScore: 1
maxIoU: 0.3

2.5.7. Training of medaka embryos detector

For the training of a medaka embryo detector, embryos were outlined with square ROIs using the ROI 1-click tools, and the plugin described in section 3.4.1 was used to cropped the image regions. Besides, rotation and flipping were selected in the plugin GUI, which yielded 288 unique positive training images. The corresponding artificial background images were generated by the plugin. The training was done via the GUI software of (Ahmadi 2017) with the workstation 1 (see section 2.1.2). The training configuration was set to use 90% of the positive training images and to generate 1000 negative images by random cropping. The training was set to reach a maximum number of 3 stages, using an individual false positive rate per stage of 0.02 (maxFalseAlarmRate) and a minimum true positive rate of 0.99 (minHitRate). The detection window dimensions were sampleWidth x sampleHeight 24x24 pixels corresponding to a pool of 162 336 possible features. Within a stage, the maximal depth of a decision tree (maxDepth) was set to 3 (instead of the default of 1). The memory buffer for the precalculated feature values and indexes were both set to 3 GB. All other parameters were left to default (acceptanceRatioBreakValue:-1, stageType: BOOST, featureType:HAAR, boostType:GAB, weightTrimRate: 0.95, maxWeakCount: 100, mode: BASIC). The training lasted 1 min 42 secs on the workstation 1 and resulted in a 3-stage cascade.

2.5.8. Detection with the medaka embryo detector

For the detection of medaka embryos from section 2.2.1, 10 full resolution images (2048 x 2048 pixels) each containing 4 embryos randomly positioned and oriented in well of 96-well plates were used. The detection was done with a custom detection script in Fiji for Figure 40.A-B and with the plugin described in section 2.5.4 for Figure 40.C-D. The parameters for Figure 39 and Figure 40 were as following:

Detection
scale factor: 1.1
min/max width: 300 – 500
min/max height: 300 – 500

The NMS-specific parameters are indicated in the figure. *minNeighbors* corresponds to the parameter for the *cascade.detectMultiScale* OpenCV function. The *minScore* and *maxIoU* (or *maxOverlap*) value were passed to the function *cv2.dnn.NMSBoxes*.

2.6. Object-detection with deep-learning detectors

The methods described in this section require the GluonCV and MXNET libraries in python (see section 2.1.7).

2.6.1. Generating training set

For the training of an object-detector, one needs to provide both the location of objects in the training image as bounding-boxes and the associated object-category for each bounding-box. For GluonCV, this information is stored either in a text file (.lst) or a as a binary record file, as indicated in the GluonCV documentation (GluonCV Contributors 2020a). The lst file has one row per image, with all bounding-boxes of this image reported in this line. For each bounding-box, the corresponding object-category is reported as an integer (Figure 16). The binary record file is then generated from the lst file and the image files, using a command line utility im2rec.py available with GluonCV. This record file contains both the annotations and downscaled versions of the training images.

At the time of this project, no user-oriented software was available to generate this kind of file, so a custom workflow using Fiji for the object annotations, and KNIME for the conversion of the annotations was developed. The Fiji script should be executed after outlining object-regions with rectangular ROIs stored in Fiji's ROI Manager for a given class exclusively. Upon execution, the script iterates over the ROI and for each of them, reports the normalised ROI coordinates (xmin, ymin, xmax, ymax) to an output csv file (Figure 15). This was repeated for every object-classes. Then the KNIME workflow concatenates the set of csv-files to a single lst-file, and further converts it to a record file.

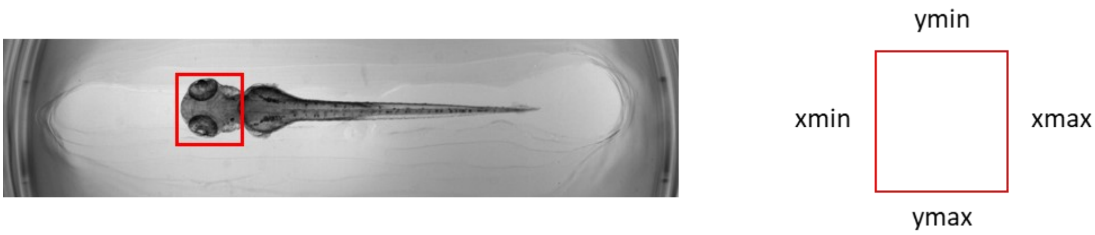
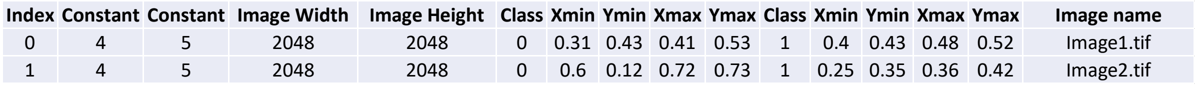


Figure 15: Encoding object-location as bounding-boxes

Here the object of interest is the head region of the zebrafish larvae outlined by the red bounding-boxes. For MXNET, the bounding boxes are encoded using the image row and column indexes corresponding to the position of the 4 bounding-box edges (xmin, xmax, ymin, ymax).



Index	Constant	Constant	Image Width	Image Height	Class	Xmin	Ymin	Xmax	Ymax	Class	Xmin	Ymin	Xmax	Ymax	Image name
0	4	5	2048	2048	0	0.31	0.43	0.41	0.53	1	0.4	0.43	0.48	0.52	Image1.tif
1	4	5	2048	2048	0	0.6	0.12	0.72	0.73	1	0.25	0.35	0.36	0.42	Image2.tif

Figure 16: Example of lst-file

This example file contains the ground-truth annotations for 2 images including the object categories (class) and the object-locations (Xmin, Ymin, Xmax, Ymax: bounding-boxes coordinates). The bounding-boxes coordinates are expressed relative to the image dimensions, which are expressed in pixels. Here both images contain each 2 objects, of class 0 and 1, but there can be a different number of objects in each image. In practice the file does not include the column headers, which are shown here for clarity.

2.6.2. Finetuning of an object-detector on custom dataset

Once a dataset has been annotated and the corresponding lst-file or rec file has been generated, the training of the convolutional neural network (CNN) can be started. Here we adapted a tutorial from the GluonCV documentation for the training of a custom object-detector by finetuning (GluonCV Contributors n.d.). A single-shot detector, using a MobileNet base for features extraction and class-prediction was finetuned. The model was initially trained on the VOC dataset for real-life objects detection. The original python script for the model finetuning was integrated in a KNIME workflow via a python scripting node, to facilitate operations like file inputs, the visualization of the error metrics over time, etc. The python node was embedded in a component node which contains additional nodes (input configuration nodes, former Quickform nodes) for the generation of a custom graphical user-interface for the setting of the training parameters (location of the training set, number of epochs, batch size, name for the output model). The model is trained for a number of epochs decided by the user. The progress of the training is reported to the command line, including the duration for each epoch. After the training, the weights for the trained model are exported to a file (.params extension), to use for object-detection on new images. A text file containing the names of the object-categories is also saved and is necesary to recover the category names from the predicted category index. The python script is configured to return the value of the error metrics for the training (class predictions and bounding-box positions). These values can then be visualized with a line plot node connected to the central component node.

2.6.3. Detection

Once a CNN model trained, it can be used to localize the objects in new images. A custom python script for model prediction was embedded in a KNIME workflow. The python script is configured to first load the model architecture used for the training (here the SSD with MobileNet base) using the function *model=gcv.model_zoo.get_model()* with the argument *pretrained=False* and the custom object-classes read from the text file generated by the training. Then the configuration for the trained model is loaded from the .params file using the function *model.load_parameters("myConfig.params")*. The images on which the detection should be run were loaded using a KNIME image reader node and recovered as numpy arrays within the python node. The images are then pre-processed using the *load_test()* function from the pre-processing module corresponding to the model architecture (here for the SSD model, from *gluoncv.data.transforms.presets.ssd*). This pre-processing downscales the images to dimensions compatible with the model and normalizes the intensity values. To visualize the detections as bounding-boxes overlaid on the original scale images, the function *gcv.utils.viz.plot_bbox* is used. To generate a mask image compatible for further processing in KNIME, the python script creates an 8-bit numpy array of dimensions identical to the original image and containing only 0 (black mask image). Then the bounding-boxes are re-scaled to the original image dimensions after detection by the model, using the function *gluoncv.data.transforms.bbox.resize(BBoxes, (smallWidth, smallHeight), (originalWidth, originalHeight))* with *smallWidth/Height* the dimensions of the downscaled images used by the model and *originalWidth/Height* the original image dimensions. The resulting bounding boxes are then drawn as plain rectangles on the mask image, using the category index as pixel value. After the python node, the *image to labelling* node converts this mask image to a mask object in KNIME, compatible with the overlay view of the *interactive segmentation view* node.

2.7. Automated microscopy

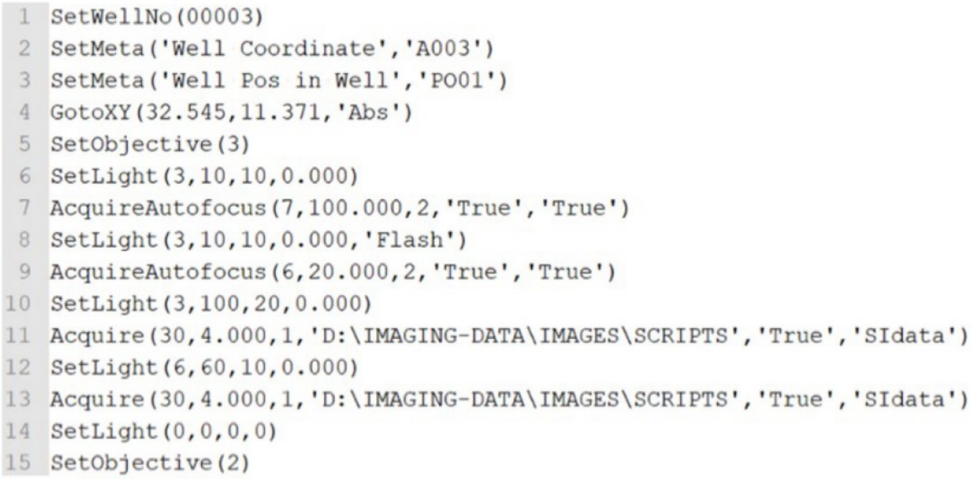
2.7.1. Feedback microscopy with the Acquifer Imaging Machine

Feedback microscopy was demonstrated on the Acquifer Imaging Machine (IM04) for the targeted high-resolution imaging of ROIs, in multiple samples mounted in 96-well plates. The IM04 supports feedback microscopy in parallel of a running acquisition, by regularly checking for the presence of custom machine commands, contained in newly created text files (.job extension) saved in a specific directory on disk *(e.g. "D:\IMAGING-DATA\JOBS"*). These text files can be written by any external process (python script, KNIME workflow etc.) which should interact with the acquisition. Examples of machine commands are shown in Figure 17. If such text files are present in the designated directory, the machine will pause the initial acquisition, execute the machine commands contained in those text files, delete the text files after execution and finally continue with the initial acquisition. The interval at which the machine checks for newly created files can be setup in the acquisition software. For instance, when imaging well plates, the verification can happen after imaging a specified number of wells, or after imaging the full plate with the current acquisition protocol.

For the targeted imaging of ROIs, the running acquisition is typically an overview experiment generating low magnification images. The low magnification images are saved on disk and analysed on the fly by an external ROI detection routine (e.g. a Fiji macro). For each image, the custom detection routine should identify the position of the ROI, convert the pixel coordinates of the ROI centre to machine axis coordinates for the positioning of the objective, and write the text file containing the custom machine commands to be executed by the microscope for these coordinates. The pixel size for the low magnification images, and the original XY position of the objective are both extracted by parsing the image filename, which contains the acquisition metadata. The text file with the custom microscope commands is initialised as a .tmp file, and once all commands have been written to the file it is finally renamed to a .job file, which is automatically recognized by the acquisition software of the Imaging Machine at the next verification.

In addition to a ROI-detection algorithm, a folder-watching routine is necessary to monitor image files, newly created by the overview experiment. This routine is responsible for calling the ROI-detection algorithm for every new overview image. It should be started either before the overview experiment or shortly after the start of the acquisition. However, the image directory to watch is not actually existing before the start of the acquisition. A new directory is systematically created by the acquisition software of the microscope for every new acquisition, to prevent overwriting existing data. The name of this directory includes a unique timestamp which is not known in advance. Therefore, the acquisition software of the microscope is writing at the beginning of the acquisition a small text file at a known location *("D:\IMAGING-DATA\JOBS\Image_Location.pth*"), which contains a single line of text with the image directory for this acquisition. The folder watching routine should thus be initialised to first read this *Image_Location.pth* file to know which directory to actively watch for the rest of the acquisition. The folder watching is typically implemented as a while loop, running forever until stopped by the user. It could also be configured to stop once a certain number of images have been found.

The full workflow is thus as follow: the acquisition is started and the *image_location.pth* file is written to disk shortly after the first image acquisition. The folder watching is started, checks for the presence of the *image_location.pth* file. If it is not found, the procedure is paused for a few seconds before checking for the file presence again. Once it is found, the image directory is read from the content of the *image_location.pth,* and the directory watching is started. Any image present in the directory is processed for ROI detection. As the overview acquisition progresses, new images are saved to this directory and will be similarly processed for ROI-detection. For every image, the detection routine verifies that the image corresponds to the expected channel or Z-slice. If the image is as expected, the detection routine is run and a text file with the custom machine commands for the targeted imaging of the detected ROIs is written to disk as described above. This text files will be picked up by the Imaging Machine and executed at the next verification.



```
SetWellNo(00003)
SetMeta('Well Coordinate','A003')
SetMeta('Well Pos in Well','P001')
GotoXY(32.545,11.371,'Abs')
SetObjective(3)
SetLight(3,10,10,0.000)
AcquireAutofocus(7,100.000,2,'True','True')
SetLight(3,10,10,0.000,'Flash')
AcquireAutofocus(6,20.000,2,'True','True')
SetLight(3,100,20,0.000)
Acquire(30,4.000,1,'D:\IMAGING-DATA\IMAGES\SCRIPTS','True','SIdata')
SetLight(6,60,10,0.000)
Acquire(30,4.000,1,'D:\IMAGING-DATA\IMAGES\SCRIPTS','True','SIdata')
SetLight(0,0,0,0)
SetObjective(2)
```

Figure 17: Example of a custom script with machine commands for the control of the ACQUIFER Imaging Machine

Example of job file containing custom commands for the acquisition at a particular location, here the well A3. 2 consecutive autofocus are carried out (coarse and fine) followed by the imaging with 2 different channels (SetLight + Acquire). Reproduced from (Pandey et al. 2019)

2.7.2. PlateViewer plugins for semi-automated acquisition

The visualization software of ACQUIFER (the PlateViewer (ACQUIFER 2017b)) is compatible with custom image-analysis by external programs in headless mode (e.g. Fiji, python) via a plugin mechanism relying on command-line calls and a convention for the input/output in the form of text files. With the current version of the PlateViewer (1.7.3), ROI-detection plugins should return the coordinates of a single rectangular ROI per image following the following protocol. A first csv file containing the file paths of the images selected by the user in the PlateViewer interface is saved to disk (input-table). This file may also contain the coordinates in pixels of an optional search region defined by the user in the PlateViewer. The csv file is a small table with one row per selected image. The external program (e.g. python or ImageJ) is then called by the PlateViewer with this input csv file as argument, and the predefined detection script. The detection script should be configured to open the input csv file, and to run the detection for every image listed. This can be achieved using a for-loop to process each line of the csv file successively. For each loop iteration, the image file should be loaded in memory, if a search region is defined the image can be cropped or only the delimited region can be loaded in memory and finally the detection should be run for this image. The results of the successive detections should be written to a new output csv file with a similar tabular structure, saved to a predefined directory on disk. It should contain one row per processed image with 1 column for the image path, 4 columns with the coordinates in pixels of the bounding-box for the detected image-region (X, Y, Width, Height, with X,Y the coordinates of the top left pixel of the bounding box) and one column for an optional string argument (e.g. a detection score).

Once the detection script terminates, the PlateViewer automatically reads the newly generated output csv file and the detections are rendered as rectangular ROIs on the images, in the PlateViewer interface. The ROIs can be ultimately refined if necessary, before updating the acquisition protocol.

2.7.3. Single-template matching PlateViewer plugin

For the template matching PlateViewer plugin, the ROI-detection is assured by a custom jython script in Fiji. The script reads the input table generated by the PlateViewer, which contains the file paths of the selected images, the coordinates for the search region as well the path of the image containing the template and the coordinates of the bounding-box outlining the template. The first step of the detection script is thus to crop the image containing the template. Once the template cropped, the script loops over the list of image files, loads the images one by one in memory and runs the template search on the successive images. Finally, for each image, the coordinates of the detected bounding-box are stored to a new line of an output csv file (x, y, width, height) together with the template matching score.

2.8. Image-classification

2.8.1. Classification using template-matching

A KNIME workflow was designed for image-classification using template matching, based on a set of template images representing distinct classes. The workflow relies on the multi-template-matching python package (version 1.6.1) but differs from the multi-template-matching KNIME workflow. Indeed, with this workflow, separate single template matching searches with a single object expected per image are carried out to derive the score for each template. Especially here there is no non-maxima suppression since the goal is not to localize but to classify. The resulting detections are collected and the score for each template are represented as a histogram, together with the user-defined score threshold (Figure 45). The image is assigned the category of the template which returned the highest score, provided it is above the user-defined threshold. If no score is above the threshold, the image is classified as an outlier. The KNIME workflow is available on GitHub (Thomas 2019a) or via the KNIME Hub (Thomas 2020b).

For the binary classification of kidney morphologies, the template and images were first cropped from full-resolution images, by defining a bounding box centred on the centre of mass of the organ as explained in (Pandey et al. 2019). The wild type template was generated by an average projection of 15 images from different specimens, to compensate for the variable fluorescence intensity within the organ. For the multi-class classification, the same wild type template was used than for binary classification, while for the long and cystic phenotypes, an average projection of 5 images was done to yield the templates.

2.8.2. Classification with deep learning - Requirements

KNIME workflows for the training of a CNN model for image-classification and for the prediction of image classes provided a trained model were developed. Those are available with documentation README.md files in the subdirectory *KNIMEworkflows* of the GitHub repository (Thomas 2020j).

The workflows were last tested with KNIME 4.2.2. All required KNIME extensions are automatically installed when opening the KNIME workflows for the first time (including the KNIME python extension), except for the KNIME Image Processing - Deep Learning extension. Click here to install it. In addition to KNIME and its python extension, a python environment with the required dependencies should be set up (e.g. in Anaconda, see section 2.1.8). Finally, KNIME should be set up to use the correct python environment (*File > Preferences > KNIME > Python*), but if Anaconda is used, KNIME can also automatically create a new preconfigured environment.

It is advised to have a CUDA-compatible NVIDIA GPU to speed up the training (2Gb GPU memory should be enough, but more is better). The GPU is automatically recognized by the Keras nodes in KNIME if the “gpu” versions of the TensorFlow and Keras python packages have been installed.

Images of fluorescently labelled pronephroi in transgenic *Tg(wt1b:EGFP)* zebrafish larvae used for the training and validation of the deep learning model for classification are available on Zenodo, with ground-truth category annotations and a pretrained model at: http://doi.org/10.5281/zenodo.3997728 . These files are available under a Creative Commons Attribution 4.0 International License (CC-BY). Finally, video tutorials about the workflows are available on YouTube.

2.8.3. Classification with deep learning - Workflow and CNN model

The KNIME workflows for the training of a deep-learning model for image classification perform the data pre-processing, the preparation of a custom model architecture and the training of this model. For the model architecture, a VGG16 base (Simonyan and Zisserman 2015) pretrained on the ImageNet dataset (Deng et al. 2009) is fetched from the internet using the Keras library and its weights are frozen. The output of the base network is flattened to a 1D vector (Keras Flatten layer). A fully connected layer (also called dense layer) with 64 neurons (also called units) with a ReLU activation function is added on top. Dropout with a probability of 0.5 is applied on the output from this 64-neuron layer (Keras Dropout Layer).

For the binary classification workflow, a last Keras dense layer containing a single neuron with a sigmoid activation is added on top of the 64-neurons dense layer. The output of the network is thus a single numerical value corresponding to the probability for the first class (p1). The probability for the second class is simply p2 = 1-p1. For the training, a binary cross-entropy is used for the computation of the loss.

For the workflow with 2 or more categories, the last layer added on top of the 64-unit dense layer is a fully connected layer with as many units as categories. Softmax is used as activation function for this layer. The output of the network is thus an array of the probabilities for each possible category. Finally, for the training, a categorical cross-entropy is used for the calculation of the loss.

For both training workflows, the default parameters for the training are 5 epochs with batch sizes of 8 images. For the rest of the parameters, the default values from the example workflow are used: RMSprop optimizer, learning rate: 10^{-4}, Rho: 0.9, Epsilon: 10^{-8}, learning rate decay: 0. As part of the pre-processing the images are automatically downscaled to 224 x 224 pixels and their intensity is normalized to the range [0-1]. The dimensions for downscaling can be adapted within a range of dimensions supported by the VGG16 architecture (minimum 32 x 32 pixels).

2.9. Interactive visualisation of multiple quantitative features

2.9.1. Interactive visualization in KNIME

The interactive visualisation of a multi-dimensional feature set is contained in a single KNIME workflow and does not require any other software except a local installation of the KNIME analytics platform. The workflow can be downloaded from the KNIME-Hub (Thomas 2020h). A video tutorial describing how to operate the workflow is available on YouTube (Thomas 2020g). Downloading the workflow and double clicking the knwf file will automatically open the workflow in KNIME and download the required KNIME extensions. The KNIME workflow takes as input an excel file, that should contain along the rows a list of samples, each of them described by multiple feature columns (Figure 18 – top table). Originally, the workflow was designed for the visualization of morphological kidney features in zebrafish embryos (Westhoff et al. 2020), but it can be adapted to other quantitative datasets. The original data table from the paper (FeatureTable.xlsx) can be downloaded from the associated OSF repository (Thomas, Laurent et al. 2020).

The feature used for colour coding of the data points (respectively lines) can be selected in the *colour manager* node. The features displayed in the PCA and parallel coordinates plot should also be specified in the configuration of these nodes.

For the parallel coordinates plot, each feature selected for the plot is represented by a vertical axis, and the axis range is automatically adapted for each feature separately. It is thus not centred on 0 contrary to some other plotting solutions offering this type of plot. The different features axis can also be interactively reordered by dragging them with the mouse.

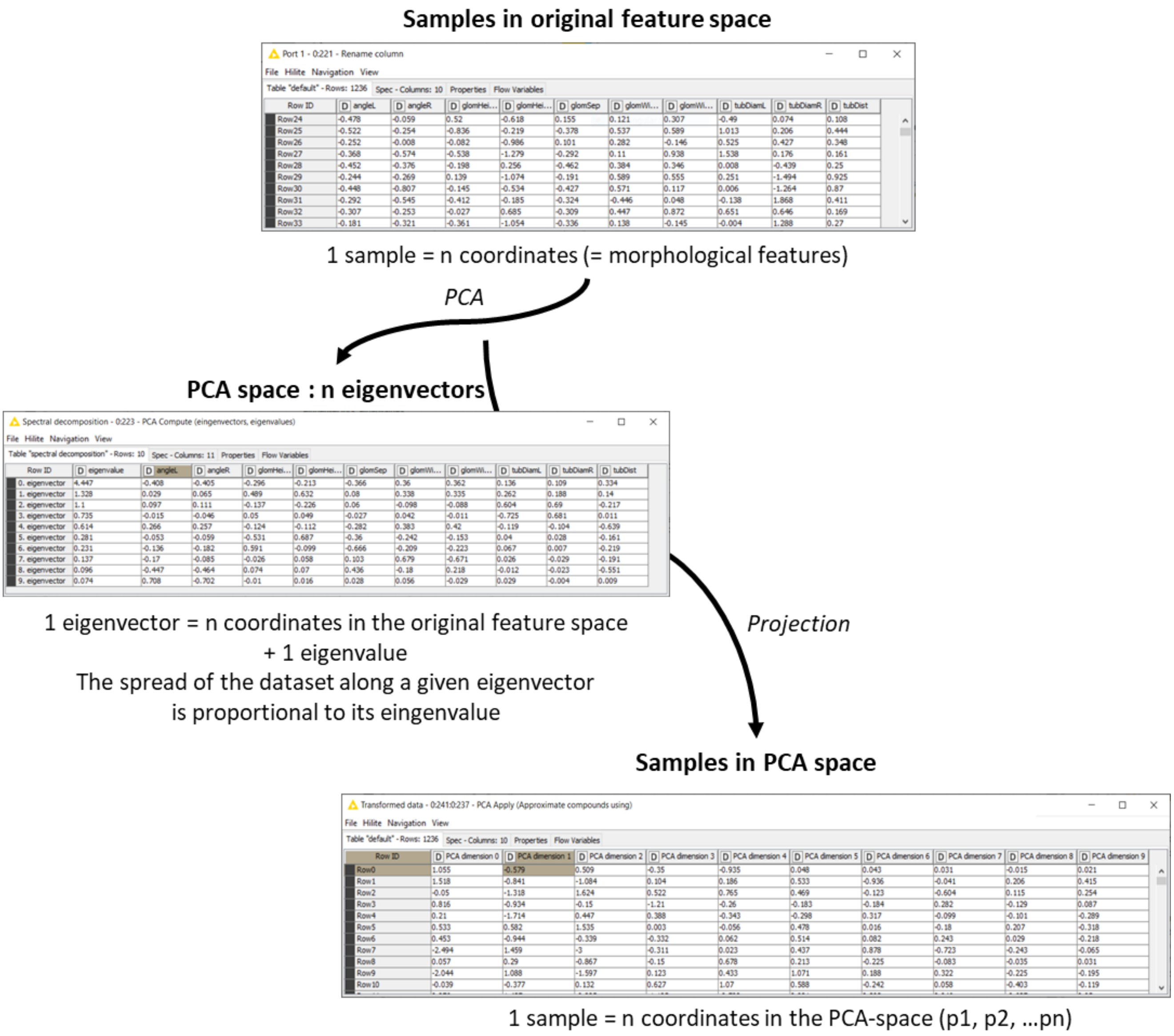
2.9.2. Interactive visualization in the TensorFlow embedding projector

The platform is accessible at http://projector.tensorflow.org/ . A custom dataset can be loaded by uploading 2 text files (csv), with samples ordered along the rows. One file should contain the feature values and the second the corresponding metadata (labels, sample category…). Custom data from a previous zebrafish screening study (Westhoff et al. 2020) to generate Figure 55 is hosted on GitHub (Thomas 2020c). The repository also contains the link to directly open the dataset in the platform.

2.9.3. Dimensionality reduction using Principal Component Analysis (PCA)

To visualize a dataset described by multiple quantitative features (n-dimensions) as a 3D point cloud, PCA was used for dimensionality reduction, to convert the n-features for each sample to 3 pseudo-features. Those 3 features are then used as x,y,z-coordinates for the 3D point cloud, in which each point corresponds to a sample. Samples with similar feature values localize in similar regions of the point cloud. It is thus more intuitive to visually identify clusters of similar samples. PCA is a statistical method which derives from the original feature space (n dimensions), a related n-dimensional space (PCA-space). The 2 spaces are related, similar to a change of origin, therefore the axis of the PCA space can be expressed relative to the original feature space axis. These axes are also called dimensions or eigenvectors. Similarly, it is possible to derive coordinates for the samples in this PCA space, by projecting the original feature values of each sample along those eigenvectors (see Figure 18). The advantage of the PCA space, is that the axes are ordered according to the dispersion of the data, meaning that the dataset is mostly spread along the 1st PCA axis and the least dispersed along the last PCA axis. The dispersion along an axis is given by the eigenvalue of each eigenvector (or axis).

Once the data projected into the PCA space, one can approximate the data by using only a fraction of the PCA axis for each sample. This is the so-called dimensionality reduction step. Using 3 dimensions, one can thus represent the original n-features in a 3D space using for each sample the values for the first 3 PCA axis only. This is similar to setting the other coordinates to 0 for each sample, instead of the original values (see Figure 19). Dimensionality reduction yields an approximation of the original data, as the information from the PCA axis of higher order is lost. Using more PCA dimensions allows to have a more precise description of the original data, ultimately using all PCA dimensions no approximation of the data is done. The degree of approximation can be calculated by deriving the fraction of the original data variance, actually represented after dimensionality reduction. The higher the represented variance, the more representative of the true distribution is the dimensionality reduced dataset, and the more faithful the 3D scatter plot. For the original paper, dimensionality reduction from 10 morphological features down to 3 PCA dimensions corresponded to representing 76% of the original data variance. PCA and dimensionality reduction are further detailed the dedicated chapter of the python data science handbook by (VanderPlas 2016), freely available online (see link in reference).



Samples in original feature space

Port 1 - 0:221 - Rename column

File Hilite Navigation View

Table "default" - Rows: 1236 | Spec - Columns: 10 | Properties | Flow Variables

Row ID	D angleL	D angleR	D glomHei...	D glomHei...	D glomSep	D glomWi...	D glomWi...	D tubDiamL	D tubDiamR	D tubDist
Row24	-0.478	-0.059	0.52	-0.618	0.155	0.121	0.307	-0.49	0.074	0.108
Row25	-0.522	-0.254	-0.836	-0.219	-0.378	0.537	0.589	1.013	0.206	0.444
Row26	-0.252	-0.008	-0.082	-0.986	0.101	0.282	-0.146	0.525	0.427	0.348
Row27	-0.368	-0.574	-0.538	-1.279	-0.292	0.11	0.938	1.538	0.176	0.161
Row28	-0.452	-0.376	-0.198	0.256	-0.462	0.384	0.346	0.008	-0.439	0.25
Row29	-0.244	-0.269	0.139	-1.074	-0.191	0.589	0.555	0.251	-1.494	0.925
Row30	-0.448	-0.807	-0.145	-0.534	-0.427	0.571	0.117	0.006	-1.264	0.87
Row31	-0.292	-0.545	-0.412	-0.185	-0.324	-0.446	0.048	-0.138	1.868	0.411
Row32	-0.307	-0.253	-0.027	0.685	-0.309	0.447	0.872	0.651	0.646	0.169
Row33	-0.181	-0.321	-0.361	-1.054	-0.336	0.138	-0.145	-0.004	1.288	0.27

1 sample = n coordinates (= morphological features)

PCA space : n eigenvectors

Spectral decomposition - 0:223 - PCA Compute (eingenvectors, eigenvalues)

File Hilite Navigation View

Table "spectral decomposition" - Rows: 10 | Spec - Columns: 11 | Properties | Flow Variables

Row ID	D eigenvalue	D angleL	D angleR	D glomHei...	D glomHei...	D glomSep	D glomWi...	D glomWi...	D tubDiamL	D tubDiamR	D tubDist
0. eigenvector	4.447	-0.408	-0.405	-0.296	-0.213	-0.366	0.36	0.362	0.136	0.109	0.334
1. eigenvector	1.328	0.029	0.065	0.489	0.632	0.08	0.338	0.335	0.262	0.188	0.14
2. eigenvector	1.1	0.097	0.111	-0.137	-0.226	0.06	-0.098	-0.088	0.604	0.69	-0.217
3. eigenvector	0.735	-0.015	-0.046	0.05	0.049	-0.027	0.042	-0.011	-0.725	0.681	0.011
4. eigenvector	0.614	0.266	0.257	-0.124	-0.112	-0.282	0.383	0.42	-0.119	-0.104	-0.639
5. eigenvector	0.281	-0.053	-0.059	-0.531	0.687	-0.36	-0.242	-0.153	0.04	0.028	-0.161
6. eigenvector	0.231	-0.136	-0.182	0.591	-0.099	-0.666	-0.209	-0.223	0.067	0.007	-0.219
7. eigenvector	0.137	-0.17	-0.085	-0.026	0.058	0.103	0.679	-0.671	0.026	-0.029	-0.191
8. eigenvector	0.096	-0.447	-0.464	0.074	0.07	0.436	-0.18	0.218	-0.012	-0.023	-0.551
9. eigenvector	0.074	0.708	-0.702	-0.01	0.016	0.028	0.056	-0.029	0.029	-0.004	0.009

1 eigenvector = n coordinates in the original feature space
+ 1 eigenvalue
The spread of the dataset along a given eigenvector
is proportional to its eingenvalue

Samples in PCA space

Transformed data - 0:241:0:237 - PCA Apply (Approximate compounds using)

File Hilite Navigation View

Table "default" - Rows: 1236 | Spec - Columns: 10 | Properties | Flow Variables

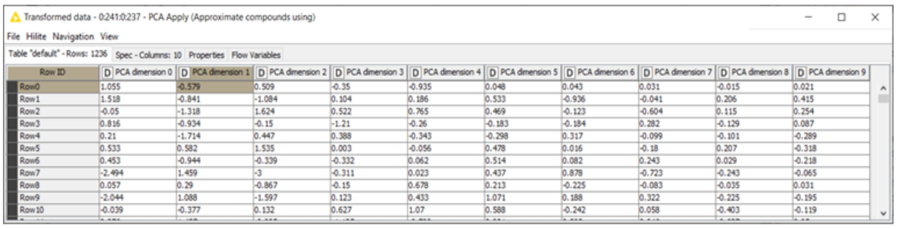
Row ID	D PCA dimension 0	D PCA dimension 1	D PCA dimension 2	D PCA dimension 3	D PCA dimension 4	D PCA dimension 5	D PCA dimension 6	D PCA dimension 7	D PCA dimension 8	D PCA dimension 9
Row0	1.055	-0.579	0.509	-0.35	-0.935	0.048	0.043	0.031	-0.015	0.021
Row1	1.518	-0.841	-1.084	0.104	0.186	0.533	-0.936	-0.041	0.206	0.415
Row2	-0.05	-1.318	1.624	0.522	0.765	0.469	-0.123	-0.604	0.115	0.254
Row3	0.816	-0.934	-0.15	-1.21	-0.26	-0.183	-0.184	0.282	-0.129	0.087
Row4	0.21	-1.714	0.447	0.388	-0.343	-0.298	0.317	-0.099	-0.101	-0.289
Row5	0.533	0.582	1.535	0.003	-0.056	0.478	0.016	-0.18	0.207	-0.318
Row6	0.453	-0.944	-0.339	-0.332	0.062	0.514	0.082	0.243	0.029	-0.218
Row7	-2.494	1.459	-3	-0.311	0.023	0.437	0.878	-0.723	-0.243	-0.065
Row8	0.057	0.29	-0.867	-0.15	0.678	0.213	-0.225	-0.083	-0.035	0.031
Row9	-2.044	1.088	-1.597	0.123	0.433	1.071	0.188	0.322	-0.225	-0.195
Row10	-0.039	-0.377	0.132	0.627	1.07	0.588	-0.242	0.058	-0.403	-0.119

1 sample = n coordinates in the PCA-space (p1, p2, ...pn)

Figure 18: Principal Component Analysis (PCA) applied to data

At the top is the original table of the dataset, with one sample per row, and a set of 10 feature columns (original feature space), here corresponding to morphological feature measurements. Using PCA, one can derive the corresponding 10 eigenvectors composing the PCA space, with their coordinates in the original feature space and eigenvalues, corresponding to the spread of the dataset along this particular PCA-axis. Finally, the original samples can be projected in the PCA space, such that each sample is represented by 10 coordinates in this PCA-space. Own contribution, reproduction and redistribution allowed under a CC-BY license.

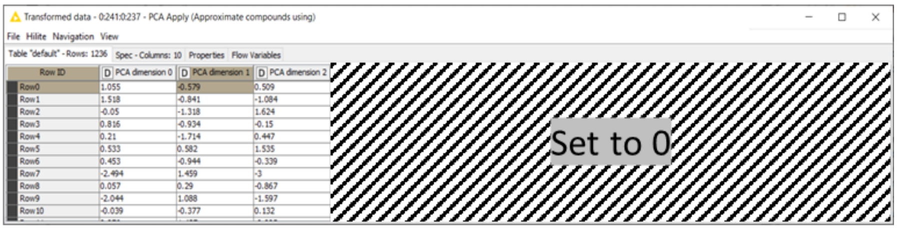
Samples in PCA space


Transformed data - 0:241:0:237 - PCA Apply (Approximate compounds using)
File
Hilite
Navigation
View
Table "default" - Rows: 1236
Spec - Columns: 10
Properties
Flow Variables

Row ID	D PCA dimension 0	D PCA dimension 1	D PCA dimension 2	D PCA dimension 3	D PCA dimension 4	D PCA dimension 5	D PCA dimension 6	D PCA dimension 7	D PCA dimension 8	D PCA dimension 9
Row0	1.055	-0.579	0.509	-0.35	-0.935	0.048	0.043	0.031	-0.015	0.021
Row1	1.518	-0.841	-1.084	0.104	0.186	0.533	-0.936	-0.041	0.206	0.415
Row2	-0.05	-1.318	1.624	0.522	0.765	0.469	-0.123	-0.604	0.115	0.254
Row3	0.816	-0.934	-0.15	-1.21	-0.26	-0.183	-0.184	0.282	-0.129	0.087
Row4	0.21	-1.714	0.447	0.388	-0.343	-0.298	0.317	-0.099	-0.101	-0.289
Row5	0.533	0.582	1.535	0.003	-0.056	0.478	0.016	-0.18	0.207	-0.318
Row6	0.453	-0.944	-0.339	-0.332	0.062	0.514	0.082	0.243	0.029	-0.218
Row7	-2.494	1.459	-3	-0.311	0.023	0.437	0.878	-0.723	-0.243	-0.065
Row8	0.057	0.29	-0.867	-0.15	0.678	0.213	-0.225	-0.083	-0.035	0.031
Row9	-2.044	1.088	-1.597	0.123	0.433	1.071	0.188	0.322	-0.225	-0.195
Row10	-0.039	-0.377	0.132	0.627	1.07	0.588	-0.242	0.058	-0.403	-0.119

1 sample = n coordinates in the PCA-space (p1, p2, ...pn)

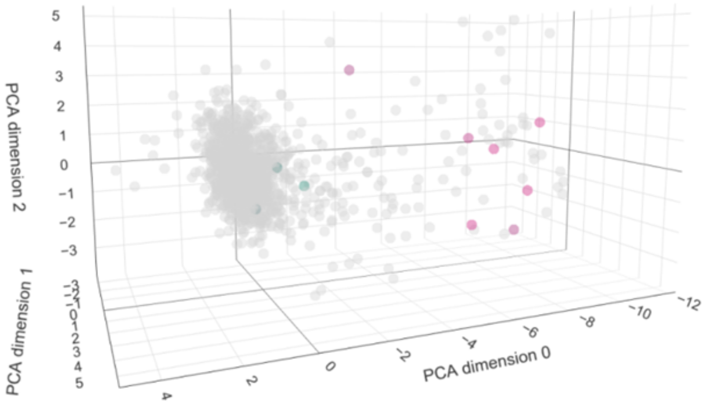
Approximate samples in PCA space


Transformed data - 0:241:0:237 - PCA Apply (Approximate compounds using)
File
Hilite
Navigation
View
Table "default" - Rows: 1236
Spec - Columns: 10
Properties
Flow Variables
Set to 0

Row ID	D PCA dimension 0	D PCA dimension 1	D PCA dimension 2
Row0	1.055	-0.579	0.509
Row1	1.518	-0.841	-1.084
Row2	-0.05	-1.318	1.624
Row3	0.816	-0.934	-0.15
Row4	0.21	-1.714	0.447
Row5	0.533	0.582	1.535
Row6	0.453	-0.944	-0.339
Row7	-2.494	1.459	-3
Row8	0.057	0.29	-0.867
Row9	-2.044	1.088	-1.597
Row10	-0.039	-0.377	0.132

1 sample = x non-zero coordinates (p1, p2, ..., px, 0,...0)

3D visualization of the non-zero components (x=3)



1 sample = 1 point

Figure 19: PCA for dimensionality reduction and 3D visualization

After projection in the PCA-space, one can visualize the distribution of the samples, using only a fraction of the PCA components. Using the 3 first principal components for each sample, one can visualize the data distribution as a 3D cloud of points. Own contribution, reproduction and redistribution allowed under a CC-BY license.

3. Results

3.1. Overview

The table below provides an overview of the publications and other outcomes of this research.

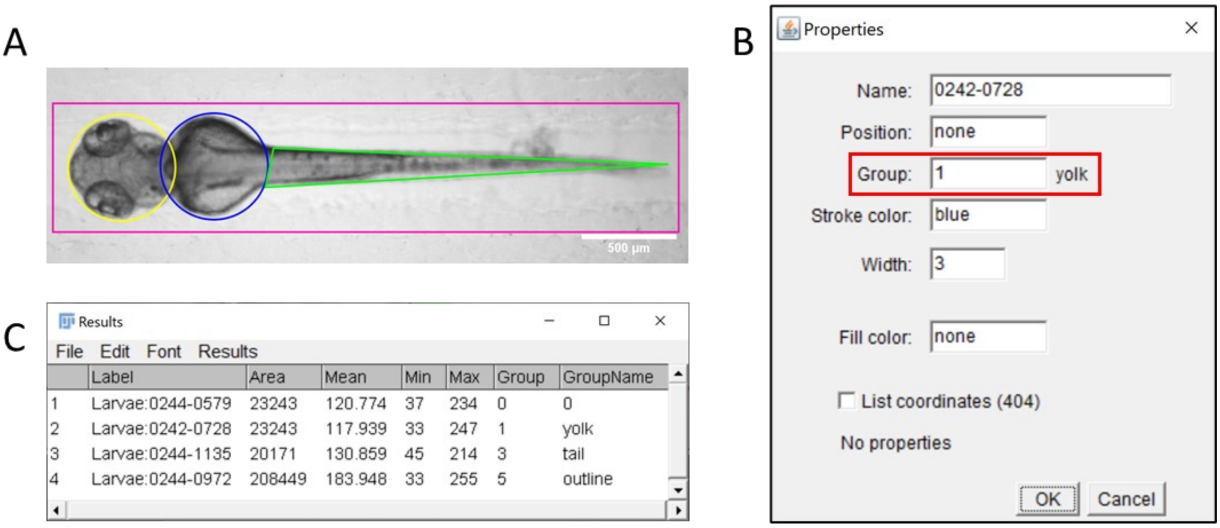
Name	Reference
Roi 1-click tools	
- Journal publication	(Laurent S.V. Thomas and Gehrig 2020)
- Source codes and documentation	(Thomas 2020k)
- Video tutorials	(Thomas 2020l)
Qualitative annotation plugins	
- Journal publication	(Thomas et al. 2020)
- Source codes, documentation and workflows	(Thomas 2020j)
- Video tutorials	(Thomas 2020d)
Multi-Template Matching	
- Journal publication	(Laurent S. V. Thomas and Gehrig 2020)
- Source codes, documentation and workflows	(Thomas 2019a)
- Video tutorials	(Thomas 2019b)
Interactive exploration of phenotypic datasets	
- Journal publication	(Westhoff et al. 2020)
- KNIME exploration dashboard	(Thomas 2020h)
- Preconfigured tensorflow embedding projector	(Thomas 2020c)
- Video tutorial	(Thomas 2020g)

3.2. Interactive annotation tools

This section describes the development and application of novel functionalities and annotation tools for the popular scientific image-analysis software ImageJ (Schneider et al. 2012) and Fiji (Rueden et al. 2017: 2; Schindelin et al. 2012). The tools facilitate the annotations of image or image-regions with applications for routine image evaluation or the training of supervised classification algorithms. The tools are described in more details in the associated publications (Thomas et al. 2020; Laurent S.V. Thomas and Gehrig 2020) and available in Fiji by activating the corresponding update sites (*Roi 1-click tools* and *Qualitative image annotations*).

3.2.1. Implementing ROI category annotation in ImageJ/Fiji

ImageJ/Fiji is a popular scientific image-analysis software with powerful visualization, processing, and analysis functionality. Regions of interest (ROI) (e.g. cells) can be outlined in images using one of the mouse-selection tools (see ImageJ documentation (Rasband and Ferreira 2012)). The ROI can then be used for further processing, for instance to limit the application of a filter or a quantitative measurement to this specific image-region, or to crop the image-region. However, at the time of this project, no built-in functionality existed to help users annotate ROI for different categories e.g. dead vs alive cells or different body-parts of small organisms. ImageJ being an open-source project, contributions to the source code can be publicly made on its GitHub code repository (ImageJ contributors 2005). Therefore, I contributed to the source code, by implementing a category index associated to ROI objects (Thomas 2020e), the so-called "group" attribute. ROIs with the same index belong to the same category. The category index can range from 0 (default, no category) to 255 and each index value is associated to a predefined colour, such that ROIs of different categories are shown with different colours (Figure 20.A). The index can be edited at any time via the ROI Manager (Figure 20.B) and is saved with the ROI object when exported to disk. Finally, the indexes and associated category names are reported in the result table with the quantitative measurements (Figure 20.C). The functionality is available from ImageJ version 1.52t and is similarly available for Fiji. More details about the functionality is available in the following presentation (Thomas 2020m).



Properties

Name: 0242-0728

Position: none

Group: 1 yolk

Stroke color: blue

Width: 3

Fill color: none

□ List coordinates (404)

No properties

Results

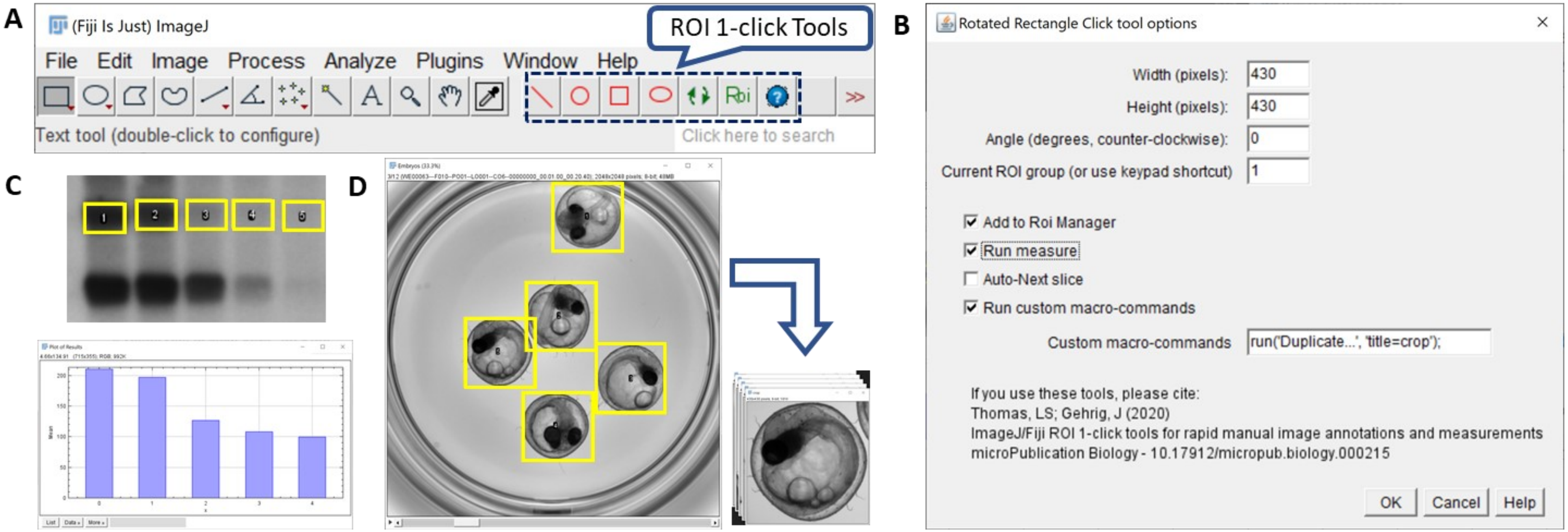
	Label	Area	Mean	Min	Max	Group	GroupName
1	Larvae:0244-0579	23243	120.774	37	234	0	0
2	Larvae:0242-0728	23243	117.939	33	247	1	yolk
3	Larvae:0244-1135	20171	130.859	45	214	3	tail
4	Larvae:0244-0972	208449	183.948	33	255	5	outline

Figure 20: Annotating different tissues using the ROI-group in ImageJ/Fiji

A) The ROI group attribute was used to annotate different body parts in this zebrafish larvae (brightfield channel, 2X magnification). Each ROI has a specific category index associated to a colour. The colour indexes are as following: yellow: 0, blue:1, green:3, pink:5, index 2 and 4 were not used, as the colour were not ideal for display. **B)** The group attribute can be edited individually for each ROI via the Properties menu of the ROI Manager. **C)** The group index and associated names are reported in the result table with other quantitative measurements. The head-region was outlined with the group 0 (default) for which no name is allowed. Own contribution, reproduction allowed under a CC-BY licence.

3.2.2. Roi-1-click tools

Manually outlining image-regions with a mouse-selection tool is still a time-consuming task when repeated for numerous images. To streamline these annotations for large datasets or large number of objects, we developed a new set of selection tools, extending the built-in ImageJ ROI tools (Rasband and Ferreira 2012) by offering more automated options (Laurent S.V. Thomas and Gehrig 2020)(Figure 21.A). With the so-called *Roi 1-click tools,* a single click generates a ROI of predefined dimensions, centred on the clicked pixel, while optional commands can be simultaneously executed. These commands include the addition of the newly drawn ROI to the ROI Manager, the reporting of quantitative measurements for the outlined image-region, the execution of a custom macro-command (e.g. cropping) and the display of the next image slice for image-stacks. The optional commands as well as the group associated to the newly drawn ROI are specified via the configuration dialog of the tools (Figure 21.B). The tools were illustrated for the annotation and quantification of bands in electrophoresis gels (Figure 21.C), and dot blots (not shown). They were also used to outline and crop objects of interest, which can be directly saved as a stack or collection of images with standardized dimensions (Figure 21.D).



Rotated Rectangle Click tool options

Width (pixels): 430

Height (pixels): 430

Angle (degrees, counter-clockwise): 0

Current ROI group (or use keypad shortcut) 1

☑ Add to Roi Manager

☑ Run measure

☐ Auto-Next slice

☑ Run custom macro-commands

Custom macro-commands run('Duplicate...', 'title=crop');

If you use these tools, please cite:
Thomas, LS; Gehrig, J (2020)
ImageJ/Fiji ROI 1-click tools for rapid manual image annotations and measurements
microPublication Biology - 10.17912/micropub.biology.000215

Figure 21: Rapid annotation of standard shapes with the Roi 1-click tools for ImageJ/Fiji

A) The 1-click ROI toolset in the Fiji toolbar. Double-clicking the tool icon displays (**B**) the associated option window to set the ROI shapes and custom actions to execute upon clicking (here for the rectangle tool). **C)** Using the rectangle 1-click tool for the quantification of bands intensity in electrophoresis gels. D) Using the rectangle 1-click tool to annotate specimen position. Using a custom command (as in B), the annotated regions are automatically cropped. Reproduced from (Laurent S.V. Thomas and Gehrig 2020) under the term of a CC-BY licence.

3.2.3. Qualitative image annotations plugins

To tackle the lack of widespread solution for the qualitative annotations of images with descriptive keywords, we developed a set of interactive annotations plugins for Fiji (Thomas et al. 2020). The plugins provide a tailored graphical user interface (GUI) with buttons, checkboxes, or dropdown menus for the assignment of the keywords, previously defined by the user. Categories or descriptive keywords can be assigned to images, or images-regions outlined by ROIs. The selected keywords are reported in a result table, together with the image or ROI identifier. The table is automatically updated as the user progresses with the annotation of new image instances. Besides qualitative annotations, any quantitative measurement as selected in the ImageJ menu *Analyze > Set Measurements* can also be reported if the *run Measure* option is selected. This option allows both a quantitative and qualitative description of the image-instance. Three plugins are provided which differ in their GUI to account for different annotation scenarios (Figure 22-Figure 24).

The *single-class (buttons)* plugin allows the assignment of a single keyword per image by clicking the corresponding button, as illustrated in Figure 22 for the annotation of the mitotic state of dividing cells. With this plugin, the user decides if the result table should contain a single category column with the clicked category keyword for each image, or one column per category with a binary code (0/1) depicting the assignment. The latter option is particularly adapted to the training of supervised classification algorithms, which typically expect for their training an array of probabilities with 1 for the actual image-category and 0 for all other categories (also called 1-hot encoding, see (Müller and Guido 2016)).

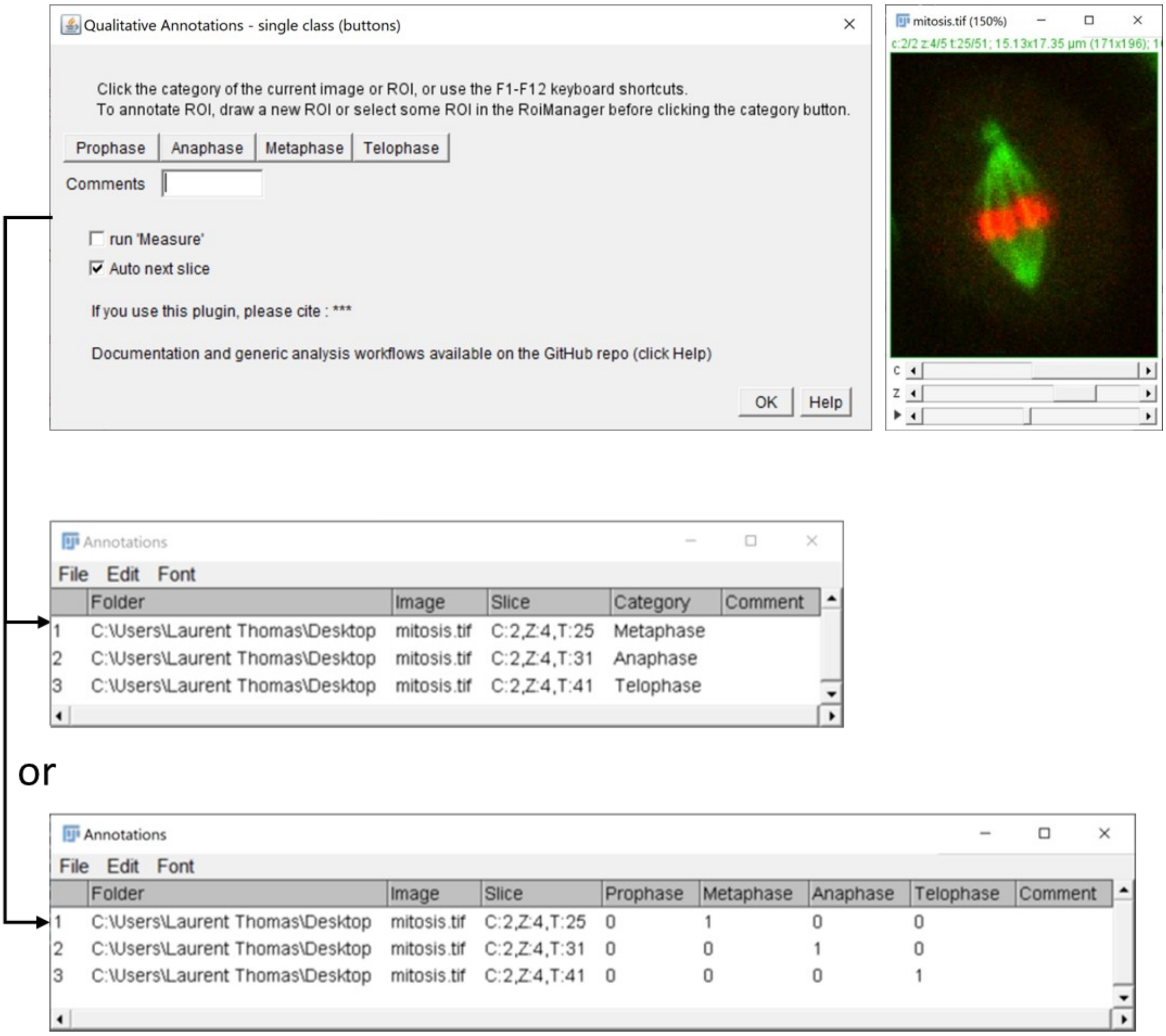
The *multi-class (checkboxes)* plugin allows multiple keywords per image, which are selected via associated checkboxes. It is illustrated in Figure 23 for the annotation of organ-specific phenotypes in transgenic zebrafish larvae. This plugin yields a result table with one column per keyword, and a 0/1 code if the keywords apply or not. The table structure is thus similar than with *the single-class (buttons)* plugin except that multiple keywords might be selected for a given image (i.e. multiple 1 for a given table row, as in row 1 of Figure 23).

The *multi-class (dropdown)* plugin allows choosing keywords from several lists of choices using dropdown menus. This plugin is convenient if multiple image features should be reported with several options for each feature. It is illustrated in Figure 24 for the phenotypic description of multi-cellular embryos outline by ROIs, with the annotation of the pigmentation, shape and texture.

The plugins are open-source and available by activating the *Qualitative Annotations* update site within Fiji. The source codes, example of analysis pipelines and example image data were also released together with the plugins (Pandey et al. 2020; Thomas 2020j). Additionally, video tutorials are available on YouTube (Thomas 2020d).

3.2.4. Generic workflows for the exploitation of qualitative annotations

To illustrate potential applications of the qualitative annotations generated using the plugins described above, generic KNIME example workflows were developed and are available via the KNIME Hub (Thomas 2020f). Example data and associated annotations to test the workflows are available on Zenodo (Pandey et al. 2020) and on the GitHub repository (Thomas 2020j). One of these workflows was developed for the visualization of the feature distribution as a sunburst chart, illustrated in Figure 25 for the phenotyping of multi-cellular embryos as in Figure 24. The workflow first enumerates the occurrence of given combinations of feature values (Figure 25.A) in the original annotation table and represents this information as a sunburst chart (Figure 25.B). This type of visualization is particularly adapted for qualitative features, to identify trends between features. A second workflow was adapted for the training of a deep learning model for image-classification, described in details in section 3.6.2. The example workflows take as input one of the tables generated by the annotation plugins and can be readily executed without major adaptation.



Qualitative Annotations - single class (buttons)

Click the category of the current image or ROI, or use the F1-F12 keyboard shortcuts.
To annotate ROI, draw a new ROI or select some ROI in the RoiManager before clicking the category button.

□ run 'Measure'
☑ Auto next slice

If you use this plugin, please cite : ***

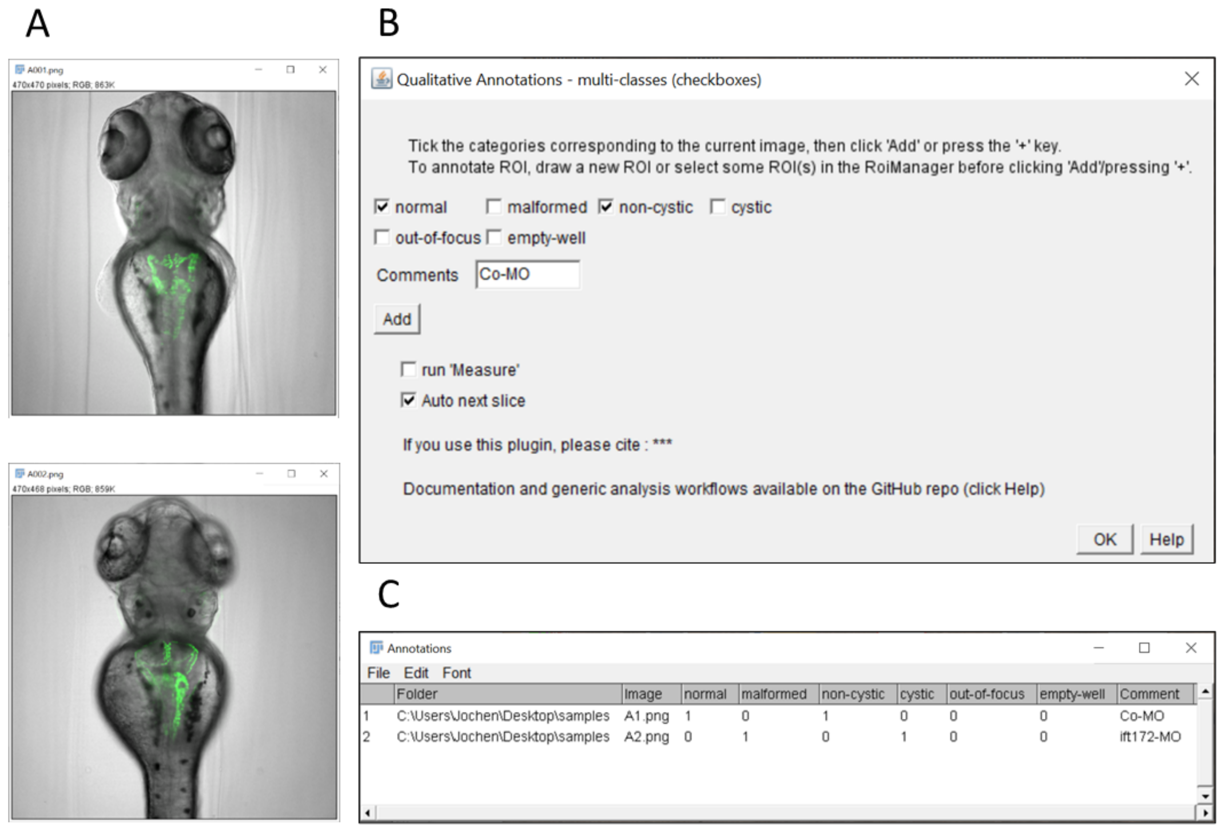
Documentation and generic analysis workflows available on the GitHub repo (click Help)

	Folder	Image	Slice	Category	Comment
1	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:25	Metaphase	
2	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:31	Anaphase	
3	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:41	Telophase	

	Folder	Image	Slice	Prophase	Metaphase	Anaphase	Telophase	Comment
1	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:25	0	1	0	0	
2	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:31	0	0	1	0	
3	C:\Users\Laurent Thomas\Desktop	mitosis.tif	C:2,Z:4,T:41	0	0	0	1	

Figure 22: The *single-class (buttons)* qualitative annotation plugin

At the top is the graphical interface of the plugin. As the plugin's name suggests, it allows the assignment of a single category per image or image-region by clicking the associated button. Here the plugin is illustrated for the annotation of the mitotic stage of dividing cells (ImageJ example image "mitosis" – credits NIH). Below the user-interface is the resulting result table, with either a single category column (middle) or multiple category column with binary encoding (bottom), depending on the initial plugin configuration. Own contribution – licenced under CC-BY.



Qualitative Annotations - multi-classes (checkboxes)

Tick the categories corresponding to the current image, then click 'Add' or press the '+' key.
To annotate ROI, draw a new ROI or select some ROI(s) in the RoiManager before clicking 'Add'/pressing '+'.

☑ normal ☐ malformed ☑ non-cystic ☐ cystic
☐ out-of-focus ☐ empty-well

Comments: Co-MO

☐ run 'Measure'
☑ Auto next slice

If you use this plugin, please cite : ***

Documentation and generic analysis workflows available on the GitHub repo (click Help)

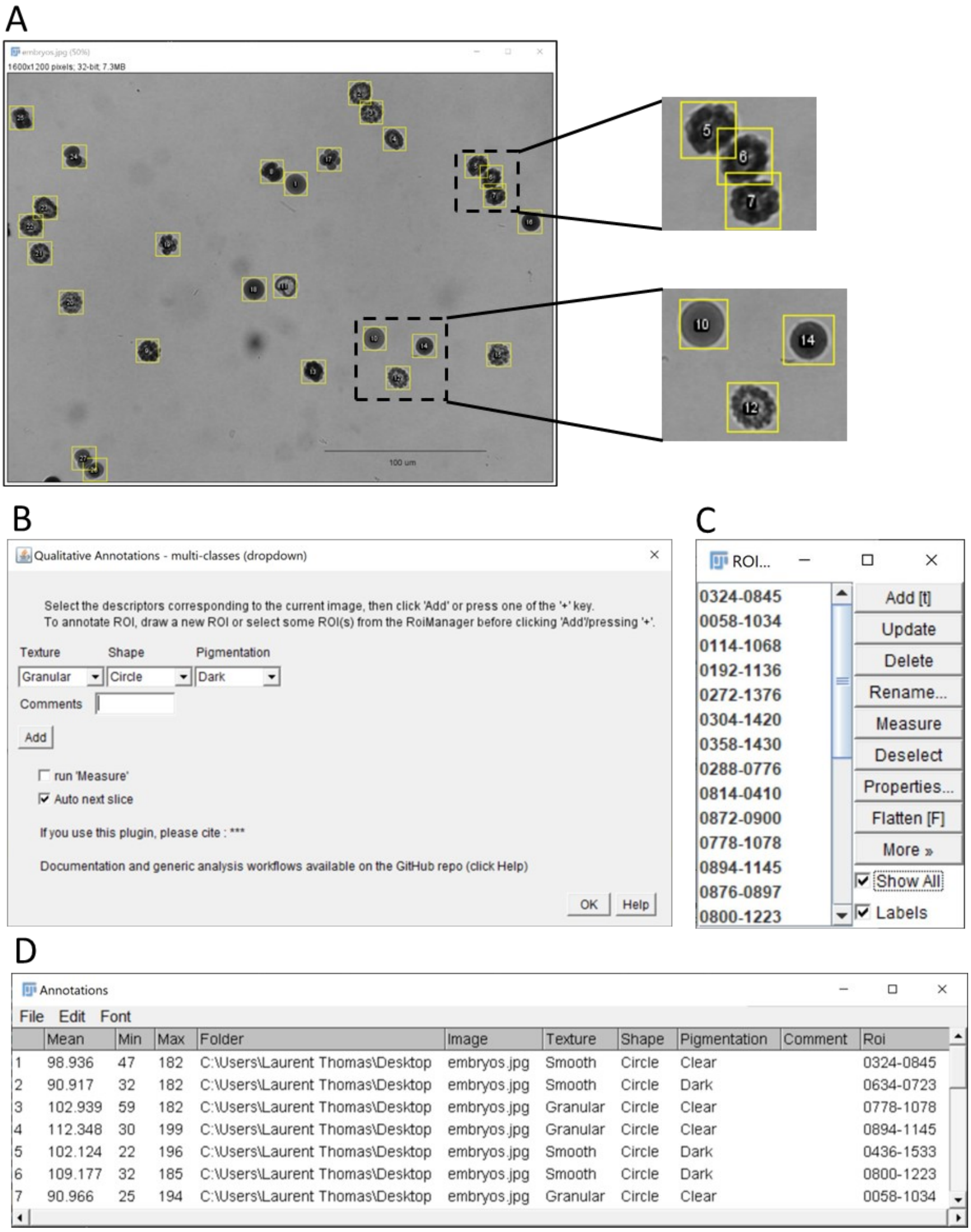
	Folder	Image	normal	malformed	non-cystic	cystic	out-of-focus	empty-well	Comment
1	C:\Users\Jochen\Desktop\samples	A1.png	1	0	1	0	0	0	Co-MO
2	C:\Users\Jochen\Desktop\samples	A2.png	0	1	0	1	0	0	ift172-MO

Figure 23: Annotation of multiple categories using the *multi-class (checkboxes)* plugin

(A) Example images of transgenic zebrafish larvae of the *Tg(wt1b:egfp)* transgenic line after injection with control morpholino (upper panel) or with ift172 morpholino (lower panel) inducing pronephric cysts. In this illustration, the plugin is used to score overall morphology and cyst formation. It could also be used to mark erroneous images (such as out-of-focus or empty wells). Images are from (Pandey et al. 2019).

(B) Graphical interface of the checkbox annotation plugin configured with 2 checkboxes for overall morphology, 2 checkboxes for presence of pronephric cysts, and checkboxes to report out-of-focus and empty wells. Contrary to the *single class (button)* plugin, multiple categories can be assigned to a given image.

(C) Resulting multi-category classification table with binary encoding of the annotations (True/False).



Qualitative Annotations - multi-classes (dropdown)

Select the descriptors corresponding to the current image, then click 'Add' or press one of the '+' key.
To annotate ROI, draw a new ROI or select some ROI(s) from the RoiManager before clicking 'Add'/pressing '+'.

Texture: Granular
Shape: Circle
Pigmentation: Dark

Comments

Add

□ run 'Measure'

☑ Auto next slice

If you use this plugin, please cite : ***

Documentation and generic analysis workflows available on the GitHub repo (click Help)

OK Help

ROI...

0324-0845
0058-1034
0114-1068
0192-1136
0272-1376
0304-1420
0358-1430
0288-0776
0814-0410
0872-0900
0778-1078
0894-1145
0876-0897
0800-1223

Add [t]
Update
Delete
Rename...
Measure
Deselect
Properties...
Flatten [F]
More »
☑ Show All
☑ Labels

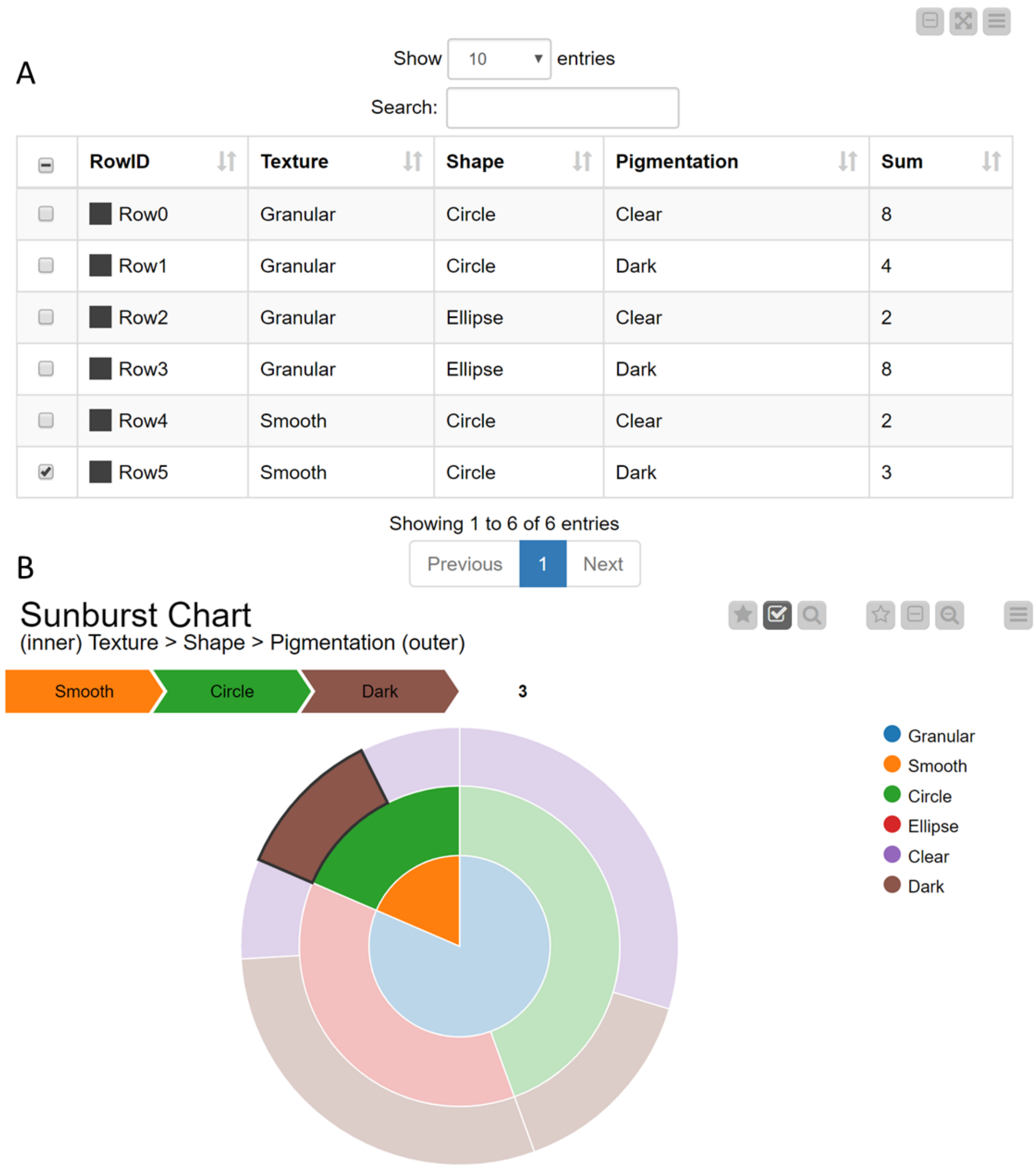
Annotations

File Edit Font

	Mean	Min	Max	Folder	Image	Texture	Shape	Pigmentation	Comment	Roi
1	98.936	47	182	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Smooth	Circle	Clear		0324-0845
2	90.917	32	182	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Smooth	Circle	Dark		0634-0723
3	102.939	59	182	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Granular	Circle	Clear		0778-1078
4	112.348	30	199	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Granular	Circle	Clear		0894-1145
5	102.124	22	196	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Smooth	Circle	Dark		0436-1533
6	109.177	32	185	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Smooth	Circle	Dark		0800-1223
7	90.966	25	194	C:\Users\Laurent Thomas\Desktop	embryos.jpg	Granular	Circle	Clear		0058-1034

Figure 24: Qualitative and quantitative annotations of image regions using the *multi-class (dropdown)* plugin

(A) ImageJ's sample image "*embryos*" after conversion to grayscale using the command *Image > Type > 32-bit*. The embryos outlined with yellow regions of interest were annotated using the "*multi-class (dropdown)*" plugin. The insets at the top shows the annotation of overlapping ROIs, here corresponding to embryos with phenotype granular texture, dark pigmentation and elliptic shape. The inset at the bottom shows other embryos with different phenotypes (10: smooth/clear/circular, 12: granular/clear/elliptic, 14: smooth/dark/circular). **(B)** Graphical interface of the multi-class (dropdown) plugin. Three exemplary features are scored for each embryo: texture (granular, smooth), shape (circle, ellipse) and pigmentation (dark, clear). Quantitative measurements as selected in the *Analyze > set Measurements* menu (here Mean, Min and Max grey level) are also reported for each embryo, when the *run Measure* option is ticked. **(C)** ROI Manager with ROIs corresponding to annotated regions. **(D)** Resulting classification table with the selected features, qualitative measurements, and associated ROI identifiers for the outlined embryos.



⊟	RowID	Texture	Shape	Pigmentation	Sum
□	■ Row0	Granular	Circle	Clear	8
□	■ Row1	Granular	Circle	Dark	4
□	■ Row2	Granular	Ellipse	Clear	2
□	■ Row3	Granular	Ellipse	Dark	8
□	■ Row4	Smooth	Circle	Clear	2
☑	■ Row5	Smooth	Circle	Dark	3

Figure 25: Visualizing qualitative data-distribution using sunburst charts in KNIME

(A) Summary table derived by the workflow from the annotation table (Figure 24.D), reporting the occurrence for each combination of qualitative features occurring in the dataset. The number of samples with a particular combination of features is reported in the column “Sum”. **(B)** Resulting sunburst chart with each qualitative feature represented as a concentric circle. The concentric order of the feature circles (inner to outer) can be adapted by the user. The sunburst chart allows the comparison of data-distribution for qualitative features, and to identify trends between features. For instance, the highlighted portion shows that all specimens with smooth appearance also have a circular shape, and most of them a rather dark pigmentation.

3.3. Object detection with Multi-Template-Matching

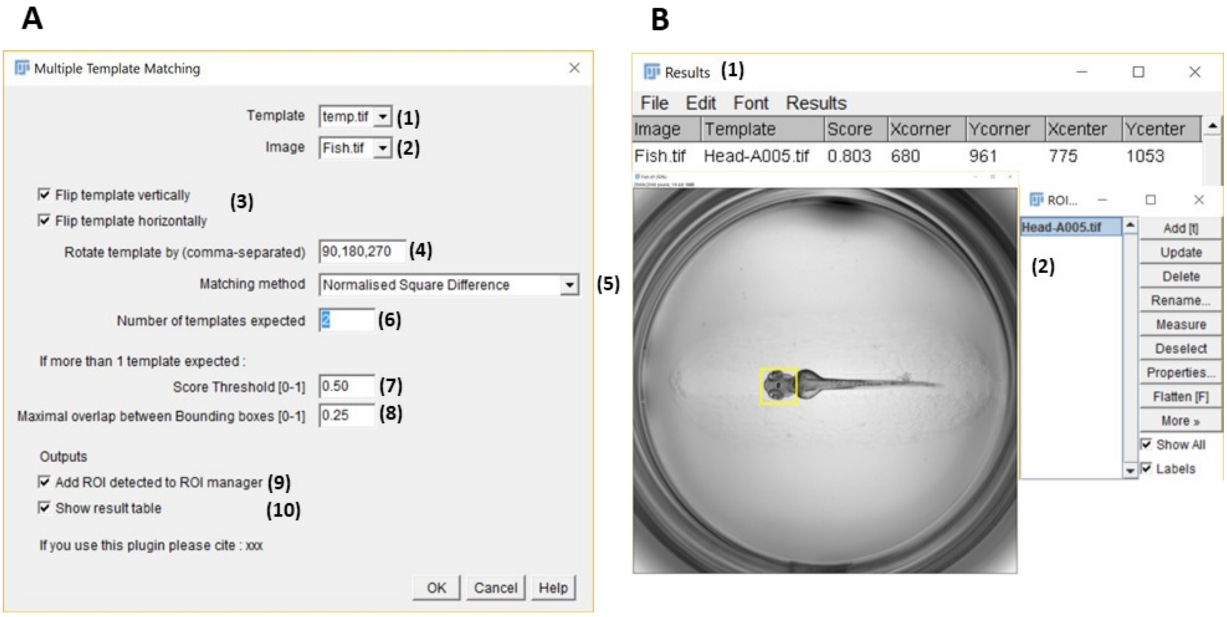
This section describes a novel implementation and benchmarking of multi-template-matching for object-detection, with enhanced detection capacities compared to the existing single-template matching. Part of the results described here are also detailled in the associated publication (Laurent S. V. Thomas and Gehrig 2020) and online YouTube tutorials (Thomas 2019b). The method was benchmarked for the localization of small organisms such as unhatched medaka embryos, as well as specific organs in zebrafish larvae. The workflow reported in section 3.3.8: "Application to eye and lens segmentation in zebrafish" was implemented by Alexandre Jeanne, master student at the time of a 2-month internship at ACQUIFER, under supervision of Laurent Thomas and Jochen Gehrig.

3.3.1. Implementation

To overcome the limitations of object-recognition with single-template matching (see section 1.6.3), a novel multi-template matching (MTM) was implemented as plugins in Fiji (Figure 26) and as a python package. Using the python implementation, a KNIME workflow was also developed (Figure 27). The implementation allows the search with multiple template images, corresponding to rotations or other transformations of the object. The core functionalities of MTM are the computation of the score map for each template, the detection of local score extrema from the set of score maps, and finally the filtering of overlapping detections via a custom non-maxima suppression strategy (Figure 29 and method section 2.4). A flowchart summarizing the implemented Multi-Template-Matching is depicted in Figure 28. A graphical user-interface (GUI) in Fiji and KNIME allows the setting of the template pre-processing (optional flipping and rotation) and detection parameters (Figure 26, Figure 27). In the plugins' GUI, only flipping and rotations of the templates are proposed as possible transformations, which are the most common transformations expected for objects in microscopy images. If flipping and rotation are selected, both the initial and flipped templates are rotated. Scaling of the templates would be a possible supplementary transformation, it is however not proposed in the interface, as in the context of microscopy, objects are expected to lie in the same size-range for a given objective magnification. However, template images representing objects of different scales can still be provided as a list of files. The Fiji plugins are macro-recordable to allow their integration in custom analysis workflows. Besides, extensive documentation and tutorials are available on the GitHub repositories (Thomas 2019a).

3.3.2. Execution of the search

The execution of the algorithm is relatively straightforward: the template (or templates) and target images are provided via the built-in image loading functionalities of Fiji, python or KNIME. For batch-processing in Fiji, list of files or a folder containing the images can be used as input too. To generate additional artificial templates, geometric transformations of the provided templates can be selected via checkboxes for flipping, and a list of angles can be entered for rotation by discrete angles. A rectangular region of interest can be provided to limit the search to a portion of the image, which has the advantage to speed up the search (Figure 30.C). Then the user should set up the detection parameters, namely the type of score for the computation of the score map, the expected number of objects (if known) and the score and overlap thresholds. The parameters are saved in memory for the next execution. After execution, the plugins return either rectangular ROIs in Fiji (Figure 26.B), a mask image in KNIME (Figure 27.C) and a list of detections with bounding-box coordinates and scores in python. The proposed implementation is fully integrated within the executing software architecture, such that the ROIs or mask images can be exploited for further analysis in Fiji and KNIME respectively. Besides, the Fiji plugin is macro-recordable, to facilitate its reutilization in custom image-processing workflows. Finally, the result table containing the template names and scores can be used for classification or ranking of the detections.



Template temp.tif (1)

Image Fish.tif (2)

☑ Flip template vertically (3)

☑ Flip template horizontally

Rotate template by (comma-separated) 90,180,270 (4)

Matching method Normalised Square Difference (5)

Number of templates expected 2 (6)

If more than 1 template expected :

Score Threshold [0-1] 0.50 (7)

Maximal overlap between Bounding boxes [0-1] 0.25 (8)

Outputs

☑ Add ROI detected to ROI manager (9)

☑ Show result table (10)

If you use this plugin please cite : xxx

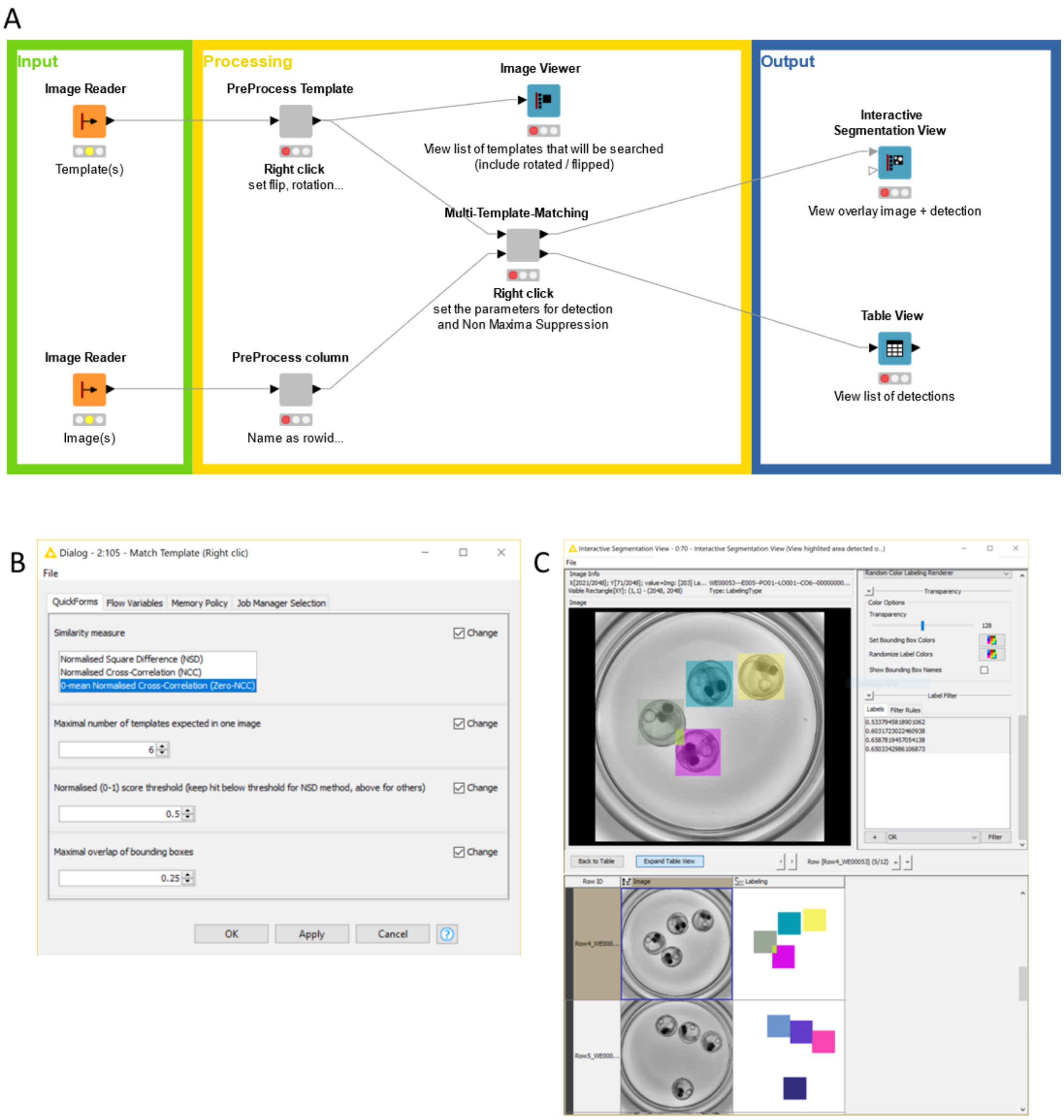
Image	Template	Score	Xcorner	Ycorner	Xcenter	Ycenter
Fish.tif	Head-A005.tif	0.803	680	961	775	1053

Head-A005.tif (2)

Figure 26: The Multi-Template Matching Fiji plugin

A) Graphical user-interface for the plugin "Template Matching Image" with: (1) Dropdown menu to select the template image of the object of interest. The template must be smaller than the image specified in 2, (2) dropdown menu to select an image (or stack of images) in which to search for the template, (3) tick-boxes to optionally generate additional templates by horizontal/vertical flipping of the initial template, (4) input field for rotation angles to generate additional templates by rotations of the initial and, if selected, flipped templates. The angles are specified in degrees with clockwise orientation and must be separated by commas, (5) dropdown menu to choose the score used for the computation of the score map (normalised square-difference, normalised cross-correlation or 0-mean normalised cross-correlation), (6) input field to specify the number of objects expected in the image, (7) input field to enter a score-threshold in the range 0-1. If the normalised square-difference is selected, only local minima with values below the threshold are returned. While for cross-correlation scores, maxima above this value are returned, (8) input field to specify the maximum value in range 0-1 for the intersection over union (IoU) between a pair of overlapping bounding boxes (Non-Maxima Suppression), (9) tick-box to select if the detected Regions Of Interest (ROI) should be added to Fiji ROI Manager, (10) tick-box to specify if the result table should be displayed at the end of the execution. Parameters 7 and 8 are only required if several objects are expected in each image.

B) Outputs of the plugin with (1) result table with each row containing the names of the image and template, the prediction score and coordinates of the top left corner and centre of the predicted bounding box, and (2) the detected ROI appended to the ROI Manager and highlighted on the image (yellow).



Dialog - 2:105 - Match Template (Right clic)

File

QuickForms | Flow Variables | Memory Policy | Job Manager Selection

Similarity measure ☑ Change

- Normalised Square Difference (NSD)
- Normalised Cross-Correlation (NCC)
- 0-mean Normalised Cross-Correlation (Zero-NCC)

Maximal number of templates expected in one image ☑ Change

6

Normalised (0-1) score threshold (keep hit below threshold for NSD method, above for others) ☑ Change

0.5

Maximal overlap of bounding boxes ☑ Change

0.25

OK Apply Cancel

Figure 27: Multi-Template-Matching in KNIME

A) Screenshot of the KNIME workflow. The template and images are provided in the Image Reader nodes on the left side, the processing happens in the central component node called 'Multi-Template Matching' containing a python node calling the python implementation. The parameters for multi-template matching can be configured via a graphical user-interface (see B) by right clicking on the node. The predicted locations can be visualised in the Interactive Segmentation View node on the right side (as shown in C). A result table containing the bounding box position, dimensions and correlation score is also returned (Table view node, output not shown). **B)** Graphical user-interface of the central "Multi-Template Matching" component node for the configuration of the detection parameters, similarly to the Fiji implementation. **C)** Predicted locations as viewed in the Interactive Segmentation View node. The predicted locations are composed into a mask and overlaid on the image.

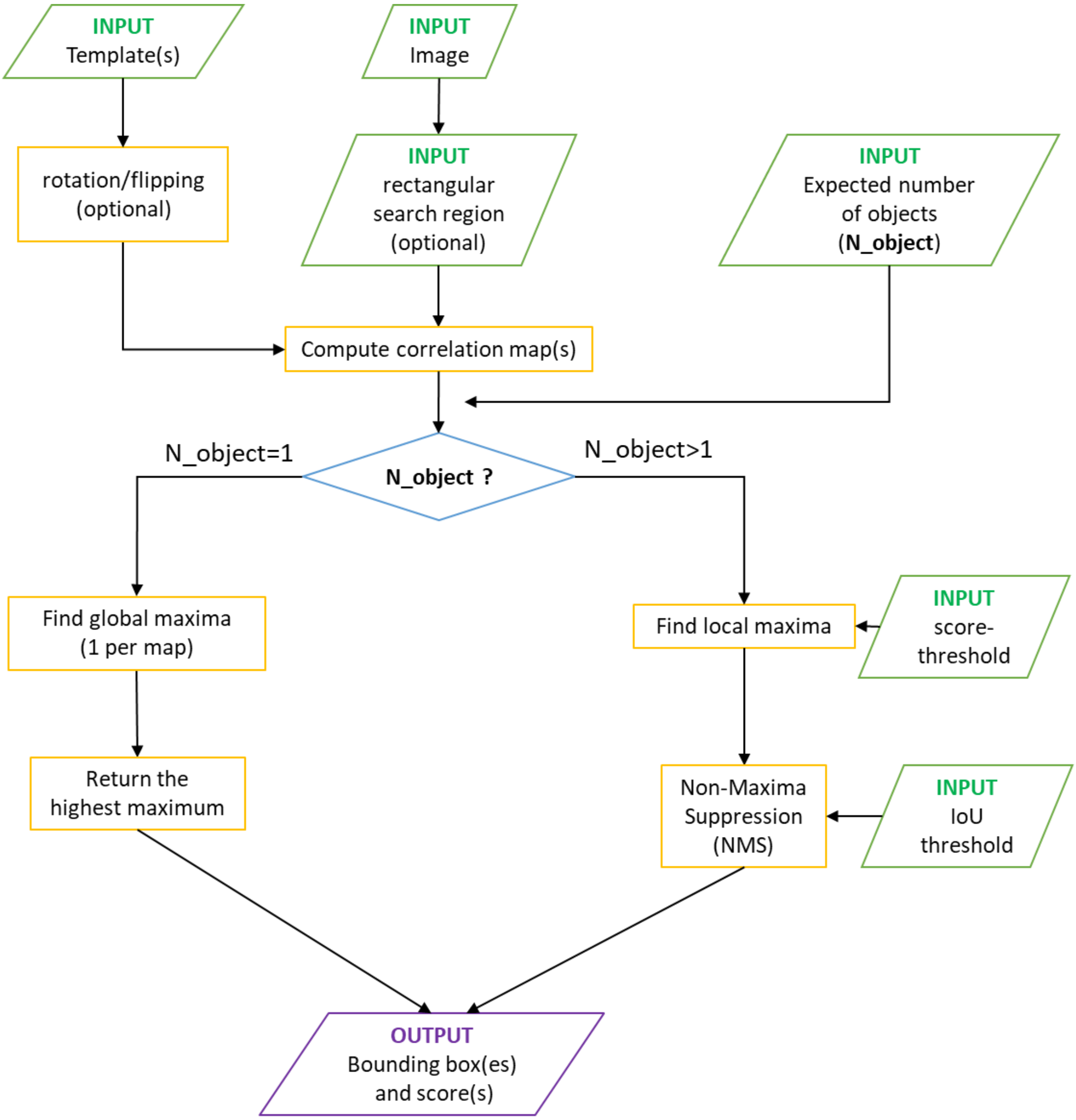


Figure 28: Flowchart of the implemented Multi-Template-Matching

The chart illustrates the execution sequence when a correlation score was selected. For difference-based score, the pipeline is identical except that a difference map is computed, minima are detected instead of maxima and the lowest minima are returned. (IoU: Intersection over Union)

3.3.3. Overlap-based Non-Maxima Suppression

Like other sliding-window search algorithms, template matching is likely to detect a single-object several times at slightly shifted locations. Redundant detections are even more probable when using multiple templates, which are likely to match similar image-regions (Figure 29.C). The predicted object-positions are thus not a simple sum of the detections from each template. To remove redundant detections, an overlap-based non-maxima suppression (NMS) strategy was adopted (see section 1.5.3 for introduction about NMS and section 2.4.7 for the implementation). The NMS is executed after the detections from the individual template searches are collected, and remove detections overlapping above a user-defined threshold with higher-score detections. The overlap is computed between pairs of bounding-boxes as the ratio between the intersection and union area of the bounding-boxes (Intersection over Union – IoU). Figure 29.E-D illustrates the intersection and union area for a pair of overlapping bounding-box. From the bounding-box coordinates, the corners of the rectangle corresponding to the intersection can be derived and the corresponding area calculated. The union area is then calculated as *Union = Area_BBox1 + Area_BBox2 – Intersection*. The overlap threshold (max IoU) is a parameter of the detection, a default value of 0.5 is proposed. The NMS can be executed until a given number of objects are localized or for the full set of candidate bounding-boxes.

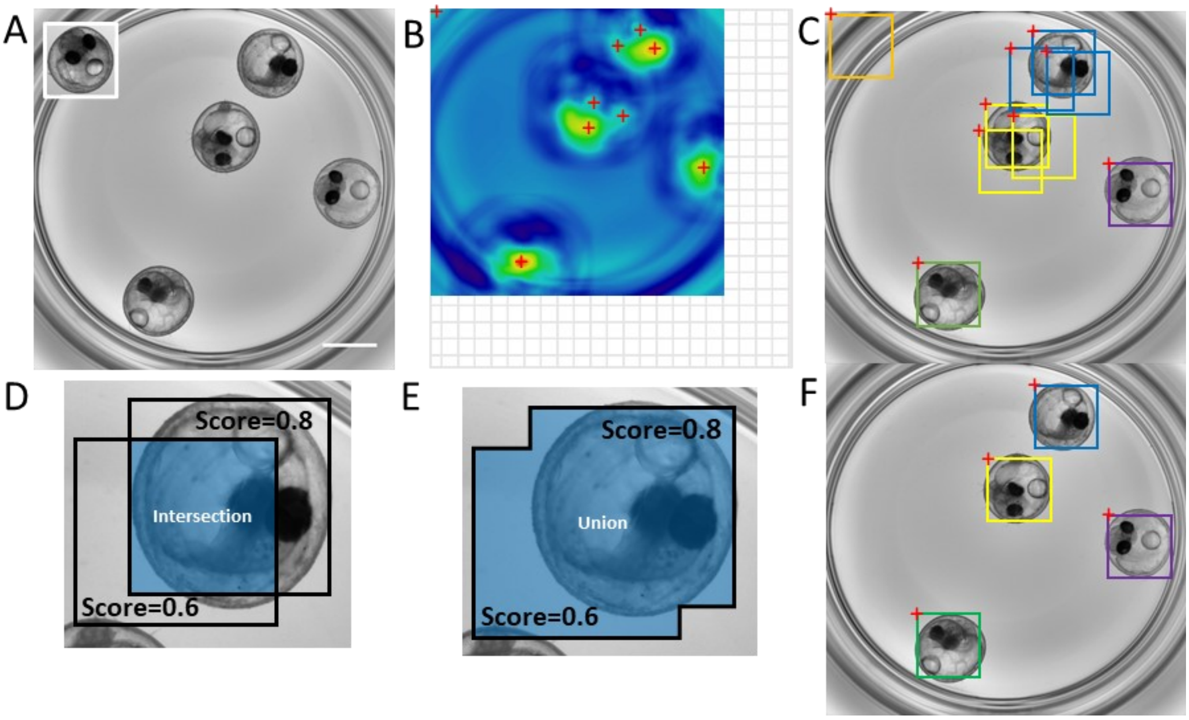


Figure 29: Multi-template matching and Non-Maxima Suppression for the detection of randomly oriented and positioned medaka embryos

A) Image in which the search is performed (2048x2048 pixels - scale bar: 1mm) and template as inset (400x414 pixels). **B)** One of the derived correlation maps from A: red crosses indicate possible local maxima before Non-Maxima Suppression (NMS). **C)** Bounding boxes associated to the maxima shown in B and overlaid on the searched image. Colours are highlighting overlapping bounding boxes. The bounding box dimensions are identical to the dimensions of the template used for the search. **D)** Intersection area (in blue) between 2 overlapping bounding boxes. Each bounding box is associated to a detection score. **E)** Area of the union between the overlapping bounding boxes. **F)** Resulting object detections after NMS with a maximal intersection over union (IoU) of 0.25, to return the N_objects=4 best detections.

3.3.4. Optional search-region

By default, the sliding-window search covers the full image. When the object is known to be localized within a fraction of the image, the search can be limited to this area. A simple approach would be to crop the image before running the search, however the resulting detections would there have pixel coordinates relative to the cropped image, which differ by a fixed offset from the coordinates in the original image. This might impact feedback microscopy applications which have to translate the original image coordinates to stage or objective coordinates. Therefore, to spare the need for additional calculations, the implemented template matching supports the definition of a rectangular search area. In Fiji, this is achieved by drawing a rectangular ROI before running the plugin, while in python an optional argument corresponding to the position and dimensions of the rectangular search region in the image (x, y, width height) can be provided. In those cases, the image is internally cropped, and the detections are automatically recalculated for the original image dimensions. Providing a search region has two benefits: it speeds up the search compared to the full-size image (Figure 30.C) and reduces the risk of false-positive detections of background regions.

3.3.5. Detection with a single template

For simple cases, object-detection can readily be achieved with a single-template. The implemented multi-template matching allows the search with a single template image when no template pre-processing is selected (no rotation nor flipping). This approach was successfully applied for the detection of the head region in microscopy images of dorsally oriented zebrafish larvae (see datasets section 2.2.2 - Figure 30). The head region was successfully detected in all cases, even upon slight morphological changes (Figure 30.B, e.g. well C7), altered specimen orientation (Figure 30.B, e.g. well B8) or partial occlusion (Figure 30.B, e.g. well E10). The detection with the Fiji implementation takes about half a second on a full resolution 2048x2048 pixels image (Figure 30.C). Thanks to the mounting of the specimen in agarose gel, the search can be limited to a fraction of the image, which noticeably reduces the time for the search (Figure 30.C).

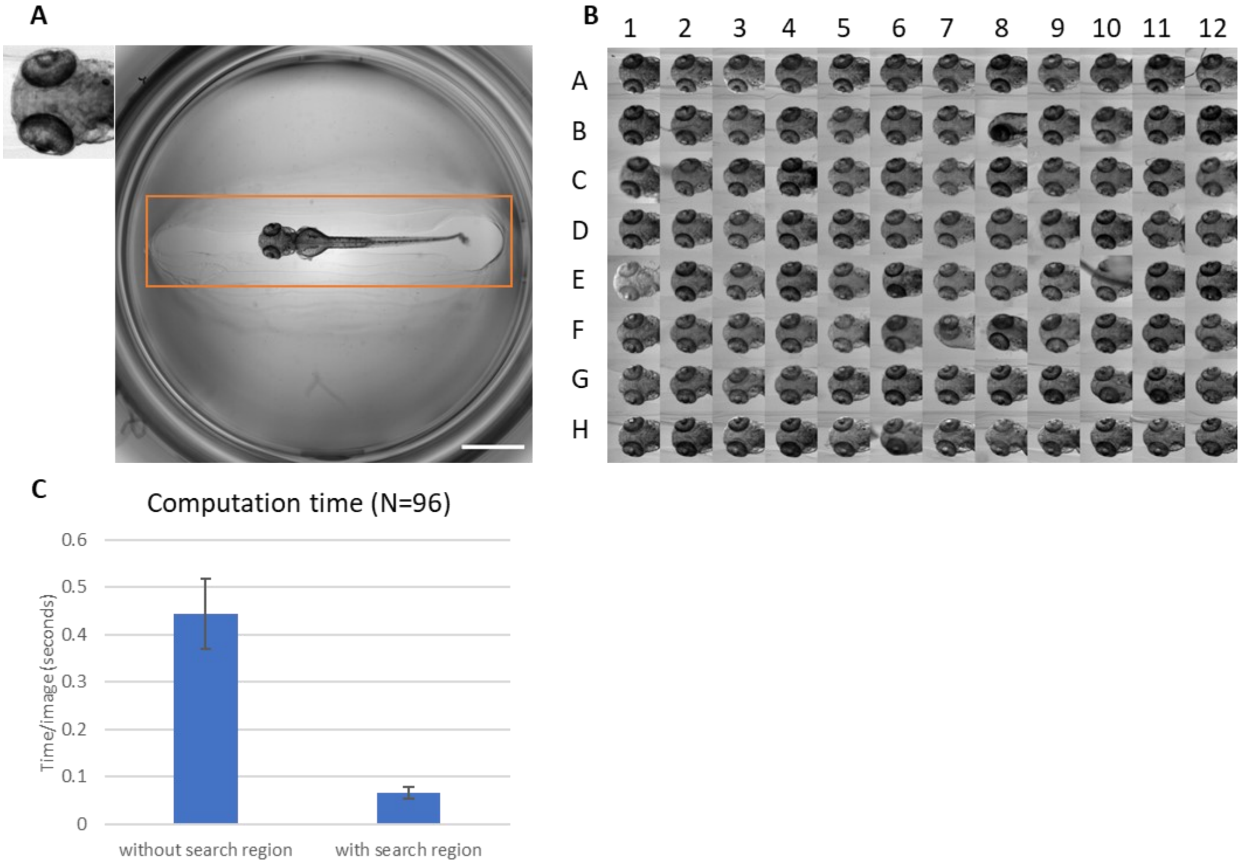


Figure 30: Single template matching for head region detection in zebrafish larvae

(A) Searched image (2048x2048 pixels, scale bar: 1mm) with template as inset (188x194 pixels), optional search region in orange (1820x452 pixels). **(B)** Montage of the detected head regions within a 96-well plate (100% detection success with or without search region). **(C)** Mean computation time per image (error bars show standard deviation – N=96 images) using the Fiji implementation of MTM on the laptop configuration. Prior information about the position of the sample within the field of view (e.g. provided a standardized sample mounting) can be used to specify a search region, drastically accelerating the computation and reducing the chance of incorrect predictions

3.3.6. Detection with multiple templates

The real advantage of multi-template matching is the detection with multiple template images, which enhances the range of detectable patterns compared to a single-template search. This improved capacity is demonstrated in Figure 31 for the detection of unhatched medaka embryos (see dataset section 2.2.1), which, due to their spherical shapes, adopt random positions and orientations in wells of microtiter plates. Using MTM with a single template, the embryos are localized but with a major offset in Figure 31.A (yellow). While on a dataset of 10 images, half of the embryos are not found at all (Figure 31.B). Using additional templates generated by rotation and flipping of the original template image, all embryos are correctly localized and enclosed within the predicted bounding-boxes for the image in Figure 31.A (green). For the dataset of 10 images, the results are also improved: all embryos were localized and almost all of them are fully enclosed within the bounding-boxes (Figure 31.B, D). However, the search with multiple templates increases the computation time linearly (Figure 31.C : 0.25 sec x 12 templates = 3 secs).

3.3.7. Robustness to noise and occlusion

It was shown in Figure 30, that template matching performs relatively robustly in presence of experimental variability which degrades the expected object-intensity signature, such as mild specimen morphology changes, altered specimen orientation or partial occlusion. The impact of noise on the detection was also investigated. To do so, the results of MTM was compared between an image and a noise-corrupted version of the image, using the same template and parameters for the search (Figure 32 A,B). The resulting detections have almost identical positions in both cases, and the same template image matched a given object in both images (Figure 32 C,D). The template matching score is however reduced when the image is corrupted with noise, which is expected as the noise increases the divergence with the reference template intensities.

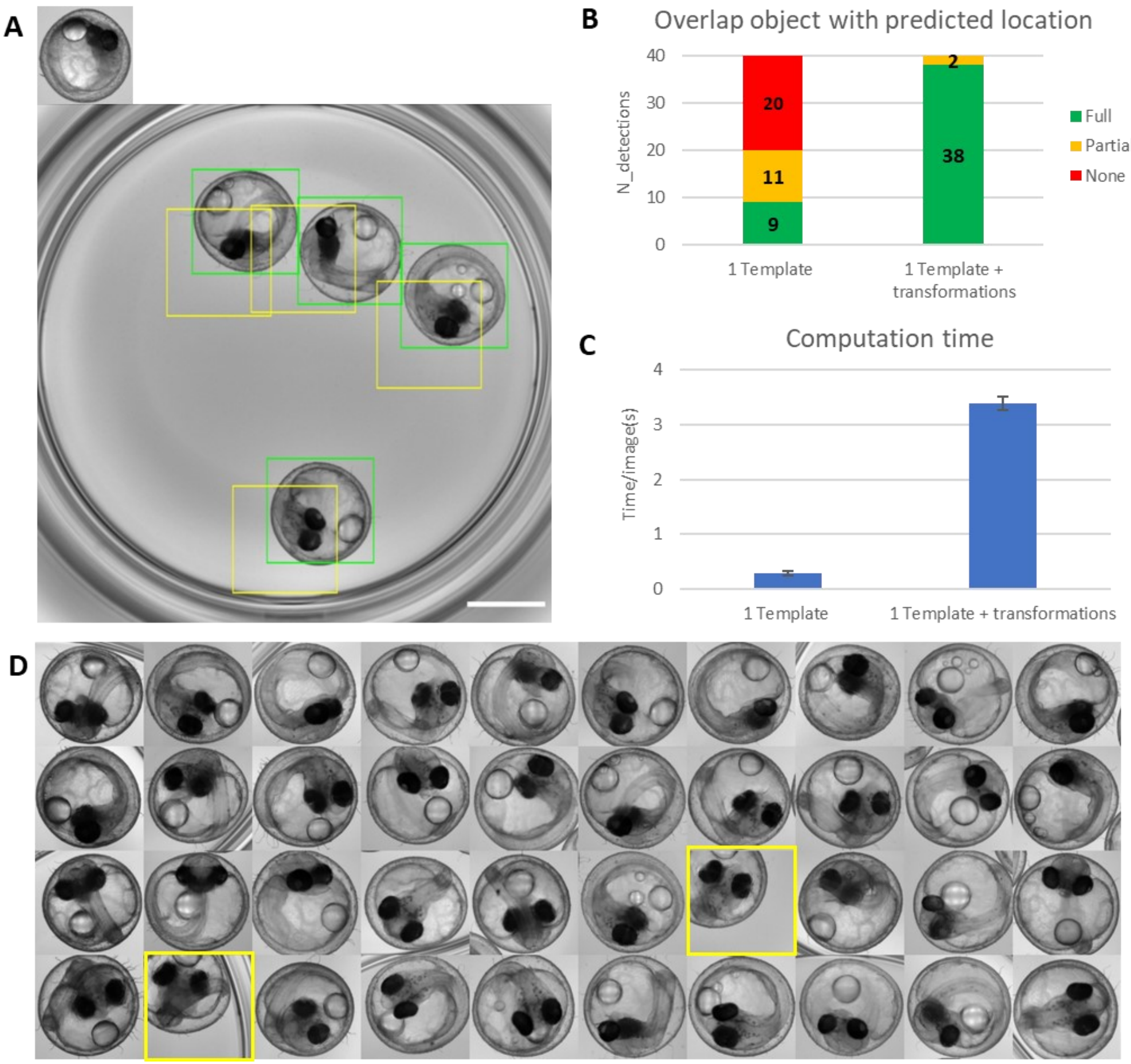
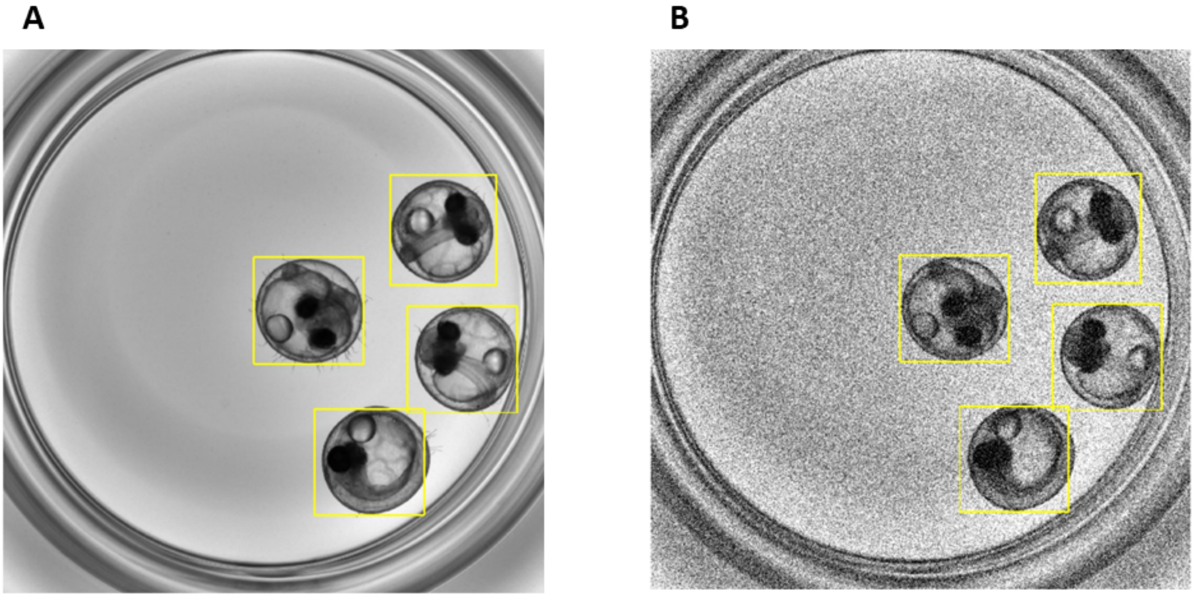


Figure 31: Multi-template matching for the localization of randomly oriented and positioned medaka embryos

(A) Initial template (410x420 pixels) and one of the images in which the search is performed (2048x2048 pixels, scale bar of 1 mm). The yellow bounding boxes indicate predicted locations when only the original template in A is used for the search, the green boxes indicate predicted locations when using a set of 12 templates (original, horizontal and vertical flipping, rotation of the original and flipped templates by 0°, 90°,180° and 270°). Parameters for the detection: score type: 0-mean normalized cross-correlation, N=4 expected objects per image, score threshold:0.35, maximal overlap between bounding boxes:0.25. **(B)** Result of the detections for 10 images each containing 4 embryos (i.e. 40 embryos in total). The objects are either predicted as fully enclosed within the detected Roi (full), partially enclosed (partial) or not detected (None). See detected region in D. **(C)** Mean computation time per image (error bars show standard deviation) for the different conditions using the laptop configuration. The computation time for each image scales with the number of templates. **(D)** Montage of the detected regions for 10 images as in A, each containing 4 embryos (1 column/image). The montage corresponds to the benchmark “1 Template+transformations” as in B and C. Yellow bounding boxes indicate the 2 detections classified as Partial in B.



C

Bounding Boxes (X,Y,Width,Height)	Score	Template rotation(°)
[946, 784, 414, 400]	1,00	0
[1525, 968, 414, 400]	0,59	180
[1173, 1354, 414, 400]	0,55	180
[1459, 474, 400, 414]	0,54	90

D

Bounding Boxes (X,Y,Width,Height)	Score	Template rotation(°)
[946, 784, 414, 400]	0,76	0
[1524, 968, 414, 400]	0,46	180
[1172, 1354, 414, 400]	0,44	180
[1459, 474, 400, 414]	0,43	90

Figure 32: Benchmarking effect of noise on the template matching search

(A) Original image (2048x2048 pixels). **(B)** Image as in A corrupted with artificial noise (normally distributed random noise – mean:0, standard deviation:50). **(C, D)** Result of multi-template matching for respectively A and B. The template used is a crop of the specimen in the middle of image A (hence a correlation score of 1 for the first row of table C). Parameters for the detection: rotation of the template: 90,180° - score type: 0-mean normalised cross-correlation - N=4 expected objects per image – score threshold: 0.3 – maximal overlap between bounding boxes: 0.3. The python code to reproduce the figure is available as a jupyter notebook on the GitHub repository https://github.com/multi-template-matching/MultiTemplateMatching-Python/blob/master/tutorials/NoiseBenchmark.ipynb .

3.3.8. Application to eye and lens segmentation in zebrafish

Single and multi-template matching have potential applications in image-analysis workflows as a generic pre-processing step to localize the regions of interest. This was demonstrated for the detection of the eye and lens in zebrafish larvae with a custom workflow performing a 2-step template matching for the robust localization of the eye, followed by edge detection and segmentation (Figure 33 – see method in section 2.4.8). This workflow could be used to automate fluorescence intensity measurements in the eye-region, as done for the “eye-assays” (Hanke et al. 2015) (see section 1.2.2). The results of the lens and eye detection are shown in Figure 34, with a test set of 24 images. The 1st template matching successfully localized the head in all except 1 image. The second template matching for the identification of the eye within the detected head region was successful for all 23 images. The lens was then outlined by edge detection with the Canny edge detector followed by circle detection from the edge map using the Hough circle transform. The lens was correctly outlined in 22 images out of 23 images and localized with a slight offset in 1 image. The eye was segmented by thresholding and ellipse-fitting on the resulting mask which was successful for 22 out of 23 images.

3.3.9. Integration in the ACQUIFER PlateViewer

To facilitate the application of template matching and the visualization of the results for large datasets, object-detection by template matching was also implemented as a plugin in the visualization software of the Acquifer microscope, the PlateViewer (see section 3.7.2, Figure 50, Figure 51). Together with the built-in functionality of the Plate Viewer, the plugin allows the localization and semi-automated imaging of a user-defined ROI automatically for each sample of the plate. This functionality is particularly adapted for the targeted imaging of specific tissues or organs in model organisms.

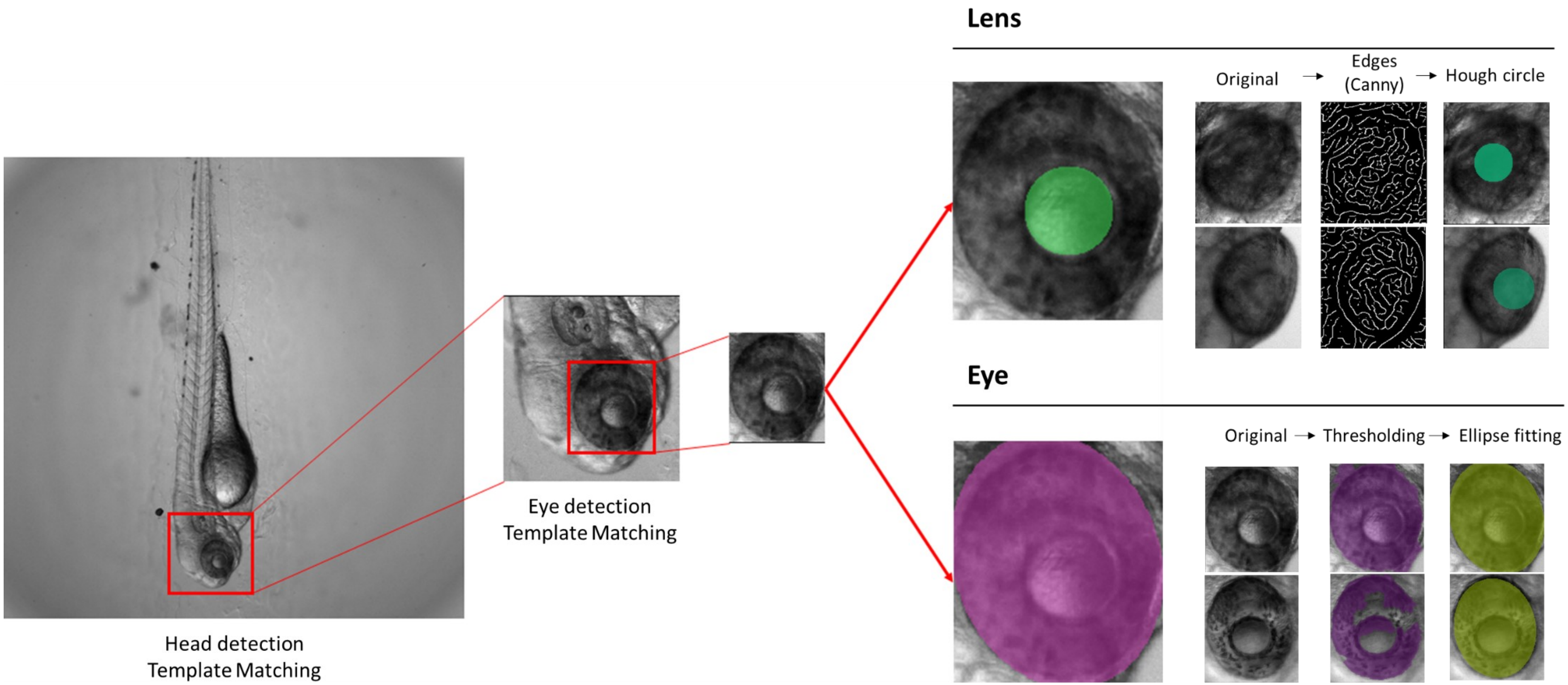
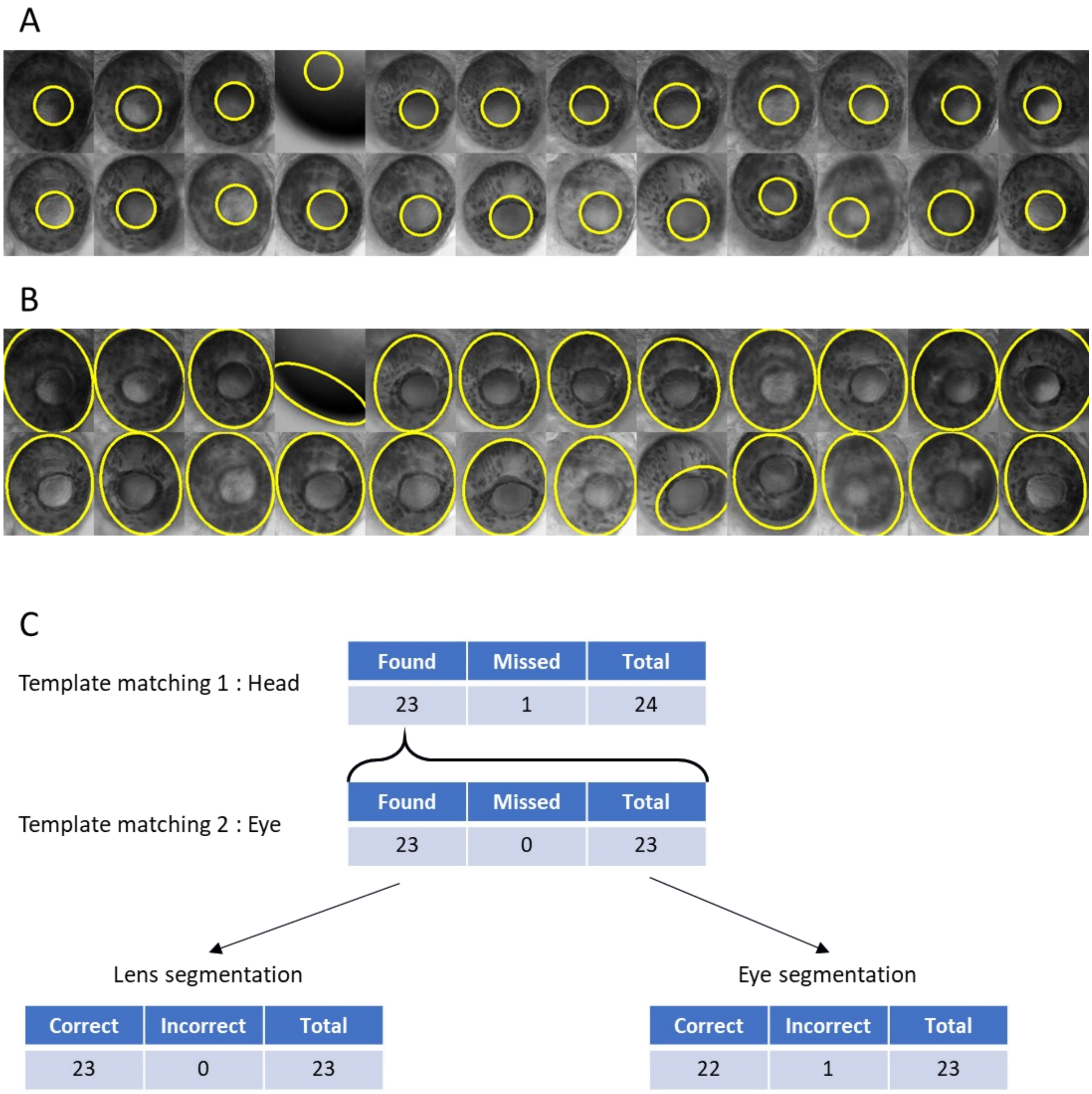


Figure 33: Using template matching in image-analysis workflows: eye and lens segmentation in zebrafish larvae

The eye-region is first localized in the image using a 2-step template matching. A first template search is used to localize the head region. A second template matching localizes the eye-region within the detected head-region. The eye and lens are finally segmented using classical image-processing methods, for instance segmentation and ellipse fitting. Zebrafish larvae were laterally oriented in wells of 96 well plate as in (Wittbrodt et al. 2014), and imaged at 4X magnification with the brightfield channel on an ACQUIFER IM04. This workflow and figure were contributed by Alexandre Jeanne, during his internship at ACQUIFER under supervision of Laurent Thomas and Jochen Gehrig.



Template matching 1 : Head

Found	Missed	Total
23	1	24

Template matching 2 : Eye

Found	Missed	Total
23	0	23

Lens segmentation

Correct	Incorrect	Total
23	0	23

Eye segmentation

Correct	Incorrect	Total
22	1	23

Figure 34: Results of the eye and lens segmentation in zebrafish using the custom 2-step template matching workflow

The detection was carried out with the workflow as in Figure 33 with N=24 zebrafish larvae, imaged at 4X magnification with the brightfield channel on an ACQUIFER IM04. The 4th image from the left in the top raw corresponds to a false-positive detection by the first template matching search. **A)** View of the segmented lens region as detected by canny edge detection and Hough circle transform overlaid on the eye-region detected after 2-step template matching. **B)** View of the outline of the eye as detected by intensity thresholding and ellipse fitting, overlaid on the eye region as in A. **C)** Sum up of the outcome for each detection step.

3.4. Object-detection with cascade detectors

This section describes the development of interactive annotations tools to generate ground-truth data compatible with the training of cascade detectors, and the implementation of a Fiji plugin for the detection with pre-trained cascade, customised to benefit from overlap-based non-maxima suppression. Finally, custom cascade detectors for zebrafish larvae and medaka embryos are also illustrated. The initial plugin version was implemented by Alexandre Jeanne under supervision of Laurent Thomas, and further developed with e.g. non-maxima suppression by Laurent Thomas.

3.4.1. Fiji-plugin for positive training set generation

While OpenCV releases (OpenCV team 2020) come with an interactive annotation tool to generate the positive training set for cascade detectors, this tool is not very convenient to annotate large amounts of images, neither is the display adapted for 16-bit grayscale images. Therefore, to facilitate ground-truth annotations for the training of cascade detectors with scientific images, a dedicated Fiji plugin was developed. The plugin should be executed after outlining the object of interest with rectangular ROIs, stored in the ROI Manager (see section 2.5.1 for help with the annotation). The ROIs should have aspect ratios close to the dimensions that will be used for the training, to prevent any deformation of the object upon downscaling (see section 1.7.4). Alternatively, the ROI 1-click tools can be used to quickly generate ROIs of predefined dimensions (see section 3.2.2).

Once all object regions are annotated for the current image or image-stack, the plugin can be executed. For each object-region outlined by a ROI, the plugin will first verify that the aspect-ratio matches the dimensions that will be used for the cascade training. When this is the case, the outlined image-region is cropped, downscaled to those dimensions, and saved as an image file to a designated folder on disk. If the aspect-ratio does not match the training dimensions, a window is prompted requesting the user to either skip the ROI or to proceed as such. Data-augmentation by flipping and/or rotation of the outline image-regions is also proposed in the GUI of the Fiji plugin to generate additional training images. Besides the training images, a pre-formatted text file (info.dat) with the file-paths to the training images can be generated, which is necessary when training a cascade directly via the command-line. The resulting image files can then be used to train a cascade with the GUI-based cascade training software of (Ahmadi 2017) or using the OpenCV command line training executable.

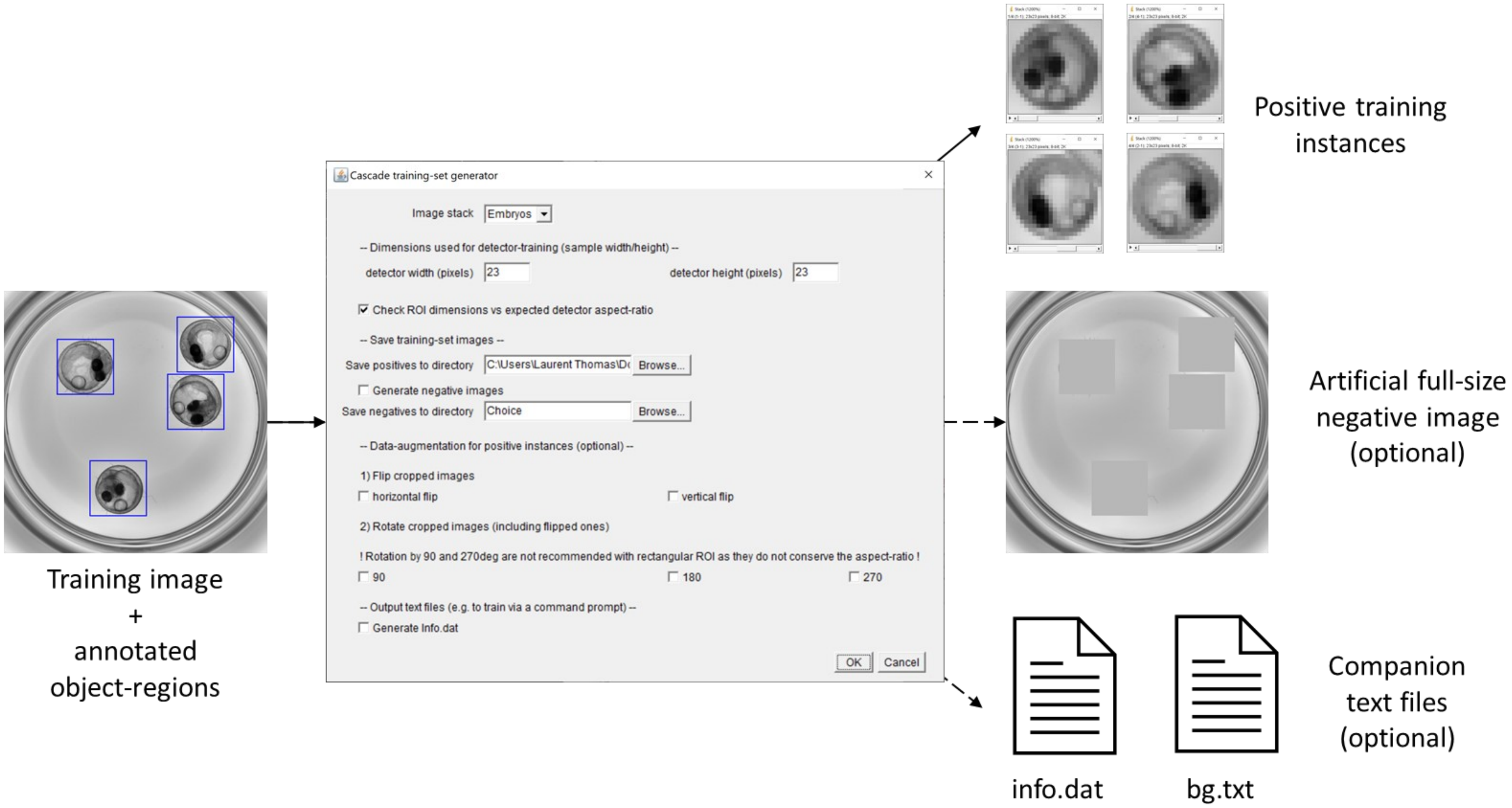
3.4.2. Generation of artificial background images

For the training of cascade detector, the negative training set is composed of images representing the background, which are randomly cropped by the OpenCV cascade training executable to generate negative training instances. Obviously, those large images should not contain any object, to make sure the randomly cropped image-regions are free of the object-of-interest. For microscopy, if the samples are mounted in 96-well plates, these would correspond to images of empty wells (or filled with the medium), imaged with the same acquisition parameters than the experiment. In practice it is not very convenient to spare several wells of a plate to image the background exclusively. Therefore, the Fiji plugin was designed to provide an alternative to easily generate artificial background images suitable for the OpenCV random cropping routine. To do so, once the positive image-regions have been annotated and saved to disk, the outlined object-regions are replaced by the most common grey value of the image, and the resulting full-size image is saved to disk. A companion file bg.txt containing the file paths to the artificial images is also created, which is necessary when training with the command line utility. Examples of such background images are shown in Figure 35 and Figure 36. Although such artificial background images are not realistic, they still contain all possible non-object regions a sliding-window search could come across. This approach is suitable if the outlined object-regions do not represent a large fraction of the image area, to prevent the artificially filled area to have a major contribution to the actual training-set.

3.4.3. Generating hard negative training instances

When training cascade detectors, it was often observed that the model converges quickly during the training but performs poorly with lots of false-positive detections on test images. This suggests some divergence between the training set and the test set. While the positive training samples are manually outlined, the negative training instances are generated by the cropping routine of the OpenCV training executable. While the routine is convenient to yield a high number of negative training instances from a limited set of large negative images, there is little control of what the negative training set finally contains. We came with the hypothesis that the cropping routine may yield mostly background regions that are relatively straightforward to distinguish from the object. This could explain the rapid convergence of the training, as those regions are rapidly rejected within a few stages, but the poor performance on real images which contains a diversity of non-object regions which were not represented in the negative training set. This is likely to happen when the background images contain mostly uniform areas (e.g. the bottom of a well).

To prevent this behaviour, one can enforce challenging negative training instances by manually outlining background image-regions (bubbles, shadows, edges…). A second Fiji plugin, similar to the plugin for the positive training set, was thus developed for the annotation of negative image-regions. Again, rectangular Rois of identical aspect-ratio than the detection window should be used for the annotation of background regions. The outlined regions are also downscaled to the expected detector dimensions, which should effectively bypass further cropping by the OpenCV routine.



Cascade training-set generator

Image stack: Embryos

-- Dimensions used for detector-training (sample width/height) --

detector width (pixels) 23

detector height (pixels) 23

☑ Check ROI dimensions vs expected detector aspect-ratio

-- Save training-set images --

Save positives to directory C:\Users\Laurent Thomas\D(

☐ Generate negative images

Save negatives to directory Choice

-- Data-augmentation for positive instances (optional) --

1) Flip cropped images

☐ horizontal flip

☐ vertical flip

2) Rotate cropped images (including flipped ones)

! Rotation by 90 and 270deg are not recommended with rectangular ROI as they do not conserve the aspect-ratio !

☐ 90

☐ 180

☐ 270

-- Output text files (e.g. to train via a command prompt) --

☐ Generate Info.dat

Figure 35: Custom Fiji plugin to generate the training set for cascade detectors

See legend next page.

Figure 35: Custom Fiji plugin to generate the training set for cascade detectors

On the left is an example of image containing the objects of interest outlined with rectangular regions of interest. Upon execution of the custom plugin, a dialog window (middle) requests the dimensions expected for the cascade training and used to downscale the outlined regions, the output directory to save the positive training instances and the optional outputs. On the right are the outputs after execution. At the top are the positive training instances which are downscaled versions of the outlined regions. These are saved to the designated directory on disk. In the middle is the artificial background image, generated by replacing the outlined object regions with the most common grey value of the image. At the bottom, the txt and bat files contain informations about the image file-path and the locations of the objects in the image. Those files are necessary for the training via the OpenCV command line executable.

3.4.4. Zebrafish larvae detector

A custom cascade detector was trained for the detection of zebrafish larvae, dorsally oriented within wells of a 96-well plate, imaged with the Acquifer Imaging Machine at the lowest magnification (2X) (see method section 2.5.5). Figure 36.A shows example of positive object-annotations (specimen outlined in yellow), negative background annotations (red). The resulting training images after cropping and downscaling by the plugin are shown in Figure 36.C,D, while the artificial background image is shown in Figure 36.B. In total, 60 zebrafish larvae were outlined for the positive training set, and the corresponding 60 artificial background images were generated with the Fiji plugin. 29 additional negative regions were also manually outlined. From the 89 negative images, the training utility was configured to generate 300 negative images by random cropping. The training was set to reach a maximum of 5 stages, with a maximal false positive rate per stage of 0.35 and a minimum true positive rate per stage of 0.99. The training terminated after completing 2 stages in 5 min on the laptop configuration, as the number of negative training images was exhausted.

In the resulting trained cascade, the classifiers of both stages rely on 2 Haar-like features, which can be visualized overlaid on a training image in Figure 37. On a test set of 33 images, when the highest-score detection is retained, the larvae were correctly localized in 24 images, localized with a slight offset in 7 images, and not found in 2 images (Figure 39.A). The average detection time per image was around 40 milliseconds per image (Figure 39.C). When the expected number of object is not mentioned and only NMS is used, the previously undetected specimens are found but a false positive detection is also returned in addition to the specimen for 5 images (Figure 39.A).

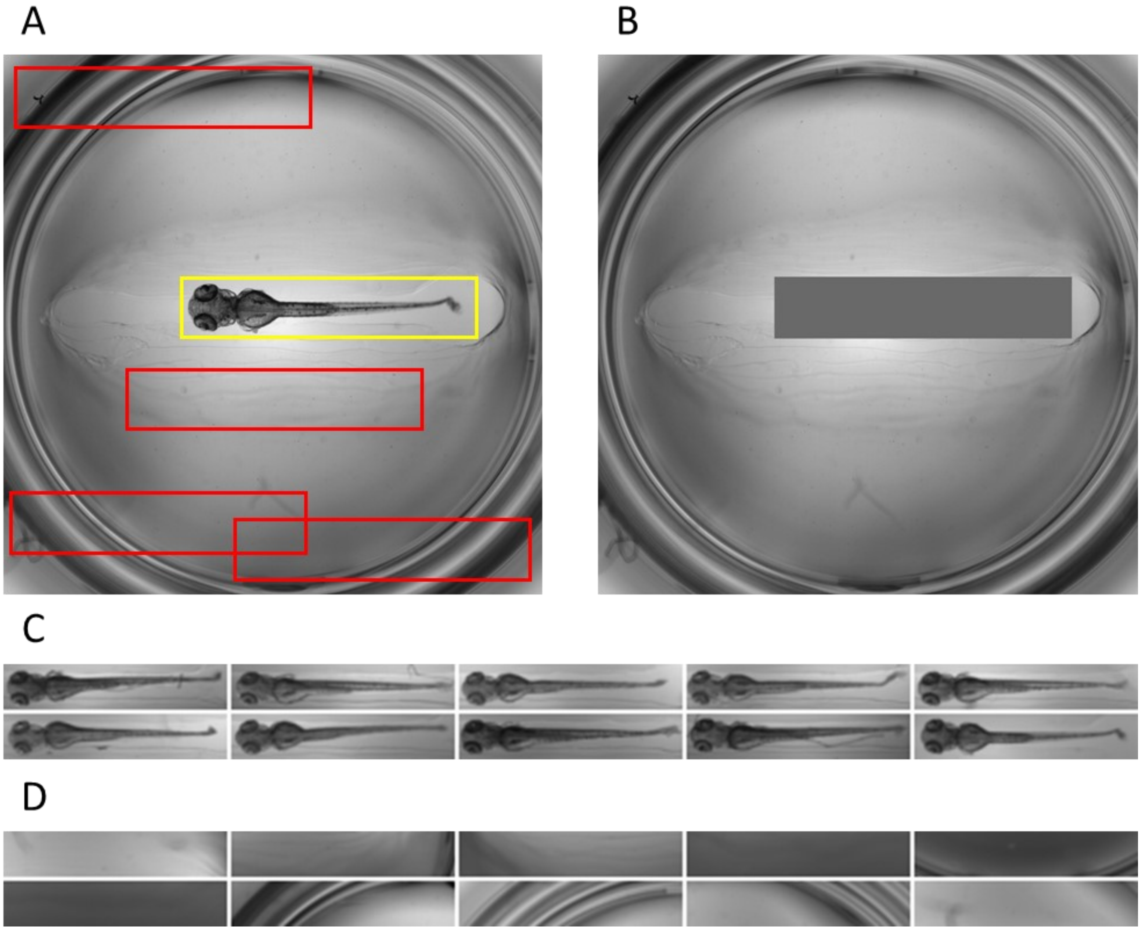


Figure 36: Ground truth annotations for the training of a zebrafish cascade detector

A) Example of image used to generate some training instances. The object of interest is outlined in yellow, and background regions annotated to force negative training instances are outlined in red. **B)** Artificial full-size negative image, generated by replacing the outlined object with the most common grey value of the original image. This image is used by the OpenCV training executable to generate negative training instances by multiple cropping at random locations. **C)** Corresponding training instances after cropping and downscaling to the detector size (113 x 23 pixels) with the positive annotation plugin in Fiji. **D)** Corresponding negative training instances, after cropping and downscaling with the negative annotation plugin. Those images are used for the negative training set in addition to the artificial image in C.

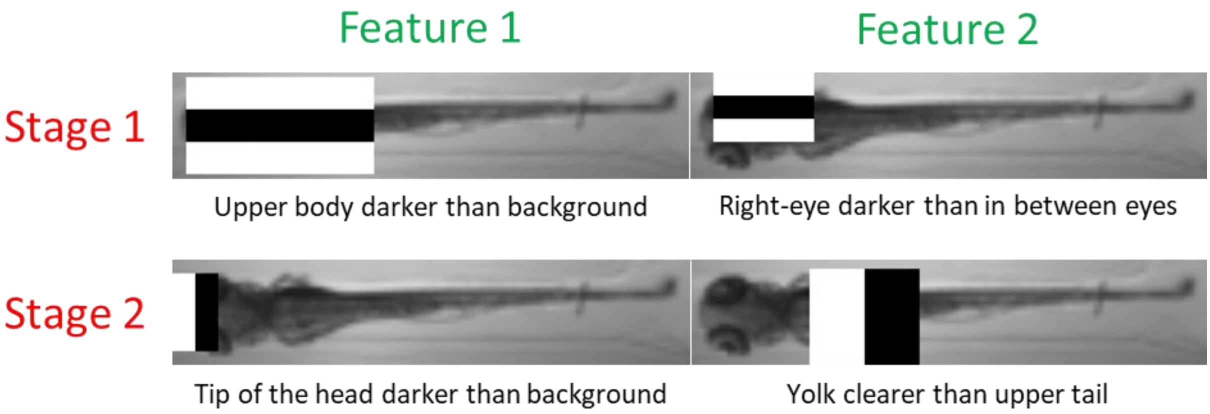
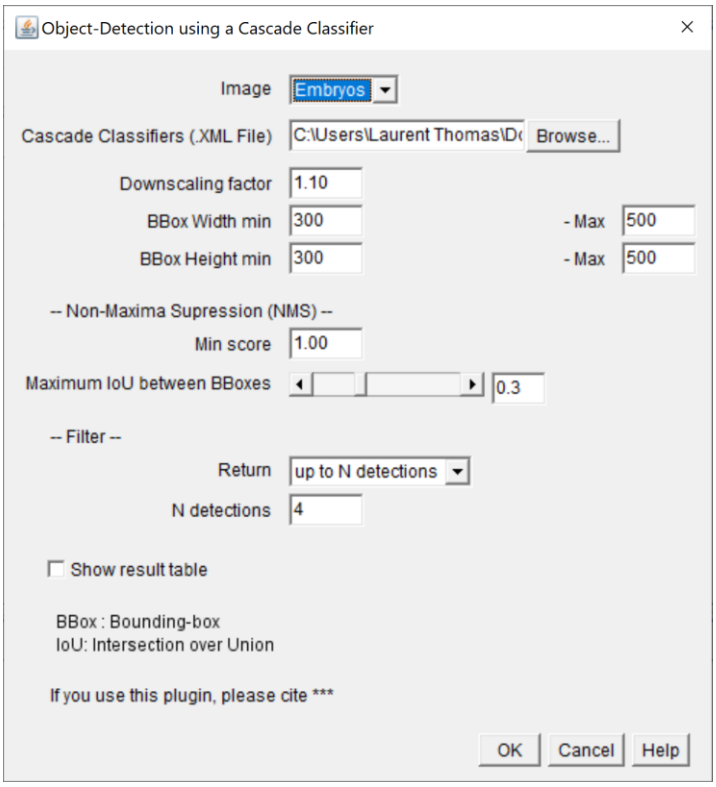

Feature 1
Feature 2
Stage 1
Upper body darker than background
Right-eye darker than in between eyes
Stage 2
Tip of the head darker than background
Yolk clearer than upper tail

Figure 37: Haar-like patterns retained during the training of a zebrafish larvae detector

The patterns are overlaid on a training image, scaled to the dimension used for the training (113 x 23 pixels). The cascade contains 2 stages, each of them relying on 2 Haar-like features. The intensity patterns potentially represented by the Haar-like pattern is indicated below the images. The overlay was generated by the *opencv_visualisation.exe* executable see method section 2.5.3.



Object-Detection using a Cascade Classifier

Image: Embryos

Cascade Classifiers (.XML File): C:\Users\Laurent Thomas\D(

Downscaling factor: 1.10

BBox Width min: 300 - Max: 500

BBox Height min: 300 - Max: 500

-- Non-Maxima Supression (NMS) --

Min score: 1.00

Maximum IoU between BBoxes: 0.3

-- Filter --

Return: up to N detections

N detections: 4

□ Show result table

BBox : Bounding-box
IoU: Intersection over Union

If you use this plugin, please cite ***

Figure 38: Graphical user interface of the cascade detection plugin in Fiji

The graphical user-interface allows selecting a custom cascade classifier for the detection of images in Fiji. The rest of the parameters concern the detection (downscaling factor and bounding-boxes dimensions), non-maxima suppression (min score, max IoU) and filtering of the list of detections (top-N detections).

3.4.5. Fiji-plugin for detection with trained cascade

Once a cascade trained for the detection of a particular object, the resulting xml file containing the detector structure allows the detection of objects in any grayscale image. To do so, the xml file can be loaded in custom scripts via the OpenCV library or in python scripts via the scikit-image python library (van der Walt et al. 2014). The graphical user-interface program from (Ahmadi 2017) also provides functionalities for the detection, however it is limited to a preview of the detections as bounding-boxes overlaid on the images, and thus cannot be included in larger image-analysis workflows. To provide an end-user solution for the detection with cascades, and to facilitate the integration of these detectors in custom image-processing pipelines, an additional Fiji plugin with graphical user-interface and compatibility with macro-recording was thus developed (Figure 38). It relies on the OpenCV library via the conversion utilities of the IJ-OpenCV package (Domínguez et al. 2017), similar to the Multi-Template-Matching plugins. The plugin can perform the detection on any image or image-stack opened in Fiji, provided the xml file of the cascade only. The GUI for the detection plugin contains parameters for the cascade, as well as for non-maxima suppression. The parameters related to the detection with the cascade include the minimal and maximal bounding boxes dimensions as well as the relative step-size for the image-pyramid generation. A smaller step-size yields a higher number of images in the pyramid and so a longer detection time. The minimal and maximal bounding-boxes dimensions likewise influence the range of scales covered by the image-pyramid, as for a given image-scale the cascade detector yields bounding-boxes of constant dimensions. The parameters for the NMS are unchanged compared to the Multi-Template Matching and include the minimum score to retain a potential detection and the maximum overlap between 2 candidate bounding-boxes. The outcome of the detection is similar than with the Multi-Template-Matching plugins. Predicted object-locations are returned as rectangular ROIs stored in the ROI manager. The resulting ROIs can thus directly be used for further processing. A result table with the detections coordinates and scores is also returned.

3.4.6. Medaka embryos detector

A cascade detector was similarly trained for the detection of medaka embryos in wells of 96 well plate (see method in section 2.5.7). The training lasted 1 min 42 secs and resulted in a 3-stage cascade. The results of the detections are detailed in section 3.4.7 and in Figure 39.B. When NMS is used and the number of expected object is mentioned, the cascade correctly localized 37 embryos, 1 embryo was localized with a slight offset and 2 embryos were not found (Figure 39.B - rightmost). The average detection time per image was 16 milliseconds with these settings (Figure 39.C).

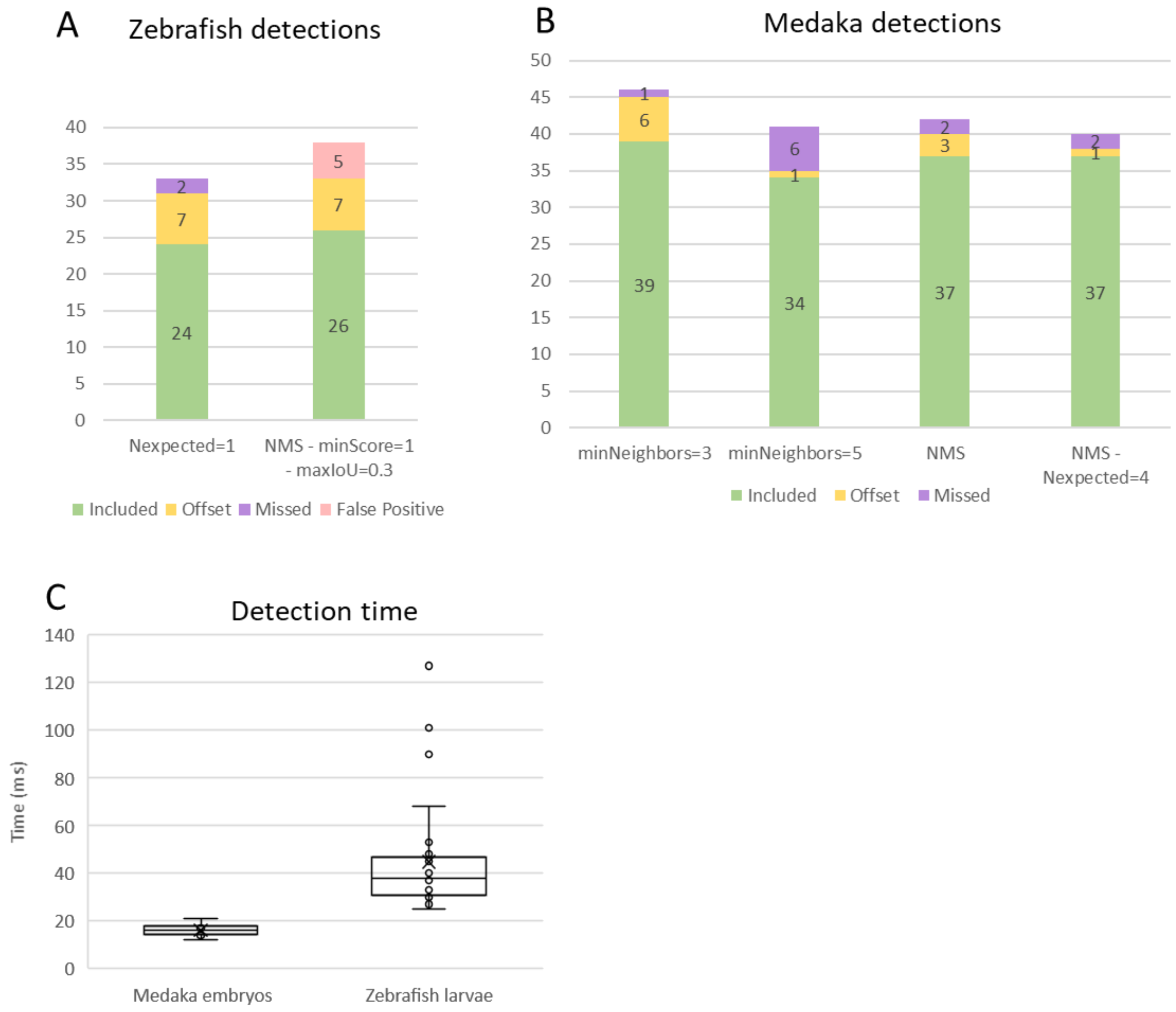


Figure 39: Benchmarking cascade detectors for microscopy

A) Localization of zebrafish larvae for 33 test images containing each 1 larva as in Figure 36. For A and B, the detections are reported as either fully enclosing the object (included) or partially enclosing the object (offset). Specimens not found are reported as "missed" detections, while "False Positive" corresponds to background detections. **B)** Localization of medaka embryos for 10 images containing 4 embryos each (40 embryos in total). The histogram reports the number of detections for different NMS strategy as in Figure 40. The 2 rightmost conditions (NMS) correspond to results with the proposed Fiji implementation. **C)** Individual detection times per image in Fiji using the laptop configuration on full resolution images (2048 x 2048 pixels). For zebrafish, the times corresponding to *Nexpected=1* as in panel A are reported. For medaka, the times for the condition *NMS – Nexpected=4* as in panel B are reported. The cross-shaped datapoints correspond to the average value.

3.4.7. Overlap-based NMS improves state-of-the-art cascade detectors

Object-detectors relying on sliding-windows typically yield overlapping detections, which require a non-maxima suppression (NMS) strategy to yield the most significative detections (see section 1.5.4). The cascade detection utility of OpenCV comes with a built-in NMS, relying on a single parameter: the minimum number of candidate bounding-boxes overlapping above 50% to return one genuine detection (*minNeighbors*). This approach is not ideal as it does not take into account the individual score of each bounding-box, such that multiple low-score candidates could yield a hit detection when highly overlapping, while a single high-score bounding-box might be discarded if not other bounding-box sufficiently overlaps. Besides, redundant detections overlapping by less than 50% will still be returned, and deciding on a value for the *minNeighbors* parameter is not intuitive. Figure 40-A,B illustrates the results with 2 different values for the *minNeighbors* parameter (3 and 5 respectively) for the detection of medaka embryos. With 3 neighbours most specimens are detected but multiple overlapping detections are returned. Increasing to 5 effectively prevents overlapping detections but results in missing several specimens (see also Figure 39.B). The overlap-based NMS presented for multi-template matching (see section 2.4.7) is generic by design and can thus be applied to any pool of bounding-boxes, provided a score for each detection. Therefore, the Fiji plugin for the detection with cascades was designed to bypass the built-in NMS of OpenCV by setting *minNeighbors* to 1 (i.e. no overlapping bounding-box needed) such that all potential object-locations are initially returned. The pool of detections is then passed to the more reliable overlap-based NMS which yields the final set of genuine detections. The result with this strategy is illustrated in Figure 40-C,D. With a minimum score of 1 and maximum overlap of 30% most specimens are correctly localized with almost no overlapping detections. Decreasing the score threshold (e.g 0.5) could potentially help detecting the undetected specimens. Ultimately, when the number of objects is known in advance, this information can be used by the NMS procedure to further avoid unexpected detections (Figure 40.D, leftmost image and Figure 39.B).

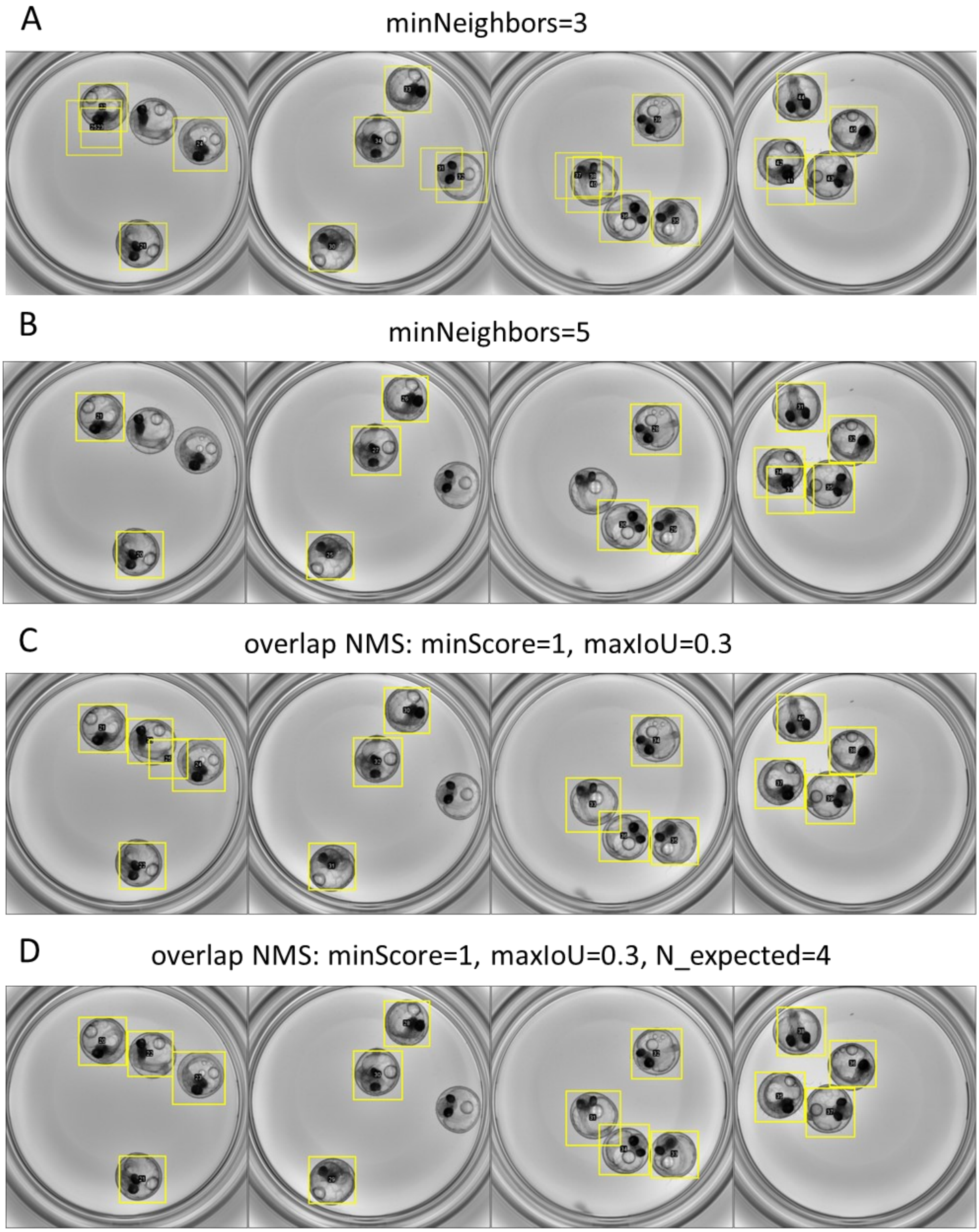

A
minNeighbors=3
B
minNeighbors=5
C
overlap NMS: minScore=1, maxIoU=0.3
D
overlap NMS: minScore=1, maxIoU=0.3, N_expected=4

Figure 40: Comparison of non-maxima suppression methods for cascade classifiers

Results with a custom medaka detector, for a set of selected images. **A,B)** Using the built-in OpenCV NMS, with no minimum score, **C)** Using overlap-based NMS and **D)** Using overlap-based NMS and indicating the expected number of objects.

3.5. Object detection with deep-learning detectors

This section describes the development of user oriented KNIME workflows using the python libraries MXNET/GluonCV, for the training of a deep learning object detector with custom images, and its subsequent application for detection. We also propose here a strategy for the annotation of ground-truth object-locations in Fiji, and present benchmarks of training time with different settings. The workflows were illustrated for the detection of head and yolk regions in zebrafish larvae.

3.5.1. Ground-truth annotations in Fiji

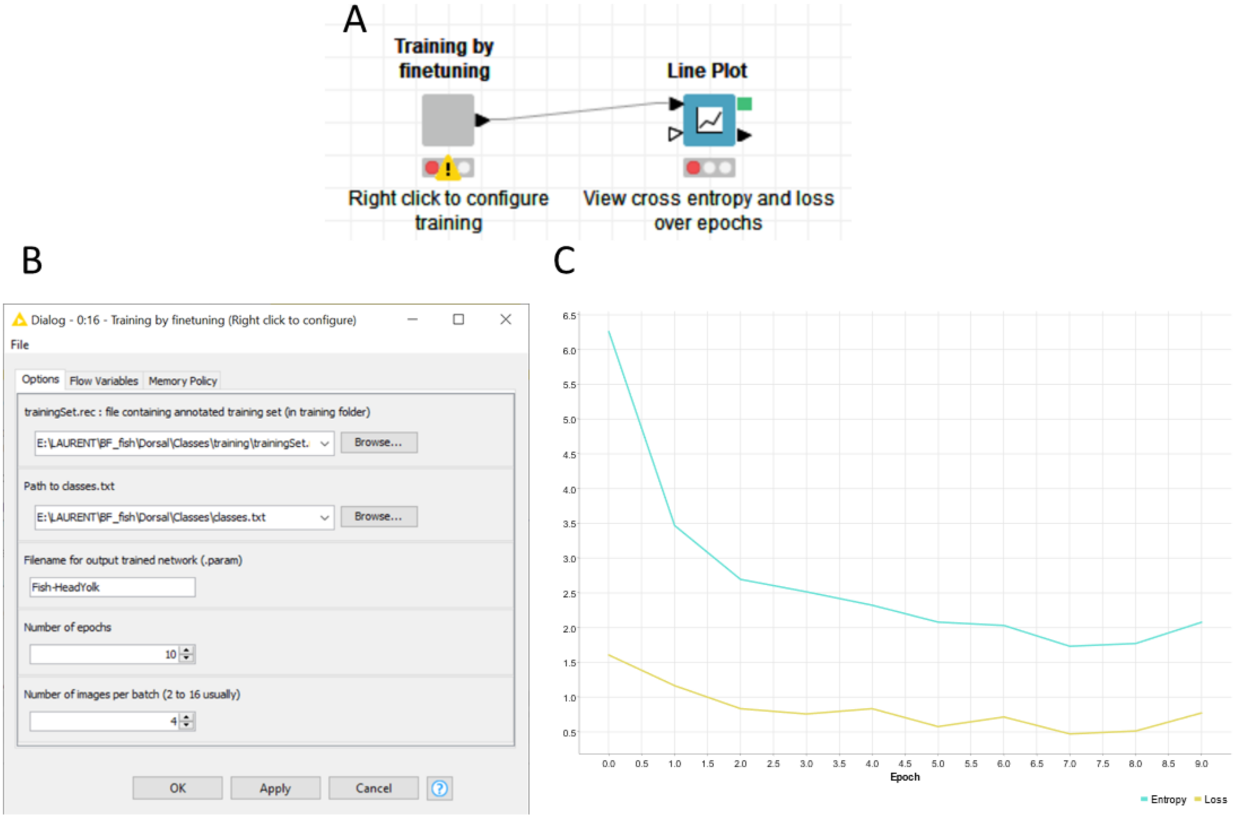
At the time of this project, no graphical interface solution existed for the annotation of object-locations and corresponding classes in images, that would propose the encoding as a .lst or .rec files as advised by the GluonCV documentation for the training of custom object detector (GluonCV Contributors 2020a) (Figure 15, Figure 16). Therefore, to facilitate ground-truth bounding-box annotations for the training of custom object-detectors with GluonCV/MXNET, a dedicated Fiji script was developed to take advantage of the annotation functionalities offered by ROIs in Fiji (see section 2.6.1). Similarly than for the cascade detector plugin (section 3.4.1) the objects should be previously outlined with rectangular ROI stored in the ROI Manager of ImageJ/Fiji. Again, this can be rapidly achieved with the ROI 1-click tools when the objects have similar sizes (see section 3.2.2). The script extracts the coordinates of object-regions outlined by rectangular ROIs and saves them to a pre-formatted csv file. This first step of annotation should be repeated for every object classes, such that one csv file contains the coordinates of all bounding-boxes for one class of object. Once all object-classes have been annotated, there should be as many csv files as object-classes. The collection of csv files is then transformed to a single .lst file. This is achieved by a first KNIME workflow, which also performs the conversion from the .lst to .rec format for faster training. Once the .lst file and/or .rec file generated, the data is ready to be used for training.

3.5.2. Custom model training in KNIME

The GluonCV documentation contains a tutorial for the training of a custom object-detector (GluonCV Contributors n.d.), by fine-tuning of a model previously trained for the detection of real-life objects, here a SSD detector (Liu et al. 2016) pretrained on the VOC dataset (Everingham et al. 2010). This tutorial consists in a collection of python source codes and associated explanations. To provide a more accessible solution than raw source code, the python script for the model training was adapted to work within a second KNIME workflow, via a python scripting node (Figure 41). The workflow provides a GUI for the configuration of the main training parameters (location of .rec file for the training set, number of epochs, batch-size, filename for the output trained model - Figure 41.A). After training, the model configuration is saved to a file (.*params* extension) for prediction on new images. The name of the object-classes is also saved to a text file, which is necessary to recover the class names from the predicted category indexes. Besides, the *line plot* node allows the visualization of the error-metrics on the training set (cross-entropy for the class predictions, and loss for the bounding-box positions - Figure 41.C). The training automatically runs on GPU when available and when the GPU version of the MXNET python package have been installed (see section 2.1.7).

3.5.3. Prediction with custom object detector

Once a custom model trained for object-detection with MXNET, it can be used to predict object-localization and classes in new images. A third KNIME workflow was thus designed to facilitate the detection and the visualization of the prediction with list of images (Figure 42.A). It is similar to the training workflow, except that the images are loaded using the image-processing functionalities of KNIME and then passed to a central processing node embedding the python script for the prediction. Additionally, the central nodes was designed to offer a custom GUI for the selection of the trained model, the selection of the text file containing the class names and the setting of the score threshold (Figure 42.B). Besides the detection, the python script performs the image pre- and post-processing before/after the detection with the model. This includes respectively downscaling the images before passing them to the neural-network and rescaling of the bounding-boxes to the original image size after detection. The detected regions are rendered as a mask image in KNIME so they can be visualized as an overlay on the original images (*Interactive Segmentation View* node and Figure 42.C), and further processed using the built-in image-processing functionalities of KNIME. The coordinates and dimensions of the bounding-boxes as well the associated scores for each image are also returned as a result table (*Image Viewer* “Bboxes table” node and Figure 42.D).



Dialog - 0:16 - Training by finetuning (Right click to configure)

File

trainingSet.rec : file containing annotated training set (in training folder)

E:\LAURENT\BF_fish\Dorsal\Classes\training\trainingSet.

Path to classes.txt

E:\LAURENT\BF_fish\Dorsal\Classes\classes.txt

Filename for output trained network (.param)

Fish-HeadYolk

Number of epochs

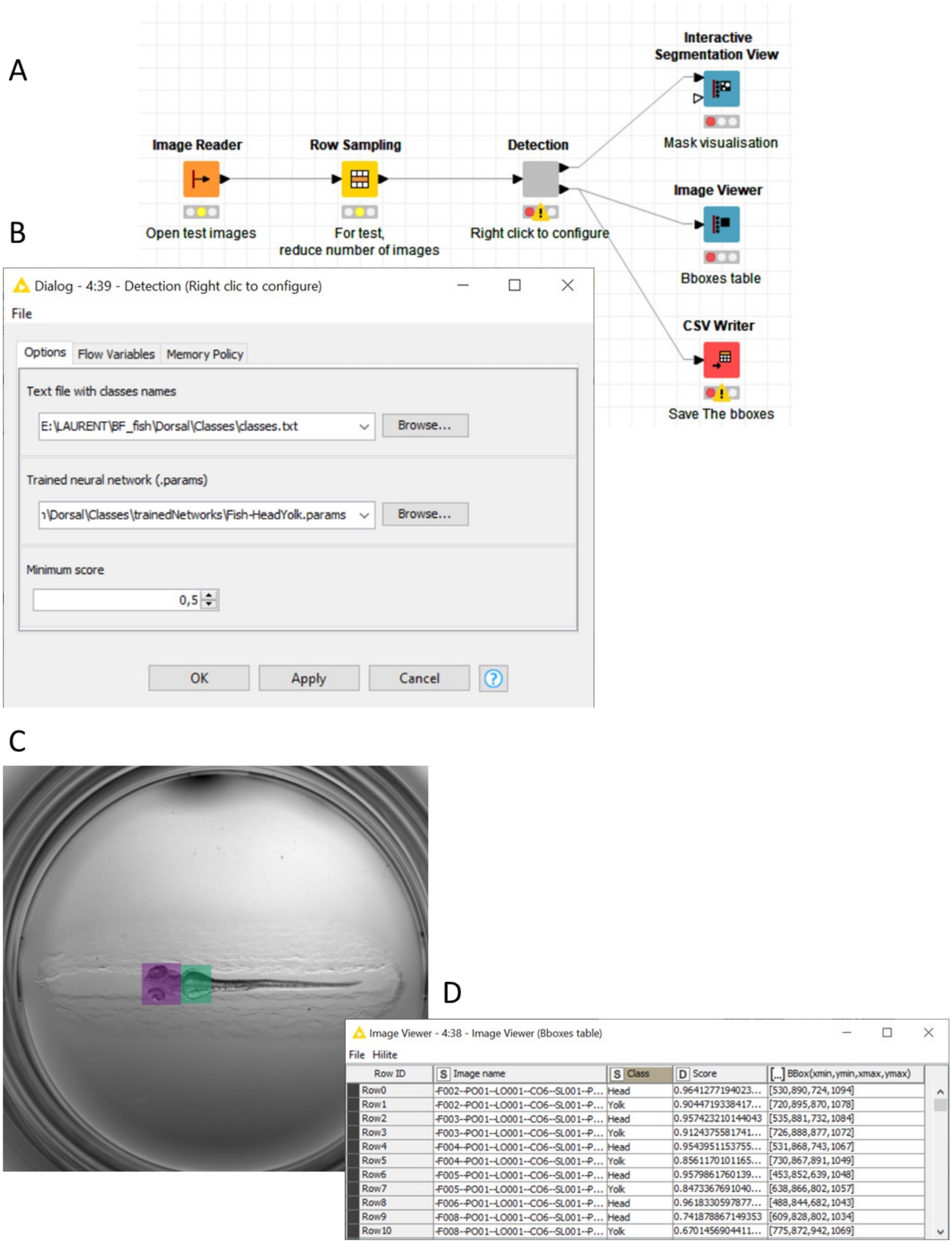
10

Number of images per batch (2 to 16 usually)

4

Figure 41: KNIME workflow for the training of a custom deep learning object-detector

A) KNIME workflow for the training of a deep learning object-detector using the MXNET library in python. The component node (in grey) contains the central python node with the script for the model training, and a few nodes to generate the configuration window (see B). **B)** Custom configuration window for the setting of the main training parameters (location of the training data, number of epochs, batch size). It is shown by right-clicking on the component node. **C)** Visualization from the line plot node showing the error metrics (cross-entropy for the class predictions, and loss for the bounding-box positions) on the training set for the successive training iterations (epochs). The error-metrics have a typical exponential decay profile for a training that converges on the training set. However, without a validation set, one cannot predict if the model will generalize well on new data. Own contribution. Licensed under CC-BY.



Dialog - 4:39 - Detection (Right clic to configure)

File

Options | Flow Variables | Memory Policy

Text file with classes names: E:\LAURENT\BF_fish\Dorsal\Classes\classes.txt [Browse...]

Trained neural network (.params): 1\Dorsal\Classes\trainedNetworks\Fish-HeadYolk.params [Browse...]

Minimum score: 0,5

OK | Apply | Cancel

Image Viewer - 4:38 - Image Viewer (Bboxes table)

File Hilite

Row ID	S Image name	S Class	D Score	[...] BBox(xmin,ymin,xmax,ymax)
Row0	-F002--P001--L0001--C06--SL001--P...	Head	0.9641277194023...	[530,890,724,1094]
Row1	-F002--P001--L0001--C06--SL001--P...	Yolk	0.9044719338417...	[720,895,870,1078]
Row2	-F003--P001--L0001--C06--SL001--P...	Head	0.957423210144043	[535,881,732,1084]
Row3	-F003--P001--L0001--C06--SL001--P...	Yolk	0.9124375581741...	[726,888,877,1072]
Row4	-F004--P001--L0001--C06--SL001--P...	Head	0.9543951153755...	[531,868,743,1067]
Row5	-F004--P001--L0001--C06--SL001--P...	Yolk	0.8561170101165...	[730,867,891,1049]
Row6	-F005--P001--L0001--C06--SL001--P...	Head	0.9579861760139...	[453,852,639,1048]
Row7	-F005--P001--L0001--C06--SL001--P...	Yolk	0.8473367691040...	[638,866,802,1057]
Row8	-F006--P001--L0001--C06--SL001--P...	Head	0.9618330597877...	[488,844,682,1043]
Row9	-F008--P001--L0001--C06--SL001--P...	Head	0.741878867149353	[609,828,802,1034]
Row10	-F008--P001--L0001--C06--SL001--P...	Yolk	0.6701456904411...	[775,872,942,1069]

Figure 42: KNIME workflow for object-detection with a custom deep-learning model

View of the KNIME workflow. The test images are loaded with the built-in image-processing functionality of KNIME. The detection is done via a python scripting node, embedded in the *Detection* component node (in grey). The component node also contains nodes to generate the graphical interface (**B**) used to set up the detection parameters (location of the classes.txt, trained model and score threshold). **C)** Detected head and yolk regions viewed as an overlay on the original image in KNIME. **D)** Result table with bounding-boxes coordinates, dimensions and classes for each detection. Own contribution. Licensed under CC-BY.

3.5.4. Head and yolk detection in zebrafish larvae

Using the training workflow, a model was trained for the detection of both the head and yolk regions in zebrafish larvae oriented dorsally in wells of a 96-well plate (Figure 42). The training was carried out for 10 epochs with 60 training images. The duration of the training was compared between GPU/CPU and input format of the training set (lst vs rec file) on the workstation 1 (Figure 43). The training was most efficient when running on GPU and using a .rec file for the training set. Using a .lst file, the training on GPU did not converge due to memory overflowing after 1 epoch. The error metrics (loss and entropy) can be visualized for the course of the training in Figure 41.C. Both metrics decrease and stabilize which indicates that the model manages to predict both the object-locations and classes correctly for most training images. The model performance was then evaluated on a separate test set of 58 images, with a minimum score of 0.5 for individual detections. The model achieves good performances even on more challenging images (out of focus, disrupted specimen, bubbles, etc.). Both the head and yolk regions are correctly detected for almost all the 58 test images. Except for 5 images, for which only 1 of the 2 regions was correctly found. In another image, an additional false positive detection was returned in addition to the head and yolk. The python node takes 39 seconds to process the 58 images (0.7 sec/image), when the detection is run on the GPU.

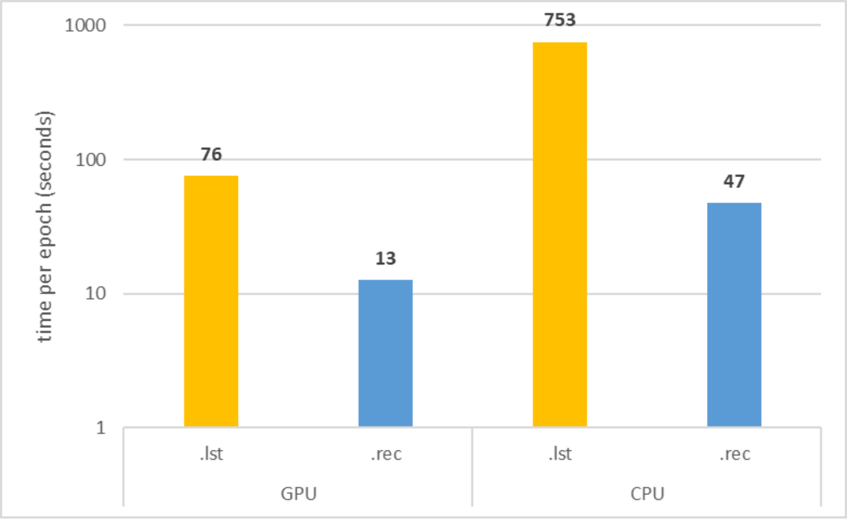


Figure 43: Computation time per epoch for different training configuration

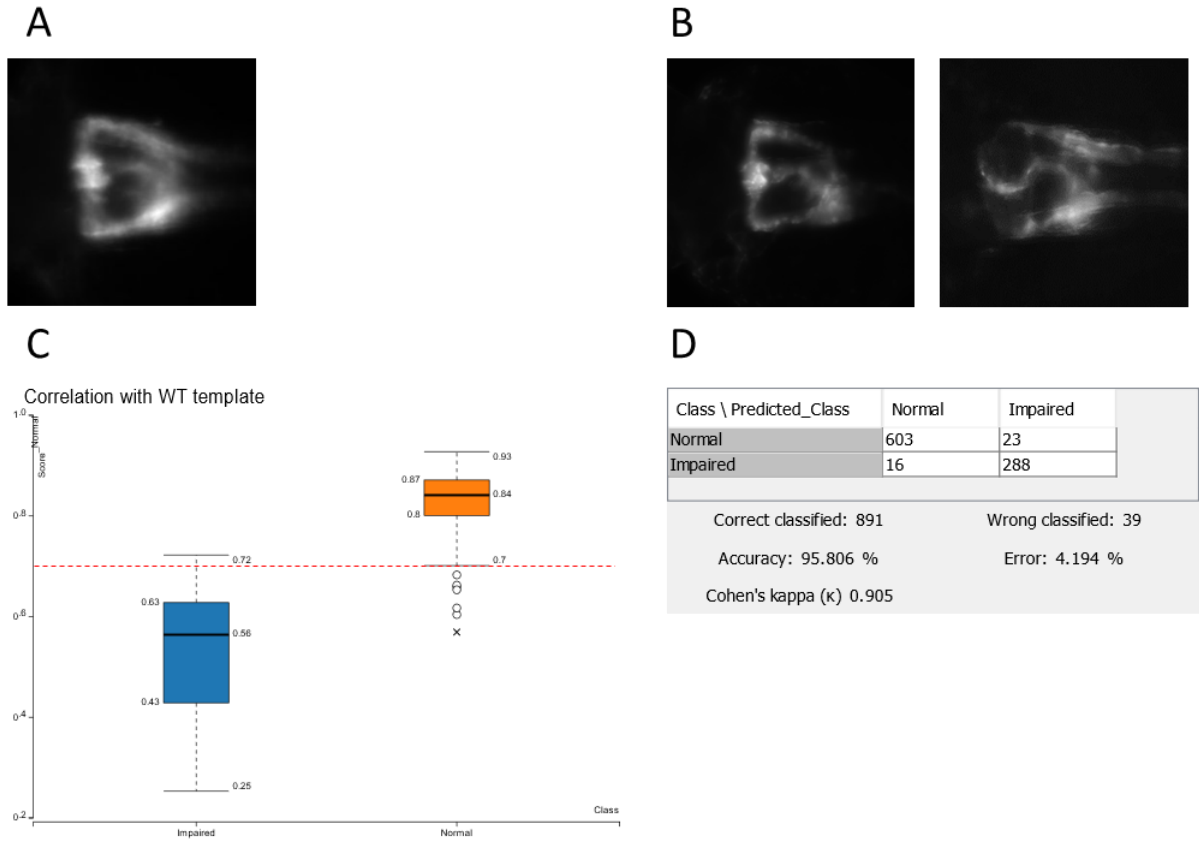
The computation time per training iteration (epoch) was compared when running on GPU or CPU, and when loading the training set from a .rec file or from the information contained in a .lst file. The training was run with 60 training images and a batch size of 2 on the workstation 1. Time axis with logarithmic scale.

3.6. Image-classification

3.6.1. Classifying zebrafish kidney morphologies with template matching

Single and multi-template matching can be used for image-classification too. When the template and target images have identical dimensions, then the result of the template matching is a single score value. Provided a single template, a threshold on this score can be used to perform binary classification (Figure 7). Images thus get classified as corresponding to the category of the template or not, depending on their correlation-score or difference-score with the reference template. Figure 44 thus illustrates the classification of fluorescently-labelled kidney morphologies (Figure 44.B) in transgenic *Tg(wt1b:EGFP)* zebrafishes as observed in a previously published toxicity screening study (Pandey et al. 2019) (see dataset section 2.2.3), using a single template representative of the wild type morphology (Figure 44.A). Images from the dataset were previously annotated as belonging to the wild-type or cystic category (1240 annotated kidney images: 840 normal, 400 cystic). A fraction of the annotated dataset (25% - 310 images: 214 normal, 96 cystic) was used to derive an optimal score threshold for the classification, similarly than with a training fraction for machine learning. This was done by visualizing the distribution of the template matching scores with the reference wild-type template for the 2 classes (Figure 44.C). From this distribution, a correlation-score of 0.7 with the reference template was chosen as threshold to classify the rest of the dataset (test fraction). Using this threshold, the classification reaches an accuracy of 95% with 930 test images (Figure 44.D). Only 4% of normal specimens were classified as impaired and 5% of impaired were classified as normal. This template-based classification pipeline was implemented as a KNIME workflow relying on the multi-template-matching python package (see method section 2.8.1).

The classification can also be completed with multiple templates each representing a different object-category. In this case, the image is classified as the category of the template returning the highest score. A score threshold can also be applied, such that if none of the templates return a score above the threshold, the image is classified as outlier. Multi-class classification with template matching is illustrated for the recognition of more than 2 zebrafish kidney morphologies in Figure 45 (see method section 2.8.1). However, the classification accuracy is lower than binary classification, due to similar intensity distributions between the reference templates, which are not easily distinguished by template matching.



Class \ Predicted_Class	Normal	Impaired
Normal	603	23
Impaired	16	288

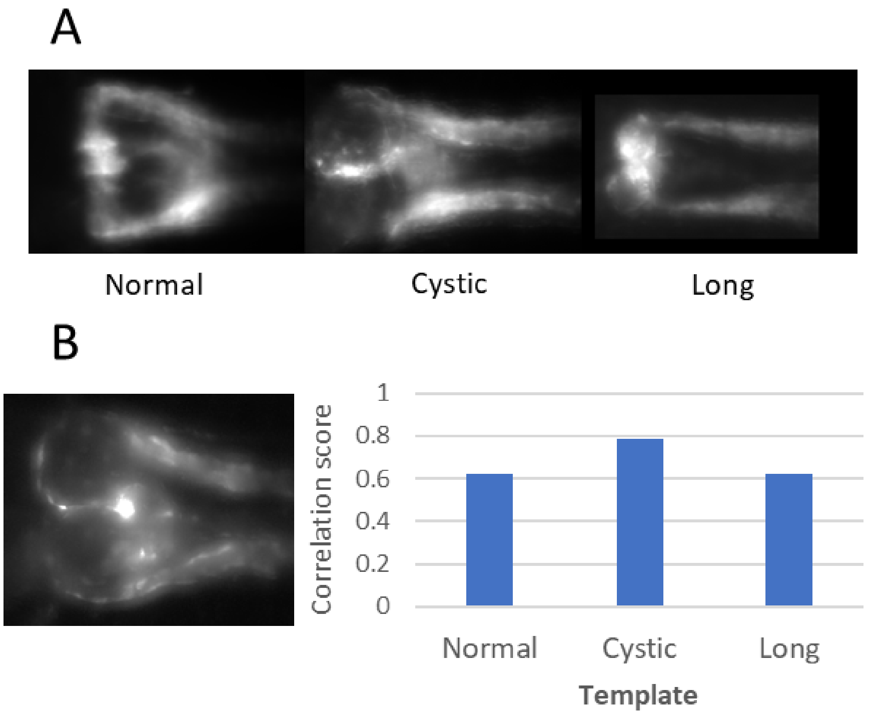
Correct classified: 891 Wrong classified: 39

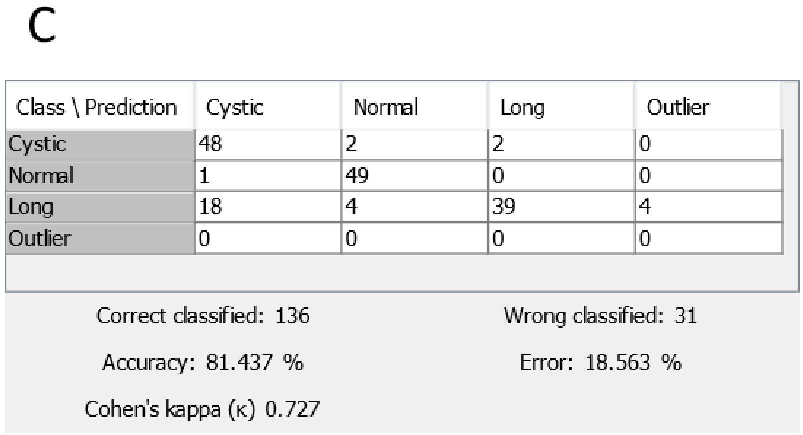
Accuracy: 95.806 % Error: 4.194 %

Cohen's kappa (κ) 0.905

Figure 44: Binary image-classification using single template matching

Single-template matching used for the binary classification of kidney morphology (normal vs impaired phenotype) in the *Tg(wt1b:EGFP)* transgenic zebrafish line, expressing the green fluorescent protein (GFP) in the pronephroi (1240 annotated kidney images: 840 normal, 400 impaired. Images are from a previously published toxicity study (Westhoff et al. 2020). **(A)** Reference template image representing the normal kidney morphology. The image is an average projection of 15 images after registration using centre of mass, to compensate for variability in the fluorescent signal. **(B)** Example of images representing the 2 morphology classes for the classification: normal on the left, and impaired on the right. **(C)** Distribution of the correlation-score with the reference template for the 2 classes, with 25% of the manually annotated dataset (310 images: 214 normal, 96 impaired). The red dashed line indicates the value of 0.7, used as a threshold for the classification of new images. **(D)** Confusion matrix depicting the accuracy of the classification for the remaining 75% of the annotated dataset (930 images: 626 normal, 304 impaired) using the 0.7 score-threshold. The accuracy of the classification is quite high (95%), with such a simple classifier. Only about 4% of normal specimen were classified as impaired and 5% of impaired were classified as normal. Own contribution, reproduction allowed according to CC-BY licence.





Class \ Prediction	Cystic	Normal	Long	Outlier
Cystic	48	2	2	0
Normal	1	49	0	0
Long	18	4	39	4
Outlier	0	0	0	0

Correct classified: 136 Wrong classified: 31

Accuracy: 81.437 % Error: 18.563 %

Cohen's kappa (κ) 0.727

Figure 45: Template matching for multiple class classification

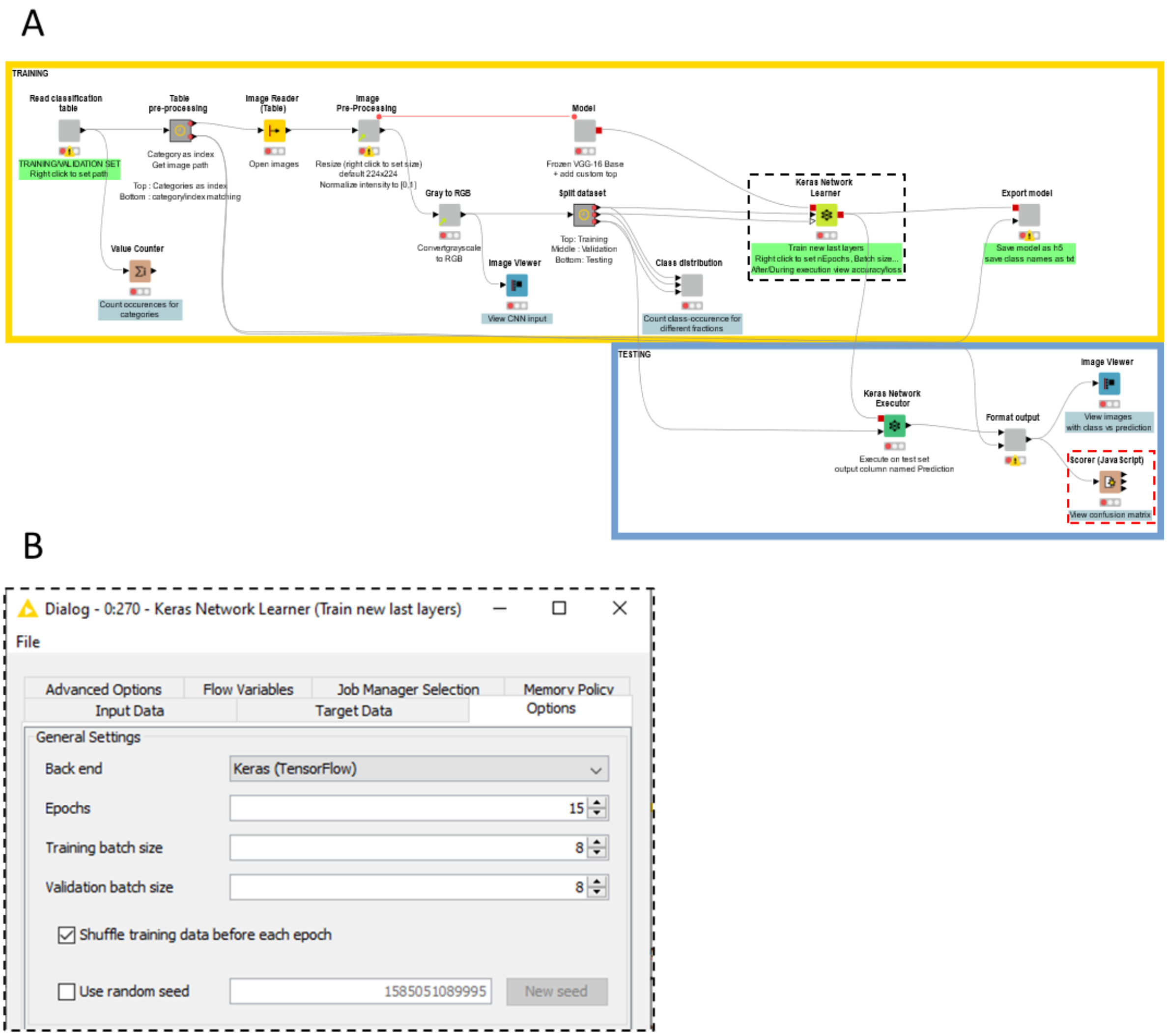
A) Template images representing 3 zebrafish kidney morphologies as observed in a previously published toxicity screening (Westhoff et al. 2020). **B)** Example of test image and score for each template. The image is assigned the category of the template with the highest score, here the Cystic category (correct). The difference of score between the classes is relatively low though, as the templates present similar pixel intensity distributions. **C)** Result of the classification for a test set of 167 annotated images (50 normal, 52 cystic, 65 long). Images for which none of the template score is above 0.5 were classified as outliers. The cystic and normal kidney are well recognized, however the long phenotype is often mis-predicted as cystic.

3.6.2. KNIME workflows for the training of deep learning image-classifiers

The previously described Fiji annotation plugins (see section 3.2.3) are particularly adapted to generate the ground-truth category annotations for the training of image-classifiers. Therefore, to illustrate an example of such application and to simplify the access of deep-learning for image-classification to non-experts, a set of KNIME workflows were developed (Figure 48, Figure 46.A), which are available and documented on the GitHub repository (Thomas 2020j). The workflows are adapted from an existing KNIME example workflow (Dietz 2020) and from guidelines from an online tutorial (Rosebrock 2014a). They rely on the Keras library (Chollet and others 2015), which is fully integrated in KNIME, and thus do not require coding. The workflows were designed to directly take as input annotation tables as generated by the *single class (button)* plugin of the *Qualitative annotation* plugins presented in section 3.2.3. The workflows are configured with sensible default parameters and thus can be executed without major adaptation. Advanced users can still tune the parameters for the model training via the configuration window of the *Keras network learner* node (Figure 46.B). 2 workflows are provided for the training of custom deep-learning models for image-classification with custom images and classes. Both workflows generate a hybrid model made of a pre-trained VGG16 base (Simonyan and Zisserman 2015) for feature extraction, completed by newly appended fully-connected layers for the classification with custom classes (see section 1.8.1: “Deep-learning models”). Only the newly initialized layers are trained, while the VGG16 base pre-trained on the ImageNet dataset (Deng et al. 2009) stays untouched (frozen). By doing so, the various features previously learned by the VGG16 base to distinguish the various real-life objects from the original training set are reused to distinguish novel object-classes. This strategy known as transfer-learning effectively reduces the need for large training set and the time to train a model (Zhu et al. 2011). The 2 training workflows differ slightly in the structure of the classification layers (see method section 2.8.3). One workflow is designed for binary classification exclusively, meaning that images are assigned to 1 category out of 2 possible categories. Therefore, the annotated training set can only contain 2 categories (e.g. positive vs negative). The classifier in this model returns a single probability as in Figure 6. The second workflow can be used for classification with 2 or more categories and returns for each image the probabilities for each category. Images are thus assigned to a single top-score category.

Provided the annotation table, the workflows load the images and associated category labels in memory, pre-process the images (downscaling and intensity normalization) and finally train the model. The annotated dataset is automatically partitioned into training, validation and test set. For the training and validation set, the error metrics for the model (accuracy or loss) can be monitored in real time along the training (Figure 47.B). For the test set, the accuracy of the classification is automatically computed and can be visualized as a confusion matrix (Figure 47.C).

Finally, the trained model can be exported to disk for prediction on new images. Dedicated workflows for the prediction are also provided.



Dialog - 0:270 - Keras Network Learner (Train new last layers)

General Settings

Back end: Keras (TensorFlow)

Epochs: 15

Training batch size: 8

Validation batch size: 8

☑ Shuffle training data before each epoch

☐ Use random seed 1585051089995

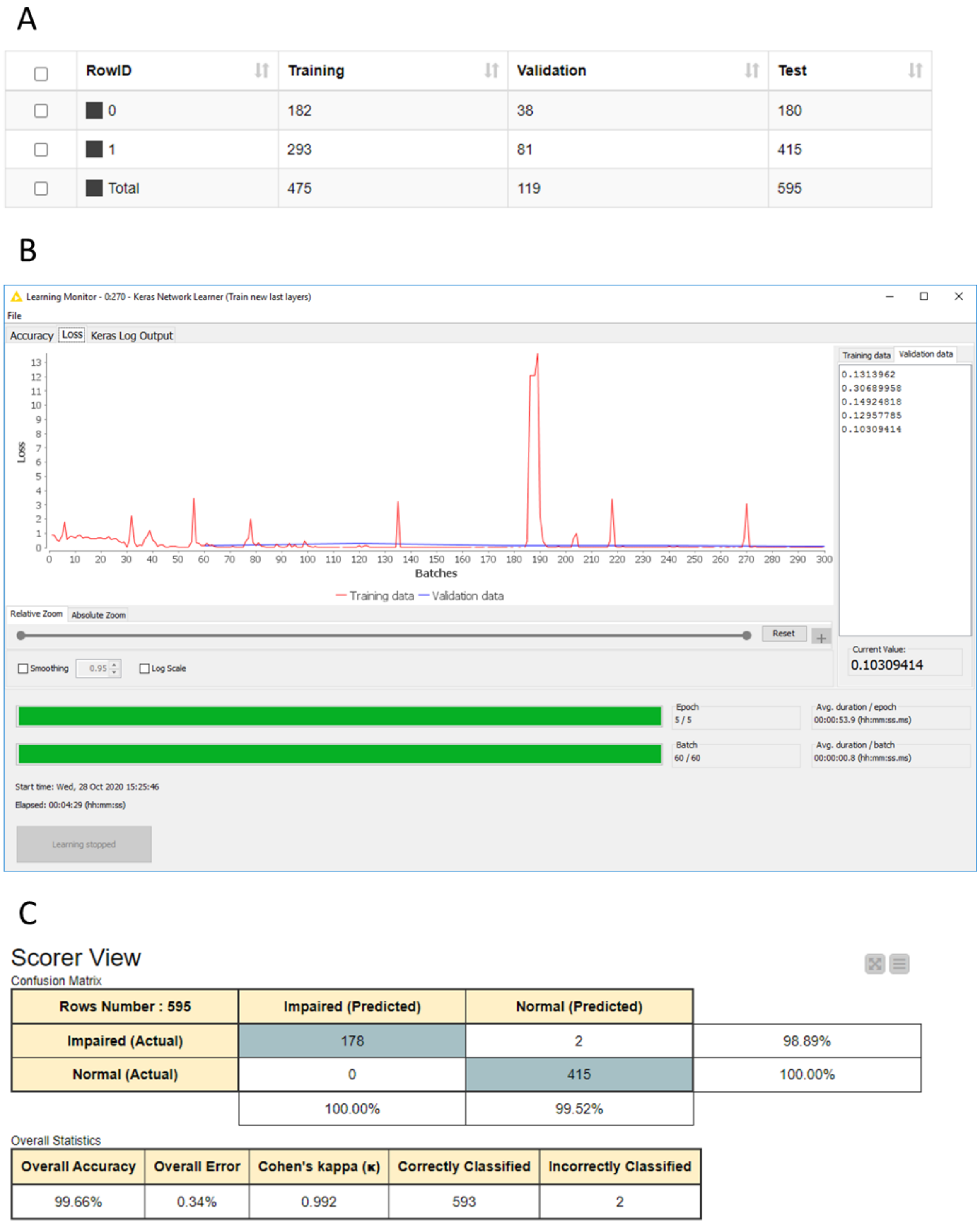
Figure 46: KNIME workflow for the training of a deep learning model for image classification

A) Screenshot of the KNIME workflow for the training of the model. A subset of the annotated data (training and validation set) is used for the model training (upper part of the workflow outlined in yellow). The remainder of the annotated data (test set) is used for the evaluation of the model performance after training (lower part of the workflow, outlined in blue). **B)** Configuration window of the *Keras network learner node* for the setting of the training parameters.

3.6.3. Classifying zebrafish kidney morphologies with deep learning

The KNIME workflow presented in the previous section was used to train a model for the identification of two distinct morphological phenotypes of fluorescently-labelled pronephroi, as previously with template matching (normal vs impaired/cystic, Figure 44). With that same dataset of about 1000 images, a first model was trained with about half of the images (see partitioning of the data in Figure 47.A). The model was trained for 5 epochs with a batch size of 8 images, which took about 4 min 30 secs on a HIVE server with a 2 GB GPU (see section 2.1.2: "Hardware configuration"). The loss for the validation set was already low after 1 epoch (0.13), suggesting that the models readily classify correctly unseen images after a single training iteration. The loss is further reduced after 5 epochs, depicting further improvement of the classification (0.10, see Figure 47.B). On a separate test set of about 600 images, the model correctly classified almost all images. Only 2 images representing the impaired phenotypes (cystic) were misclassified as normal (see confusion matrix in Figure 47.C). A similar training was carried out with fewer images of the same object but imaged at higher resolution (116 training images: 57 normal, 59 cystic, Figure 48). The training also converged but started overfitting the training set, as it can be seen by the increasing error on the validation set after the batch 45 (Figure 48.B), potentially due to the smaller training set. Still, the trained model achieved high classification accuracy (>99%), with a single misclassified image on a test set of 145 images (69 normal, 76 cystic) after training for only 5 epochs (Figure 48.C).

A last model was trained with the second KNIME training workflow for the identification of more than 2 kidney morphologies, similarly than with template matching in Figure 45. Only 20 training images per class were used (see dataset partitioning Figure 49.A). The model was thus trained for a few more iterations to achieve satisfying performance for both the training and validation fractions (15 epochs, Figure 49.B). On a separate test set of 84 images, the resulting model performs quite well again, with more than 90% overall accuracy as well as satisfying class-specific metrics (Figure 49.C).



A

□	RowID	Training	Validation	Test
□	■ 0	182	38	180
□	■ 1	293	81	415
□	■ Total	475	119	595

B

Learning Monitor - 0:270 - Keras Network Learner (Train new last layers)

```
0.1313962
0.30689958
0.14924818
0.12957785
0.10309414
```

Current Value: 0.10309414

□ Smoothing 0.95 □ Log Scale

Epoch 5 / 5

Avg. duration / epoch 00:00:53.9 (hh:mm:ss.ms)

Batch 60 / 60

Avg. duration / batch 00:00:00.8 (hh:mm:ss.ms)

Start time: Wed, 28 Oct 2020 15:25:46

Elapsed: 00:04:29 (hh:mm:ss)

C

Scorer View

Confusion Matrix

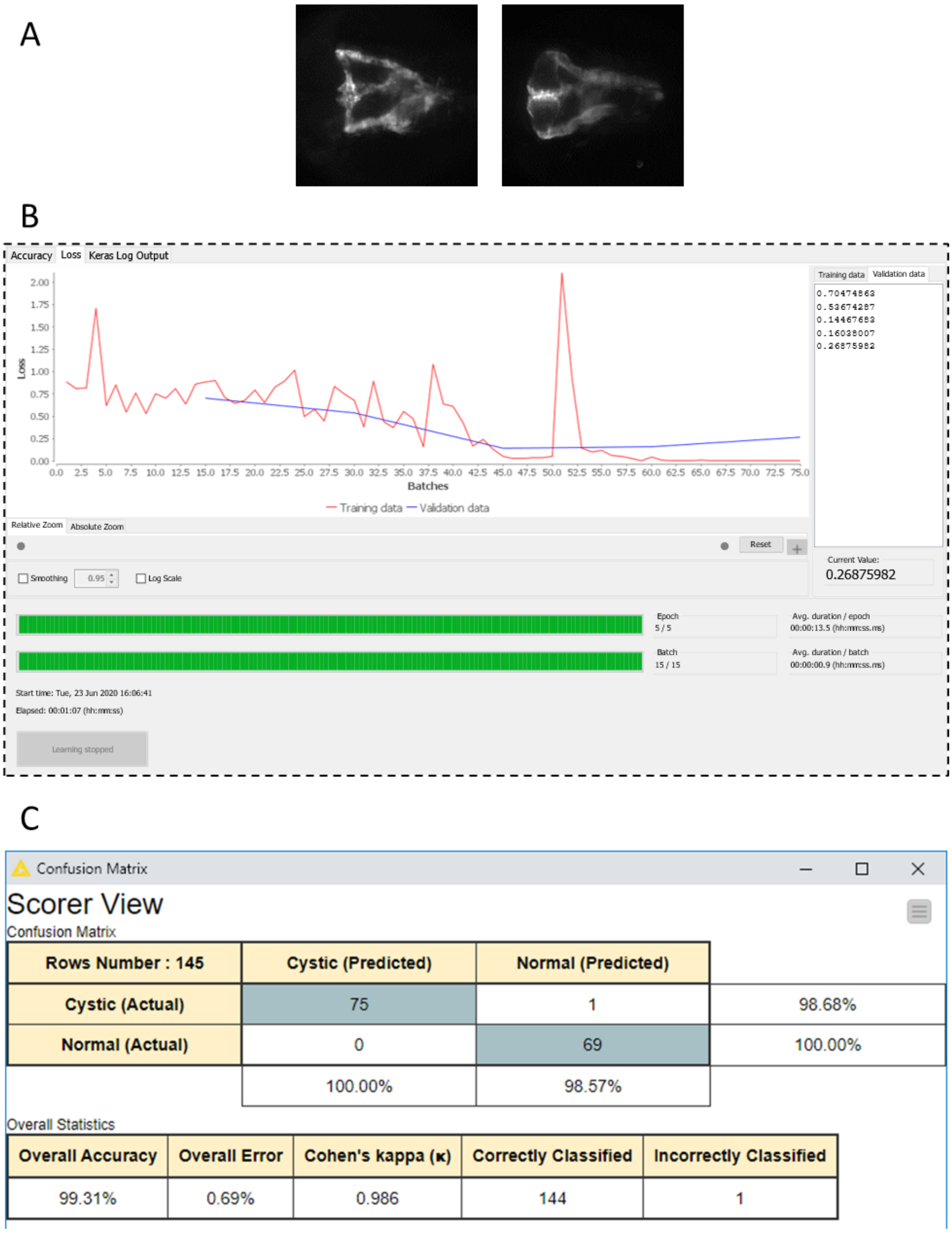
Rows Number : 595	Impaired (Predicted)	Normal (Predicted)	
Impaired (Actual)	178	2	98.89%
Normal (Actual)	0	415	100.00%
	100.00%	99.52%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
99.66%	0.34%	0.992	593	2

Figure 47: Binary classification of zebrafish kidney morphologies using deep learning (low resolution)

A) Distribution of the annotated images between training, validation and test fraction. The row ID corresponds to the category index (here 0=Impaired, 1=Normal. **B)** View of the *Learning monitor* panel of the Keras network learner node, showing the value of the loss for the training and validation batches along the training (red and blue respectively). This plot is used to estimate the progress of the model in real-time. **C)** Confusion matrix depicting the accuracy of the model on the test fraction. See Figure 48.A for example images of the kidney morphologies classified.



```
0.70474863
0.53674287
0.14467683
0.16038007
0.26875982
```

Current Value:
0.26875982

□ Smoothing 0.95 □ Log Scale

Epoch
5 / 5

Avg. duration / epoch
00:00:13.5 (hh:mm:ss.ms)

Batch
15 / 15

Avg. duration / batch
00:00:00.9 (hh:mm:ss.ms)

Start time: Tue, 23 Jun 2020 16:06:41

Elapsed: 00:01:07 (hh:mm:ss)

Confusion Matrix

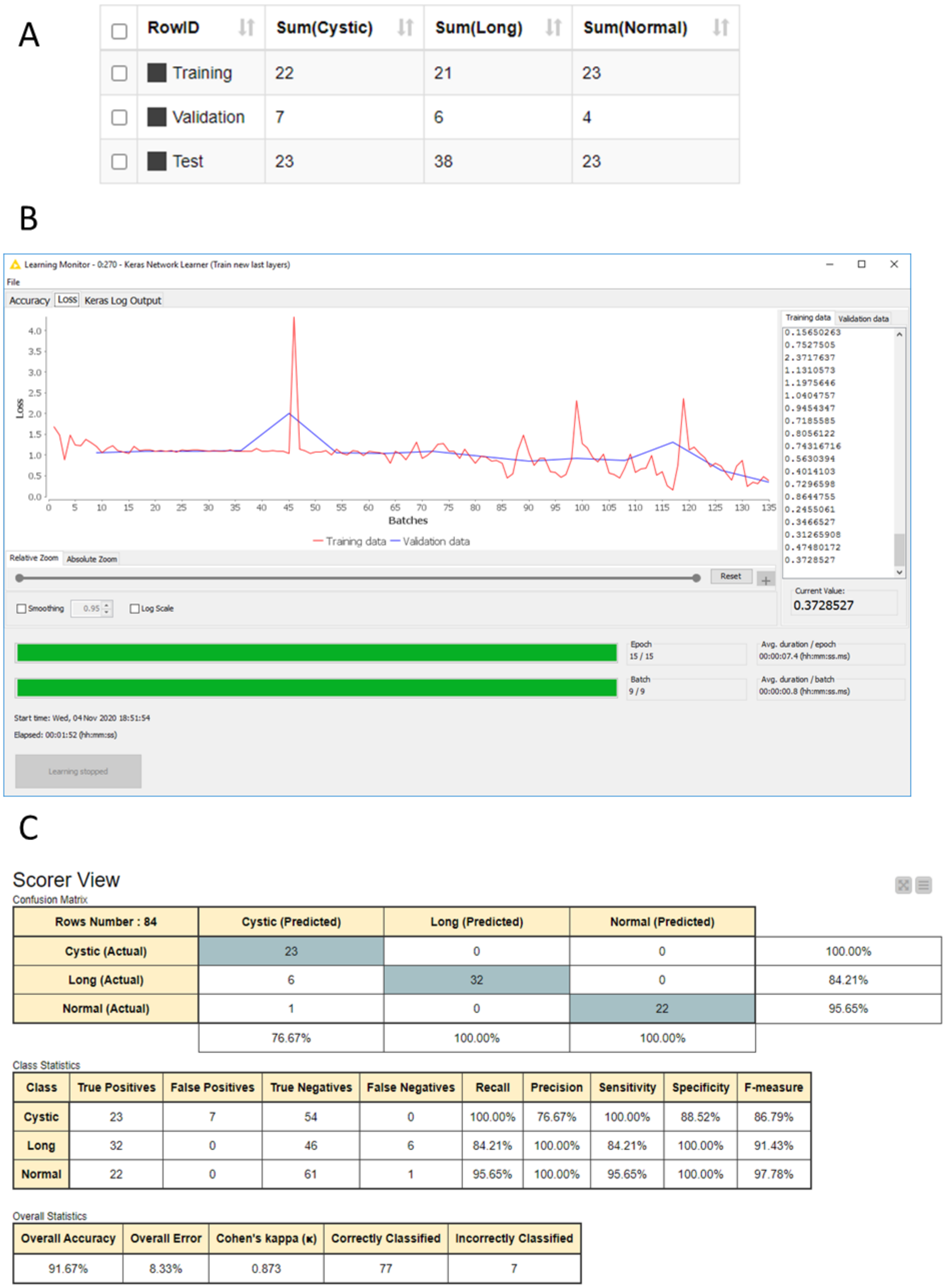
Rows Number : 145	Cystic (Predicted)	Normal (Predicted)	
Cystic (Actual)	75	1	98.68%
Normal (Actual)	0	69	100.00%
	100.00%	98.57%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
99.31%	0.69%	0.986	144	1

Figure 48: Binary classification of zebrafish kidney morphologies using deep learning (high resolution)

A) Example images representing the two classes the deep-learning model was trained to recognize. The images show fluorescently-labelled pronephroi in larvae of the *Tg(wt1b:EGFP)* transgenic zebrafish line showing 2 common morphologies (Left: Normal, Right: Cystic), observed in a previously published chemical screen (Pandey et al. 2019). **B)** View of the *Learning monitor* panel of the *Keras network learner* node. The training and validation losses decrease until about batch 45. From there the validation loss increases, suggesting the overfitting of the model on the training set. **C)** Confusion matrix as shown by the Scorer node depicting the accuracy of the classification by the trained model on a separate test set. After training for only 5 epochs with 116 training images the model achieved already a very good classification accuracy with only 1 image misclassified, out of 145 test images.



☐	RowID	Sum(Cystic)	Sum(Long)	Sum(Normal)
☐	■ Training	22	21	23
☐	■ Validation	7	6	4
☐	■ Test	23	38	23

```
0.15650263
0.7527505
2.3717637
1.1310573
1.1975646
1.0404757
0.9454347
0.7185585
0.8056122
0.74316716
0.5630394
0.4014103
0.7296598
0.8644755
0.2455061
0.3466527
0.31265908
0.47480172
0.3728527
```

Current Value:
0.3728527

☐ Smoothing 0.95 ☐ Log Scale

Epoch
15 / 15

Avg. duration / epoch
00:00:07.4 (hh:mm:ss.ms)

Batch
9 / 9

Avg. duration / batch
00:00:00.8 (hh:mm:ss.ms)

Start time: Wed, 04 Nov 2020 18:51:54

Elapsed: 00:01:52 (hh:mm:ss)

Scorer View

Confusion Matrix

Rows Number : 84	Cystic (Predicted)	Long (Predicted)	Normal (Predicted)	
Cystic (Actual)	23	0	0	100.00%
Long (Actual)	6	32	0	84.21%
Normal (Actual)	1	0	22	95.65%
	76.67%	100.00%	100.00%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
Cystic	23	7	54	0	100.00%	76.67%	100.00%	88.52%	86.79%
Long	32	0	46	6	84.21%	100.00%	84.21%	100.00%	91.43%
Normal	22	0	61	1	95.65%	100.00%	95.65%	100.00%	97.78%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
91.67%	8.33%	0.873	77	7

Figure 49: Classification of multiple zebrafish kidney morphologies using deep learning

A) Distribution of the annotated images between training, validation and test fraction. **B)** View of the model loss for the train and validation fraction along the training (15 epochs). The loss is rather stable for the first half of the training but finally decreases. **C)** Confusion matrix depicting the accuracy of the model on the test fraction. See Figure 45.A for example images of the kidney morphologies classified.

3.7. Automated microscopy

3.7.1. Feedback microscopy for targeted high-resolution imaging of regions of interest

Feedback microscopy was demonstrated with single template matching on the ACQUIFER IM04, for the targeted high-resolution imaging of regions of interest, from samples mounted in 96-well plates. A folder-watching routine was implemented in Fiji using the jython scripting language. This routine should be started together with an overview acquisition protocol on the microscope. The overview acquisition protocol was configured to image each well of the plate using the low magnification objective (2X), such that the field of view encloses the full well. The folder-watching routine is scanning at regular time interval (ex: every 2 seconds) the directory in which the low magnification images are saved, and reports for every iteration the list of newly created images. For each of these images, a custom single-template-matching code is executed within Fiji with a predefined template. As part of the detection process, the centre of the localized image region is converted from pixel coordinates to the corresponding XY objective coordinates, taking into account the pixel size for this magnification, and the original XY position of the objective for the low-resolution image (typically corresponding to the centre of the well). This information is automatically extracted from the image filename, which contains the metadata for the acquisition. Finally, a text file (.tmp) is created on disk with pre-defined machine commands to execute, targeted to the localized sample region. The text file is named according to the image currently processed, to avoid overwriting any existing file generated for a previous image. The machine commands include the positioning of the objective to the calculated coordinates, the change from the low-resolution to the high-resolution objective, the autofocus commands and finally the commands related to the acquisition (illumination settings, number of Z-slices, etc., see section 2.7.1 for details about the implementation).

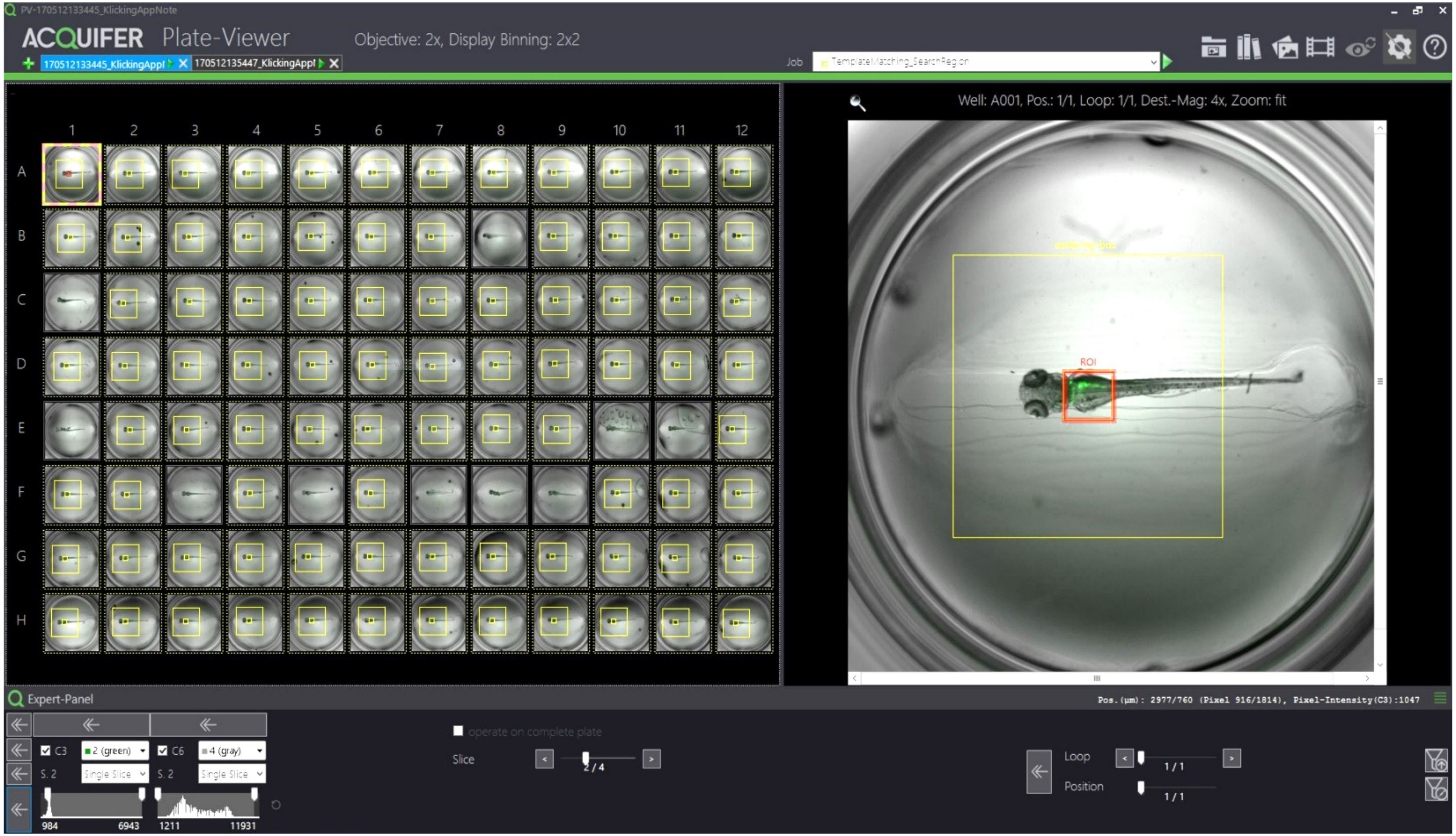
This implementation of single template matching for feedback microscopy works with any imaging channel including fluorescent ones, provided the object can be reliably localized. It was tested for the automated high-resolution imaging of the head region in zebrafish larvae from low resolution images as in Figure 30. To detect other tissues or organs, the template image should be updated accordingly, and the commands to execute for the localized regions should also be adapted to the new sample properties. The custom single-template-matching commands and the targeted microscope commands reported here are hard coded as part of the folder watching routine. Additional efforts would be necessary to adapt the pipeline to any generic ROI-detection algorithm, and to customizable microscope commands.

3.7.2. Semi-automated acquisition using PlateViewer plugins

In addition to the visualization of the data as an intuitive plate layout, the ACQUIFER PlateViewer (ACQUIFER 2017b) allows the configuration of semi-automated acquisition protocols. ROIs can be interactively defined for targeted acquisition, either by clicking on image-regions, or via external ROI-detection algorithms (e.g. Fiji macro) configured as PlateViewer plugins. The positions of the outlined regions may be used to automatically update the objective coordinates from an existing acquisition protocol, which is then executed by the acquisition software of the microscope. This simple in-place replacement of objective-coordinates facilitates the semi-automated targeted acquisition of specific ROIs, for multiple samples.

Using the plugin functionality of the PlateViewer, single template matching was implemented using Fiji as an external program (see section 2.7.2, 2.7.3). From the PlateViewer interface, the template used for the search is defined by the user by outlining a rectangular image-region in one of the plate images. Additionally, the user has the possibility to set a rectangular search-region to limit the template matching search to a fraction of the images. This search region is propagated to all images. Finally, the user should select the images for which the plugin will be executed. Upon execution of the PlateViewer plugin, Fiji is called in headless mode with a custom single template matching script for the localization of a single object, using a 0-mean normalized cross-correlation score by default. Once the search terminated, the results can be viewed in the PlateViewer as illustrated in Figure 50, together with a preview of the field of view for an objective of higher resolution centred on the detected region. Finally, the outlined ROIs can be manually refined if necessary, before updating the objective positions from an existing acquisition protocol, to perform a semi-automated targeted acquisition. The result of such acquisition is shown in Figure 51.

The single-template matching plugin is now provided by default with the PlateViewer as an example plugin and routinely used by collaborators and customers. Besides the imaging of previously demonstrated zebrafish head and yolk, it was used for the targeted imaging of the heart in laterally oriented zebrafish larvae for the screening of compound affecting epicardium development (Lupi et al. 2017) or for the localization of the tail region with application for the monitoring of the immune response over time after tail-fin amputation (unpublished).



PV-170512133445_KlickingAppNote

Objective: 2x, Display Binning: 2x2

Job: TemplateMatching_SearchRegion

Well: A001, Pos.: 1/1, Loop: 1/1, Dest.-Mag: 4x, Zoom: fit

Pos.(µm): 2977/760 (Pixel 916/1814), Pixel-Intensity(C3):1047

☑ C3 ■ 2 (green) S. 2 Single Slice

☑ C6 ■ 4 (gray) S. 2 Single Slice

□ operate on complete plate

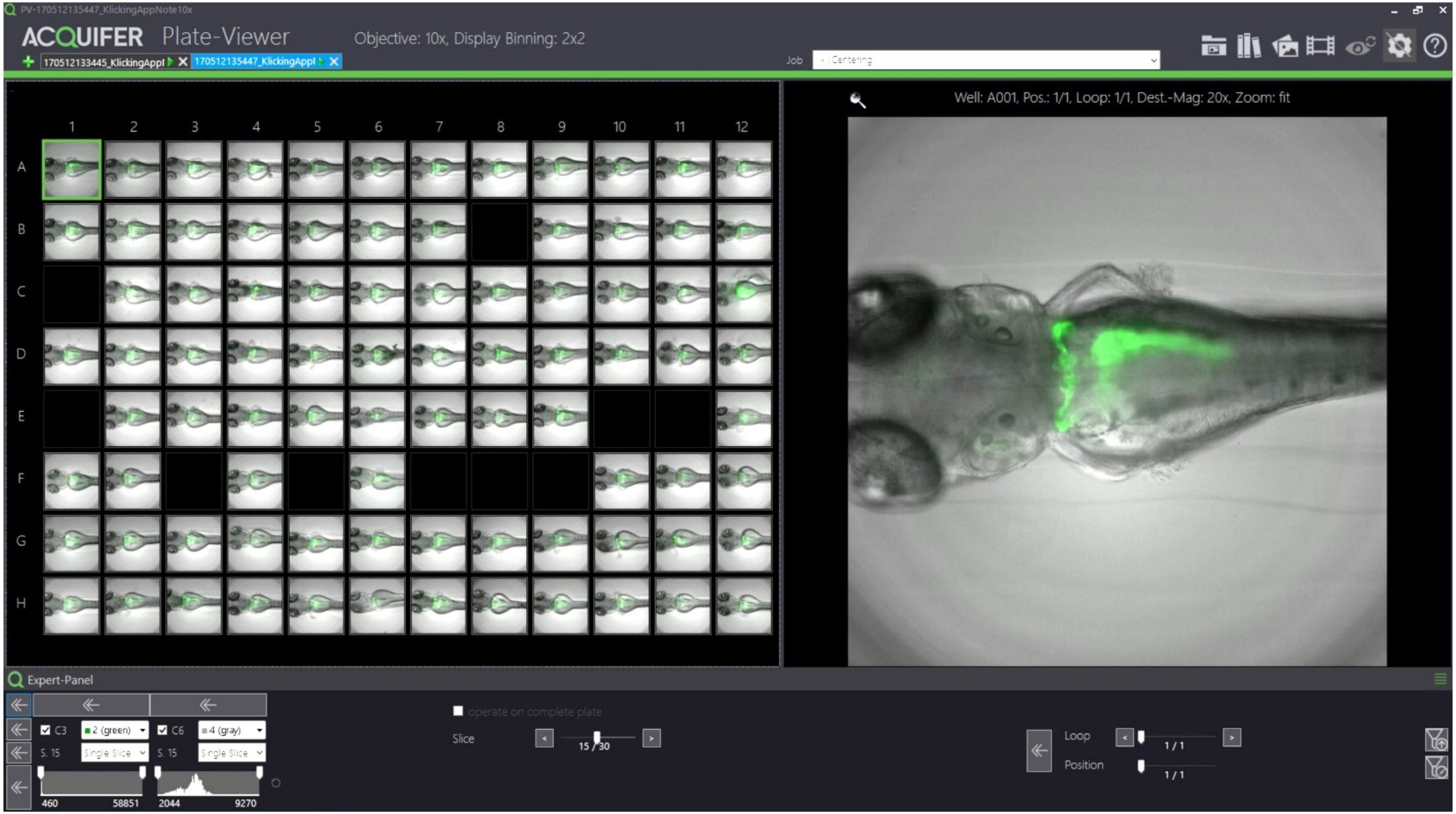
Slice 2 / 4

Loop 1 / 1

Position 1 / 1

Figure 50: View of the ACQUIFER PlateViewer after execution of a ROI detection plugin

See legend page 132.


PV-170512135447_KlickingAppNote10x
ACQUIFER Plate-Viewer
170512133445_KlickingAppl
170512135447_KlickingAppl
1
2
3
4
5
6
7
8
9
10
11
12
A
B
C
D
E
F
G
H
Expert-Panel
Loop
Position

Objective: 10x, Display Binning: 2x2

Job: Centering

Well: A001, Pos.: 1/1, Loop: 1/1, Dest.-Mag: 20x, Zoom: fit

☑ C3 2 (green) S. 15 Single Slice 460 58851

☑ C6 4 (gray) S. 15 Single Slice 2044 9270

□ operate on complete plate

Slice 15 / 30

Loop 1 / 1

Position 1 / 1

Figure 51: View of the ACQUIFER PlateViewer after execution of the semi-automated acquisition

See legends page 132.

Figure 50: View of the ACQUIFER PlateViewer after execution of a ROI detection plugin

On the left are the images from an overview acquisition (brightfield, 2X magnification) with zebrafish larvae mounted in a 96-well plate using preformed agarose gel (Wittbrodt et al. 2014). For each thumbnail, 2 bounding-boxes can be seen, resulting from the execution of the single-template-matching plugin. The inner bounding-box is the detected region. The outer bounding-box is aligned with the inner bounding-box and shows a preview of the field of view for the higher-magnification objective. The image on the right is the currently selected image (A1), which contains the image-region used as template (outlined in red) here corresponding to the yolk region. As it can be seen in the thumbnails, the search was successful for all selected wells (in dash yellow).

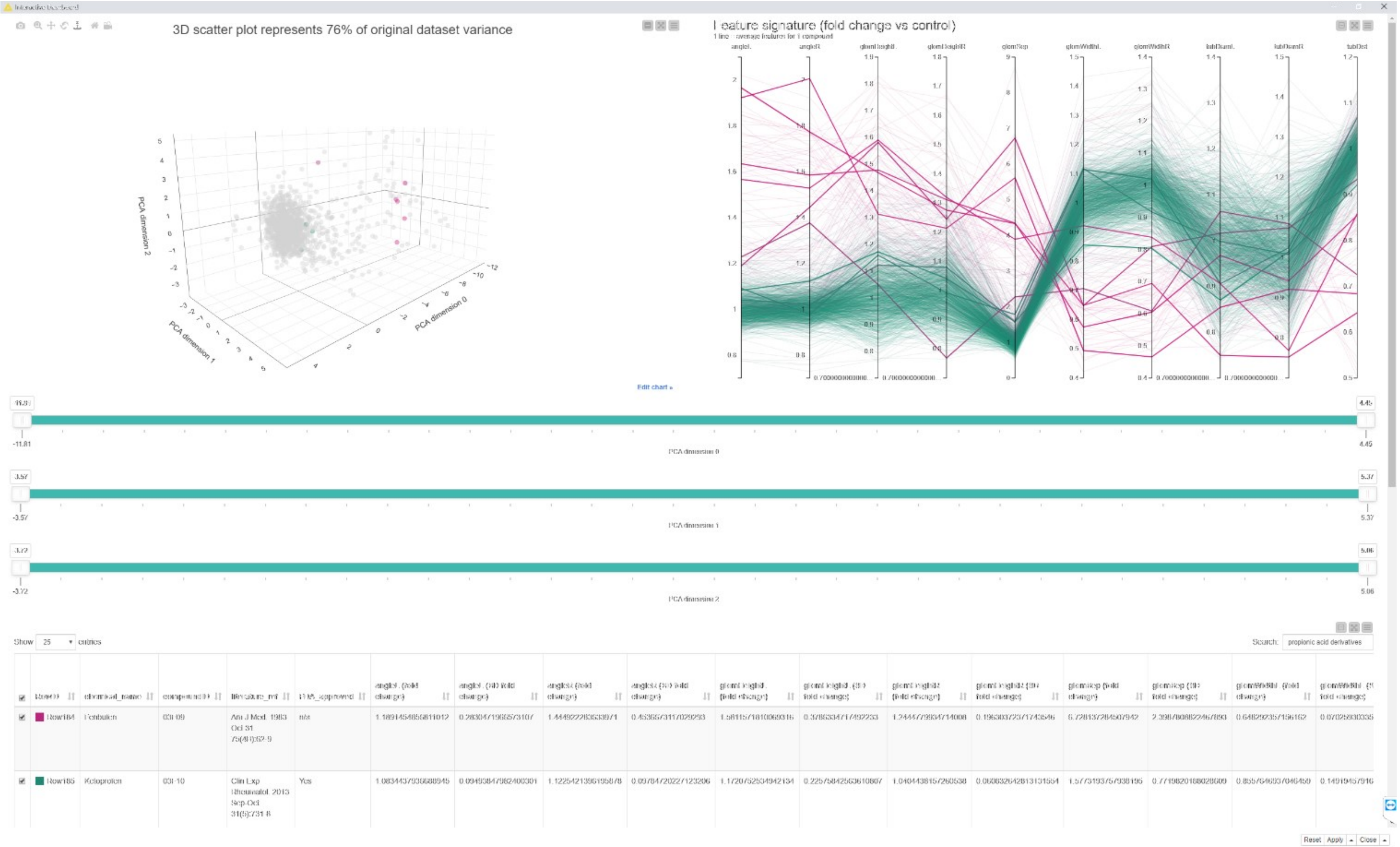
Figure 51: View of the ACQUIFER PlateViewer after execution of the semi-automated acquisition

Using the PlateViewer, an existing acquisition protocol for the imaging at high magnification can be updated to specifically image the regions previously outlined either manually or using a plugin. Here the image shows the result for the imaging using the 20X magnification objective, from the regions localized in Figure 50. Only the wells previously selected for the ROI-detection with the plugin were imaged at high magnification.

3.8. Interactive exploration of a quantitative dataset

3.8.1. A generic feature exploration dashboard in KNIME

To facilitate the exploration of large datasets of multiple samples each described by numerous quantitative features (measurements, statistical quantity, image features, etc.) an interactive exploration dashboard was designed in KNIME. The workflow takes as input a data table with samples along the rows and features along the columns. The dashboard is generated by a single KNIME workflow, relying on javascript view nodes for the visualization. 3 panels are included in the dashboard: a 3D scatter plot of the data-distribution, a parallel coordinates plot and the original data table (Figure 52). Sliders to adjust the axes range of the 3D scatter plot are also part of the dashboard. In the parallel coordinates plot, each line depicts the original feature values for a given sample. Similarly, for the 3D scatter plot, every point corresponds to a sample. The x,y,z-coordinates of the samples in this 3D space, are obtained by dimensionality reduction of the normalized sample features using principal component analysis (PCA), and using the first 3 components as coordinates. The dimensionality reduction allows to approximate the dataset distribution by reducing the initial n-sample features to 3 related PCA-coordinates (see introduction section 1.3.5 and method section 2.9.3). Samples with similar features will have similar line profiles in the parallel coordinates plot and similar positions in the 3D scatter plot. The 3D scatter plot is complementary of the parallel line plot. It allows the identification of cluster of similar samples more easily and offers a less crowded representation. The fidelity of the approximate representation provided by the 3D scatter plot is expressed by the fraction of explained variance for the original dataset. It is automatically calculated by the workflow and should be sufficient (>65-70%). The panels are interactive and inter-connected such that selecting one or multiple samples in one panel will highlight the corresponding items in the other panels, while the scatter plot can be rotated, magnified or cropped to highlight a fraction of samples.



3D scatter plot represents 76% of original dataset variance

Feature signature (fold change vs control)

1 line = average features for 1 compound

PCA dimension 0: -11.81 – 4.45

PCA dimension 1: -3.57 – 5.37

PCA dimension 2: -3.72 – 5.06

Show 25 entries

Search: propionic acid derivatives

☑	RowID	chemical_name	compoundID	literature_ref	FDA_approved	angleL (fold change)	angleL (SD fold change)	angleR (fold change)	angleR (SD fold change)	glomLengthL (fold change)	glomLengthL (SD fold change)	glomLengthR (fold change)	glomLengthR (SD fold change)	glomSep (fold change)	glomSep (SD fold change)	glomWidthL (fold change)	glomWidthL (SD fold change)
☑	🟥 Row184	Fenbufen	03I-09	Am J Med. 1983 Oct 31 75(4B):52-9	n/a	1.1891454856811012	0.2830471966573107	1.4449222835338971	0.4536573117029293	1.5811571810069316	0.3786334717492233	1.2444779934714008	0.1963037237174354 6	6.728137284507942	2.3987808822467893	0.648292357156162	0.07025830335
☑	🟩 Row185	Ketoprofen	03I-10	Clin Exp Rheumatol. 2013 Sep-Oct 31(5):731-8	Yes	1.0834437936688945	0.0949384798240030 1	1.1225421396195878	0.0978472022712320 6	1.1720752534942134	0.2257584256361080 7	1.0404438157260538	0.0608326428131315 54	1.5773193757938196	0.7719820188028609	0.8557546937046459	0.14919457916

Figure 52: View of the interactive feature exploration dashboard

See legend on next page.

Figure 52: View of the interactive feature exploration dashboard

The dashboard contains a 3D scatter plot (top left), a parallel coordinates plot (top right), 3 sliders to adjust the range of the scatter plot axes, and the original result table (bottom). One sample of the dataset corresponds to one table row, one point of the scatter plot and one line of the parallel coordinates plot. To illustrate the interactivity of the panel in the figure above, a few samples in dark pink are highlighted in the scatter plot and parallel coordinates plot, while the other items are greyed out. Hovering the mouse over the scatter plot displays the name and coordinates of the corresponding sample. The panel above shows the distribution of morphological organ features, in a toxicity screening assay (see section 3.8.2). Each sample corresponds to the average feature profile for a particular chemical compound, and the colour code represents another readout for that compound: the proportion of treated specimen that showed a normal kidney morphology. (green: 100% - all specimens with normal morphologies – dark pink: 0% - all abnormal morphologies). Own contribution, reuse allowed according to the CC-BY license. See (Westhoff et al. 2020) for a similar panel.

3.8.2. Application to a toxicity screening dataset

The exploration dashboard was used for the visualization of quantitative features from a toxicity screening in zebrafish (Westhoff et al. 2020). In this study, the toxicity of drugs from a library of compounds approved for clinical use in human (Prestwick Chemical Library®, Prestwick Chemical, D'Illkirch, France), was tested in developing zebrafish larvae of the transgenic line *Tg(wt1b:EGFP)*. The study focuses especially on the toxicity on the developing kidney, which is fluorescently labelled in this transgenic line. From the microscopy images of both the larvae and the fluorescent kidney, multiple phenotypes were scored including 10 morphological measurements describing the kidney for each embryo exposed to a compound (Figure 53). This morphological profiling was repeated for 10-12 replicate specimen for each compound, to derive an average morphological profile response per compound. In total, 4.2 TB of data corresponding to >15,000 embryos treated with 1,280 compounds were acquired and processed. Identifying compounds with similar toxicity responses is challenging with such large datasets but can be facilitated by adapted visualization solutions such as the interactive dashboard presented in the previous section. With this dataset, the parallel line plot shows the average morphological features response for each compound (1 line = 1 compound) in fold-changes compared to the control specimens (Figure 52). The colour code was chosen to represent for each compound the proportion of specimens with a normal kidney morphology after treatment (green: 100% - all specimens with normal morphologies – dark pink: 0% - all abnormal morphologies). From this parallel line plot, one can notice a cluster of "green" compounds which morphological profiles are indeed close to the control profile (fold-change value close 1). This cluster can also be seen in the 3D scatter plot (see Figure 52, colours greyed out due to highlighting of some compounds). Other compounds show noticeable impact on the morphology of the developing kidney, with some features such as the glomerular separation reaching up to 7 times the control value for some compounds (Figure 54).

Among impaired phenotypes, one morphology was more common, with rather elongated pronephroi such as in B-E, H, J-L in Figure 54, with typically higher values than the control for the first 5 features and lower values for the last 5 features, as can be seen in the parallel coordinates plot.

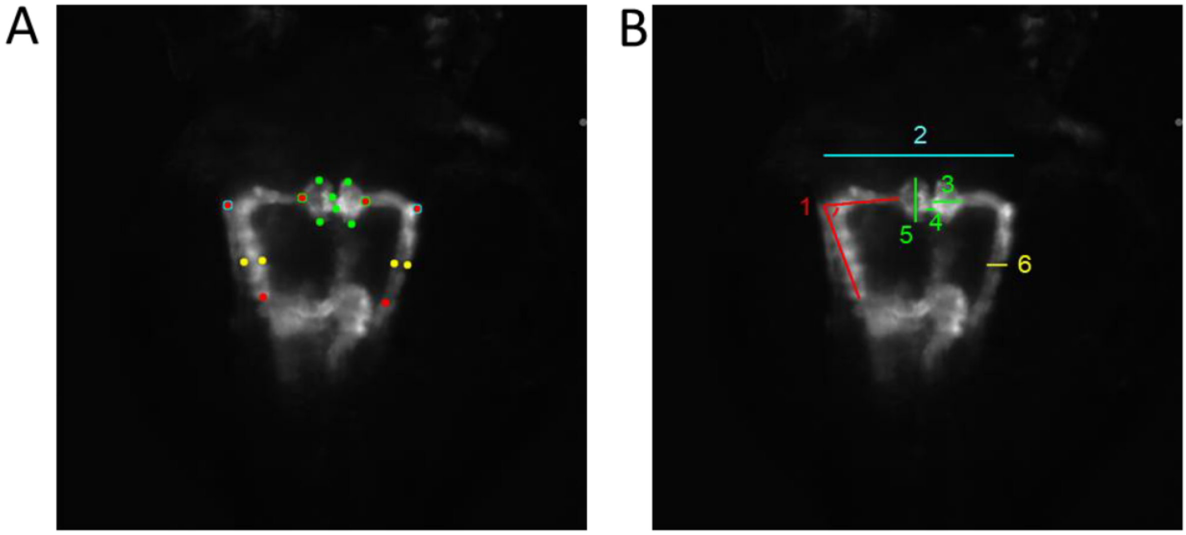


Figure 53: Images of the fluorescently labelled kidney in zebrafish of the transgenic *Tg(wt1b:EGFP)* line with manually scored phenotypes

10 morphological measurements were reported for each specimen, by manually localizing reference points as in **(A)**. **B)** Corresponding distance and angle measurements derived from the reference points (1: pronephric angle - left side 2: tubular distance, 3: glomerular height - right side, 4: glomerular separation, 5: glomerular width - left side, 6: tubular diameter - right side). Except for distances 2 and 4, each measurement was reported for both side of the organ, but only one side is depicted for clarity. Reproduced from (Westhoff et al. 2020)**.**

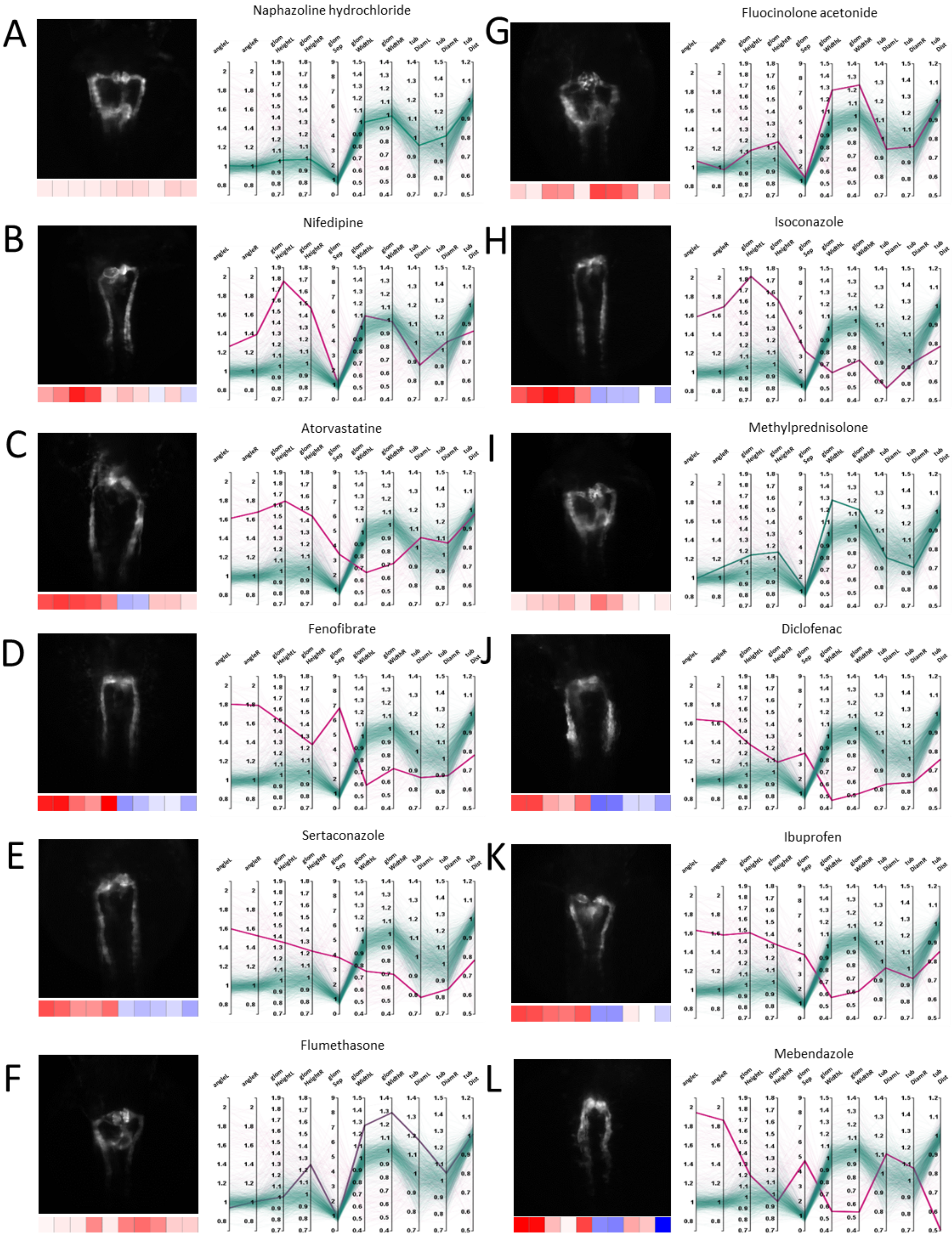


Figure 54: Selection of compounds impacting the morphology of the developing kidney

See legend on next page.

Figure 54: Selection of compounds impacting the morphology of the developing kidney

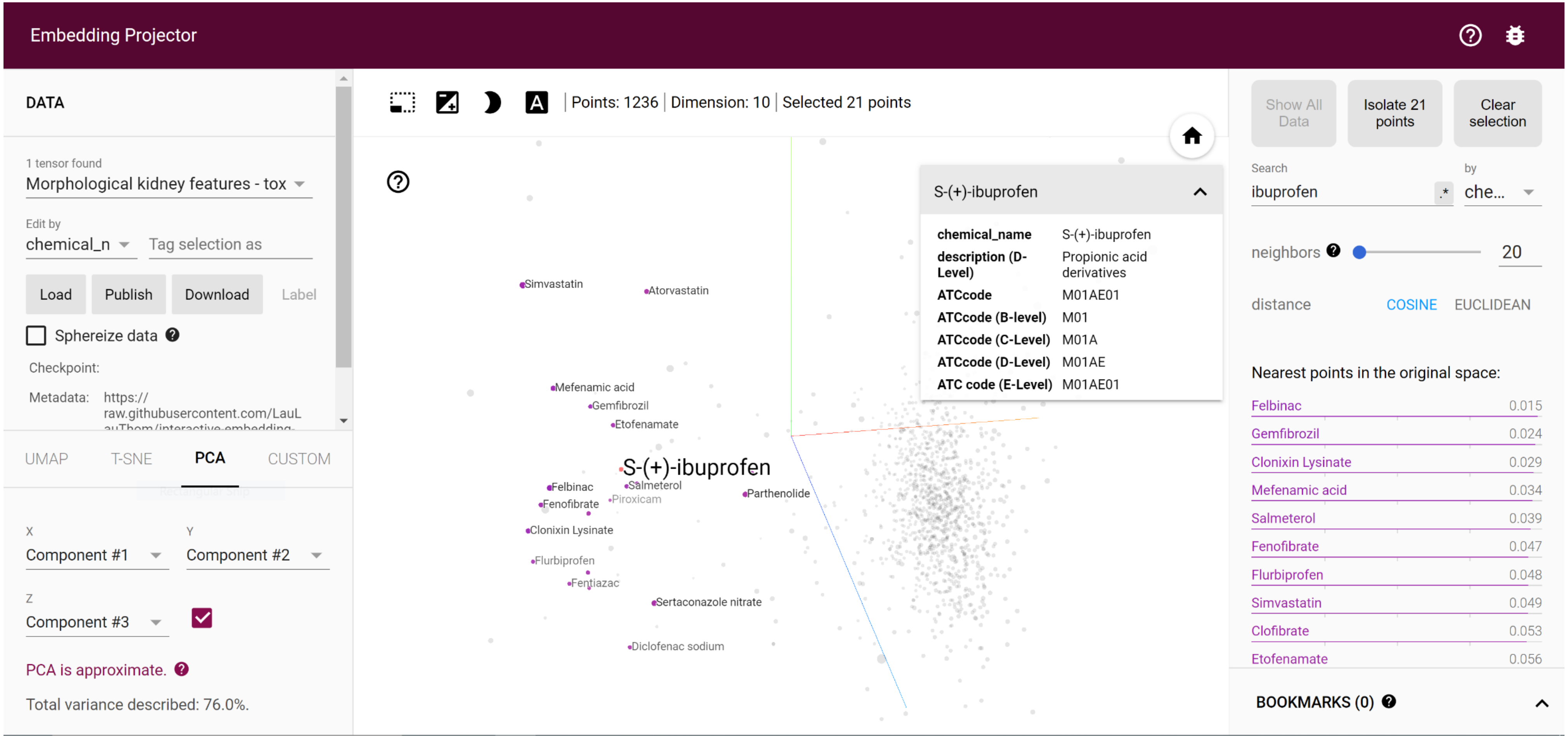
Illustrative examples of drug-induced phenotypic changes for several compound classes. For each compound a thumbnail image, a heatmap visualization (below thumbnail) and a parallel coordinates plot of quantitative morphological features are shown. For the parallel coordinates plots, features are expressed in fold-change compared to the controls, while for the heatmap the Z-score of this ratio is represented. In the parallel coordinate plots, thick lines indicate the compound of interest and thin lines represent all other compound treatments.; colour code as in with 0% (red, all abnormal) to 100% (green, all normal). Reproduced from (Westhoff et al. 2020).

3.8.3. Similar visualization using the TensorFlow embedding projector

A similar visualization was achieved using the TensorFlow embedding-projector platform, freely available online (https://projector.TensorFlow.org/). The data can be uploaded to the platform in the form of text files and is automatically rendered in the browser as an interactive point cloud using dimensionality reduction. Compared to the KNIME dashboard, the platform offers in addition to PCA, alternative methods for dimensionality reduction such as t-SNE and UMAP, which can provide a better discrimination of the data-distribution with some datasets. Figure 55 shows the TensorFlow projector platform when visualizing the same dataset than in Figure 52, also using PCA for dimensionality reduction. As expected, the cloud of points is comparable to the related panel in Figure 52. Individual samples can also be rendered using their identifier (here compound name) instead of spheres (not shown here for clarity). Moreover, the platform has a built-in nearest-neighbour search functionality, to help identifying samples with similar features. This functionality is illustrated in Figure 55 with the compound Ibuprofen for the identification of the 20 closest compounds.

Figure 55: View of the TensorFlow embedding projector interface

(see Figure next page) - The same toxicity dataset than in Figure **52** is shown here, such that each point corresponds to a chemical compound. Compounds with similar feature-profiles are localized in similar area of the point cloud. The visualization is interactive and allows rotation, translation and zooming. Using the built-in nearest-neighbour search, the 20 compounds (in violet) with closest response to the selected compound Ibuprofen (in red) are shown here. The list of neighbour compounds is also shown on the right side, together with the cosine distance between the original feature vectors of the neighbour compounds and Ibuprofen. Reproduced from (Westhoff et al. 2020). The interactive figure can be directly accessed online at the following link: http://projector.TensorFlow.org/?config=https://raw.githubusercontent.com/LauLauThom/interactive-embedding-kidney/main/configTFvis.json



Embedding Projector

DATA

1 tensor found
Morphological kidney features - tox

Edit by
chemical_n
Tag selection as

Load | Publish | Download | Label

□ Sphereize data

Checkpoint:

Metadata: https://raw.githubusercontent.com/LauLauThom/interactive-embedding-

UMAP | T-SNE | **PCA** | CUSTOM

X: Component #1
Y: Component #2
Z: Component #3 ☑

PCA is approximate.

Total variance described: 76.0%.

Points: 1236 | Dimension: 10 | Selected 21 points

S-(+)-ibuprofen

chemical_name	S-(+)-ibuprofen
description (D-Level)	Propionic acid derivatives
ATCcode	M01AE01
ATCcode (B-level)	M01
ATCcode (C-Level)	M01A
ATCcode (D-Level)	M01AE
ATC code (E-Level)	M01AE01

Show All Data | Isolate 21 points | Clear selection

Search: ibuprofen
by: che...

neighbors: 20

distance: COSINE | EUCLIDEAN

Nearest points in the original space:

Felbinac	0.015
Gemfibrozil	0.024
Clonixin Lysinate	0.029
Mefenamic acid	0.034
Salmeterol	0.039
Fenofibrate	0.047
Flurbiprofen	0.048
Simvastatin	0.049
Clofibrate	0.053
Etofenamate	0.056

BOOKMARKS (0)

Figure 55: View of the TensorFlow embedding projector interface

See legend on previous page.

4. Discussion

4.1. Annotation tools

This section describes the benefits and novel usages enabled by the ImageJ ROI-group functionality and the Fiji annotation plugins resulting from this project. It also discusses the limitations of previously available solutions, and possible perspectives of development.

4.1.1. The ROI category index

This original contribution enables novel usage of ROI objects in ImageJ/Fiji. ROIs category indexes can be used to annotate different populations of objects such as cells (dead vs alive, normal vs aberrant) or subparts of larger objects (organs, etc.). These annotations can then be turned into ground-truth information to train supervised object-detection frameworks, segmentation models or image-classification algorithms. While dedicated solutions for such ground-truth annotations exist, they are usually not adapted to scientific images, and not easily extensible. The group attribute is part of the ImageJ1 API, and thus accessible by any scripting language or plugin, hence allowing its reuse for custom applications in both ImageJ and Fiji. This new functionality does not only benefit plugin developers, it greatly facilitates routine image annotations. In particular, the category index of ROI is reported in the result table of ImageJ/Fiji together with quantitative measurements, which facilitates the aggregation of the results into statistical estimates for each category.

4.1.2. Roi 1-click tools

With the Roi 1-click tools a single-click is sufficient to execute a set of commands that would typically require multiple clicks or keyboard shortcuts, which quickly become time-consuming when repeated for each ROI. The annotation process is thus dramatically simplified, especially for large datasets or large number of objects. The tools have broad applicability where standard shapes should be outlined, or when a rough outline of objects is sufficient. They were initially motivated by the need for a semi-automated solution for the “eye-assays” in zebrafish larvae (see section 1.2.2 (Hanke et al. 2015; Hentschel et al. 2007)), to rapidly outline the eye-region and quantify the signal of a fluorescent reporter in this region. Yet the tools have various applications, such as the annotation of ground-truth object-locations with rectangular bounding-boxes or circles, and associated categories when combined with the ROI-group attribute (see 3.2.1). For instance, the tools were used to generate rectangular ROIs of fixed aspect ratio for the training of cascade detectors (see section 3.4.1). The tools can also be used as a pre-processing step to crop interesting image-regions according to standard dimensions, thus reducing the size of the dataset. Working with predefined image-dimensions is often beneficial when processing large amounts of data in batch. For instance, with deep learning models a classical pre-processing is the resizing of the images to fixed dimensions, which may deform the objects if the original images vary in sizes.

4.1.3. Qualitative image-annotations

Describing images with a few keywords is a fundamental need in many disciplines involving imaging, for instance in biomedical research for the categorization of samples or the description of complex phenotypes, in clinical imaging for image-based diagnostics, or in manufacturing for the description of object-properties. These qualifiers may report the presence or discrete count of features, describe a morphology, evaluate quality criteria, or assign images to non-exclusive categories. Automated image-classification methods exist, but they are usually designed to assign images to a single exclusive category, besides requiring substantial efforts for their implementation and validation (Lakhmi Prasanna et al. 2017; Olaode et al. 2014). Especially, any supervised classification method requires manual ground-truth annotations in the first place. For small datasets of a few dozen images, image identifier and qualitative descriptors can be reported in a simple spreadsheet. However, for larger datasets or large number of descriptors, this becomes quickly overwhelming and error prone as one needs to inspect a multitude of images while appending information to increasingly complex tables. Although software tools for such manual qualitative annotations and compatible with large datasets have been proposed (Wagner and Gehrig 2019; Westhoff et al. 2020), there was so far no widespread solution available in common user-oriented image analysis software. The qualitative annotation plugins for Fiji reported here, facilitate and standardize routine image annotations, with descriptive keywords. The plugins have various applications, from simple image evaluation to ground-truth category annotations for the training of supervised machine learning algorithms. The results table have standard structures, compatible with automated analysis by custom pipelines as illustrated with the KNIME example workflows. These example workflows should help non-expert users applying advanced visualization and analysis methods on their dataset, while more advanced users can further customize the workflows to their needs.

4.1.4. Addressing the lack of scientific image-annotation tools

Although machine-learning is getting more popular for scientific image-analysis, there is a lack of broadly applicable annotation tools for the generation of ground-truth data. Existing tools are usually designed to annotate real life photographs and not scientific images, which raises issues such as incompatible file format or display range (ex: 8-bit only images). Moreover, they are not easily extensible and support a limited set of annotation formats.

Using existing scientific image-analysis software such as ImageJ/Fiji for image annotations has several benefits. First those software are well established in the community, with a large pool of users who feel comfortable with this familiar software environment. Then these software support multiple image-formats, display options and functionalities to explore the dataset. Moreover, the annotations can be part of larger workflows (e.g. image pre-processing before annotating). With Fiji for instance, the multi-template-matching plugins can be used to quickly provide ground-truth bounding-boxes which positions can be manually refined if necessary. Finally, the software can be extended with custom scripts or plugins thanks to publicly available API (Application Programming Interface), thus allowing any standard or custom ground-truth format to be encoded.

4.2. Object-detection

While various object-detectors are available and have been described in this dissertation, they often rely on similar mechanisms. This section details such mechanisms, common practices and alternatives to the presented solutions.

4.2.1. Localization strategies

The object-detection algorithms presented here rely on different localization strategies. Template matching and cascade detectors rely on sliding windows, while the deep learning detectors use more efficient region-proposal methods or anchor boxes. The sliding window search was historically widely used and is one of the most reliable way to precisely localize objects that can be located anywhere in an image. However, it is a brute-force approach that can become quickly computationally expensive if lots of features are computed for each position of the sliding window. Another drawback is that the size of the sliding window is fixed, therefore to detect object of various sizes, either the search is repeated with artificially scaled versions of the image (pyramid), or multiple search with sliding-windows covering a range of sizes are carried out. In both cases, this further increases the computation time. Modern object-detectors completely abandoned the sliding-window approach for more efficient solutions such as region proposal algorithms. This strategy was popularized by the Region-based Convolutional Neural Network – RCNN by (Girshick et al. 2016). Another alternative is to predict the bounding box position and dimensions as part of the detector output, in addition to prediction of the object-classes. These detectors contrast with the previous proposals for which the localization and classification were always two distinct stages. The advantage of such single-stage detectors is an even faster detection speed, with comparable to slightly worse detection accuracy compared to the 2-stage detectors depending on the implementation (Zhou et al. 2019b). The most popular 1-stage detectors include YOLO (Redmon et al. 2016), SSD (Liu et al. 2016) and RetinaNet (Lin et al. 2018).

For biomedical applications, the number of object classes is typically way smaller than the hundreds of classes those detectors were initially demonstrated with, therefore the difference of detection accuracy between the 1-stage and 2-stage detectors is likely not noticeable for such applications. Besides, real-time detection is in a majority of cases not a requirement, e.g. for feedback microscopy the time-limiting factor is usually the multi-dimensional acquisition for a given position of the stage.

4.2.2. Beyond bounding boxes

Most object-detectors predict object-locations with rectangular bounding-boxes, aligned with the axis of the image. Rectangular bounding-boxes are generic and convenient for the manual annotation of objects which can differ in shapes and sizes. They are also easy to handle from a programming point of view. However, for cells or other round structures as often encountered in life-science for instance, rectangles are not ideal, and will typically include a significant portion of background regions. Therefore, post-processing such as segmentation within the detected bounding-box is necessary to recover the object-shape (He et al. 2017). Using encoder-decoder deep learning architectures, alternative object-detection solutions have been proposed, following a bottom-up approach to construct the outline of objects, rather than a top-down selection strategy from a pool of candidate rectangular regions. For instance, the outline of objects can be constructed from a set of key points, predicted by the model using methods similar to pose-estimation (Zhou et al. 2019a; Zhou et al. 2019b). This strategy was applied to the detection of glomerulus in histology images of kidney tissues, by predicting the centre and radius of circles outlining glomerulus (Yang et al. 2020), or for the detection of cells with unregular shapes (Jiang et al. 2020). Novel segmentation models have also been proposed to outline the object of interest such as CellPose (Stringer et al. 2020), or StarDist (Schmidt et al. 2018; Weigert et al. 2020) which extends on the segmentation network U-Net (Ronneberger et al. 2015).

4.2.3. Data-augmentation

Data-augmentation is often proposed as a straightforward solution to improve the robustness of machine-learning models when the availability of ground-truth data is limiting (Shorten and Khoshgoftaar 2019). With cascade detectors, the Fiji plugin for the generation of the training set reported here has an option to generate additional images by rotation and flipping of the annotated regions. While this can indeed improve the diversity of the training set, the type of transformations applied to the data should be carefully selected to reflect realistic transformations. For instance, performing rotations of the training images is not adapted if the object is known to have a fixed orientation. In such cases, data-augmentation might even hamper the training process and slow down the convergence, by virtually creating a more complex classification problem than the reality.

4.2.4. About scale and rotation

Most object-detectors are by design rotation and scale sensitive, meaning that they do not recognize the same way an object appearing at different scales or angles. Indeed, the visual patterns used for the recognition have fixed scales and orientations. This is valid for all the object-detectors reported here (template matching, cascade detectors, deep-learning detectors). With machine learning detectors, robustness to scale and rotation is typically achieved by having object at various scales and angles in the training set, to force the model to learn different features. Similarly, for template matching, rotated and scaled objects can be detected by using templates representing a discrete set of angles and scales. Cascade detectors are however scale-invariant thanks to the features which can be resized. Novel CNN architectures have also been proposed to learn features at multiscale automatically from a single image (Godinez et al. 2017). Still in most cases, reducing the experimental object variability facilitates the detection by algorithms.

4.3. Comparison of the presented methods

In this document, 3 object-detection methods are reported: multi-template-matching, cascade detectors and CNN object-detectors. Those methods have different advantages and drawbacks, especially when considering detection capacities, ease-of-use and compatibility.

4.3.1. Accessibility

One major limitation for computer vision methods is the technical burden associated with the installation of specific libraries or software, and the number of parameters that should be optimized for a specific use-case. The multi-template-matching implementation proposed here was designed to be easy to install, with a limited set of parameters to facilitate its usability. It is thus very straightforward to apply on custom data, especially compared to machine learning solutions which first require the annotation of ground truth data and the training of a model. The cascade detectors do not raise major installation issue neither, however they come with a multiplicity of parameters both for the training and the detection, which are not intuitive for non-initiated users. Deep learning methods often require specific hardware and the installation of multiple libraries (CUDA, TensorFlow, Keras, MXNET). The training of deep learning models also involves multiple parameters (number of epochs, batch size, learning rate, etc.), yet these are well documented and conserved between implementations. Besides, for the detection, only a score threshold is often required.

4.3.2. Accuracy

Multi-template matching is limited to a discrete number of recognition patterns represented by the templates, and therefore offers less flexibility than machine learning detectors which can typically account for a wide diversity of patterns, with little additional computational cost. Still the pixel intensity signature used by template matching was shown to be robust to variation of the object intensity due to change of illumination, noise or other sources of experimental variability. Yet, for classification this robustness results in a limited capacity to distinguish subtle variations of intensity patterns. Cascade detectors have access to a larger set of recognition patterns, however, the Haar-like features represent rectangular intensity patterns which might not respond well to unregular shapes observed in biology. Alternative features such as local binary patterns adapted to the distinction of textures are available with these detectors and could be explored. Still, the complexity of the cascade is often limited by the size of the training set, it is thus usual to observe a non-negligible amount of false positive detections with small to medium size training sets at first attempts with these detectors. Training the convolutional layers of deep learning detectors allows the recognition of patterns specific to the training images. Yet, training a model from scratch is time consuming and requires many images, but we showed that transfer learning or fine tuning with pre-trained models efficiently reduces this burden while still resulting in highly efficient image-classifiers and object-detectors.

4.3.3. Speed

It is challenging to achieve comparable benchmarks between the detection methods since the execution time depends on a lot of factors, including the programming language, the execution environment, the libraries used etc. Here the most comparable benchmarks are between the Fiji implementation of multi-template matching and cascade detectors, which rely on the same OpenCV libraries, converter functions and are executed in the same environment. For the detection of medaka embryos, the search with a single template is about one order of magnitude slower than with a cascade detector with full resolution images (250 ms and 25 ms respectively – see medaka detection Figure 31 and Figure 39). However, with template matching, multiple templates are often necessary to achieve comparable detection capacity than the cascade detectors, which further increases the detection time. Although, the detection with a cascade likely depends on its structure (number of stages, and features per stage), cascade detectors clearly offer higher detection speed than template matching with full resolution images. This was observed with the Fiji plugins for all the datasets benchmarked here. This difference might be explained by the nature of the features (costly normalised cross-correlation vs Haar-patterns computed by integral images) and the optimized cascading scheme. Nevertheless, the template matching search also scales with the size of the image and template, therefore a search at lower resolution would provide a speed up (see section 4.4.4). According to the literature, deep learning detectors are also expected to run in real-time. Here the detection is embedded in a python node within a KNIME workflow, which is not ideal as the images are serialized between java on the KNIME side and python. We reported the execution time for the full python node (0.7 secs/image, see section 3.5.4) which includes pre-processing the images and generating the output (mask image, etc.), and therefore is not truly indicative of the model detection speed.

4.4. Multi-Template Matching

A major outcome of this research is a versatile implementation of multi-template matching for object-detection. This section discusses the advantages of this method especially compared to the previously available single-template matching, the trade-off between detection capacity and speed, and options to optimize the method.

4.4.1. Advantages

Multi-Template Matching is an object-detection method particularly adapted to non-expert users. It is straightforward to use and benchmark, with a limited set of parameters (score threshold, overlap threshold, and if know the expected number of objects). The implementation reported here is well integrated into scientific image-analysis software, is easy to install and well documented. Although there is no need for data pre-processing, classical filters could be applied to potentially improve the detection (denoising, background subtraction). Multi-template matching method was shown to be efficient for the detection of objects in microscopy images, especially when the objects have fixed scale and limited changes of orientation, which can be achieved with dedicated sample mounting. Besides, using a normalized score for the computation of the score map, the template matching search is relatively robust to changes of illumination between the reference template and the image. This object-detection method is generic by design and was successfully applied with different imaging modalities (see 4.4.7: “Adoption by the community”).

4.4.2. Comparison with single-template matching

Single-template matching is limited to searching for a single intensity pattern, rendering it particularly sensitive to changes in orientation or perspective of objects. In comparison, multi-template matching has enhanced detection capacities by combining results from multiple template images to increase the range of detectable intensity patterns. A critical parameter for the success of the detection is thus the choice of the templates, which must be representative of the objects to localize. Besides, all pixels of the template image contribute equally to the similarity score, including pixels from the background regions. A small fraction of background around the object in the template image can however help improving the selectivity of the detection pattern. Instead of an exhaustive search with multiple templates, possibly more efficient methods for rotation and scale invariant template matching have been reported (Choi and Kim 2002; Fredriksson et al. 2007; Kim and De Araújo 2007; Marimon and

Ebrahimi 2007); however, no implementation is publicly available, and their complexity limits their usability by non-experts.

4.4.3. Capacity vs time

The search with multiple templates effectively improves the range of detectable patterns but it also comes at a proportional increase of computational time, as the iterative sliding-window search is repeated for each template. Only a discrete set of template images can be used for the search, representing a fraction of the infinite possible object states. Therefore, a trade-off should be found between the expected variability of the object and the number of templates. For microscopy application, sample mounting strategies can help reducing object-variability (e.g. standardized orientation (Wittbrodt et al. 2014)) and thus the need for additional templates. Additionally, mounting systems can constrain the samples and thus the template matching search to a limited image region, hence speeding up execution (Figure 30.C) and reducing the probability of false positive detections compared to the search with the full-field image.

4.4.4. Faster detections by downscaling

An alternative to speed up the execution of multi-template matching is to perform the search with down-scaled versions of the templates and images, and to rescale the detected bounding-boxes to the original image size. This strategy is employed by many deep-learning detection architectures that takes small size images as input to reduce memory consumption. In the present implementation of multi-template matching, this option is not proposed as it increases the complexity of the solution, besides reducing the position accuracy of detected bounding-boxes and possibly leading to false positive detections as the downscaling degrades the searched intensity pattern. However, advanced users can refer to the online tutorial[3] of the python implementation for an example of how to use downscaling to accelerate the detection.

4.4.5. Template matching with normalised score

For the computation of the score map, the described Multi-Template-Matching implementation provides access to the normalized template matching scores exclusively (normalized square-difference, normalized cross-correlation and 0-mean normalized cross-correlation). Normalization offers multiple advantages: first it offers better detection results when the template and image show

[3] https://github.com/multi-template-matching/MultiTemplateMatching-Python/blob/master/tutorials/Tutorial3-SpeedingUp.ipynb

shifted illuminations which is likely to happen in microscopy if different illumination intensities are used. Secondly, it assures that the score ranges from 0 and 1, while the non-normalized score are not bound to a finite range. This bounded range enables the comparison of scores resulting from different templates on the same image, independently of the size or intensity of the templates. Finally, having a bounded score makes it easier for users to define a score-threshold, and to propose a default threshold for the implementation.

4.4.6. With multi-dimensional images

The template matching search is only defined for 2-dimensional grayscale images. With multi-channel images, there are 2 options: either projecting the multiple channels to a single grayscale image (e.g. projection from RGB to grayscale) or selecting one channel for the detection. Similarly, there is no support for 3D template, however the search can be executed with a 2D template individually for each slice of a volume.

4.4.7. Adoption by the community

Less than a year after its publication in the journal BMC Bioinformatics and a bit more than a year after its publication as a Biorxiv preprint, the Multi-Template-Matching implementation seems to have been adopted by the international scientific community. The first citations of Multi-Template-Matching reports applications to different fields than life-science including monitoring 3D-printing (Petsiuk and Pearce 2020), and digitalization of medical records (Rho et al. 2020). While this may be unexpected, it also shows the robustness and generic potential of this object-detection method. It is virtually impossible to know how many people use the Fiji implementation, however the python implementation was bookmarked on GitHub at this day (02/09/2020) by 40 different users from various places in the world, including Germany, France, India, China, Taiwan, USA and Mexico. Besides, the YouTube tutorials count a bit more than 600 views. While this project was presented at several occasions as a poster or short-talks at international conferences in Europe, a major proportion of users seems to have discovered the project online via the referencing of the GitHub repositories, the ImageJ wiki page, the BMC Bioinformatics journal or the YouTube tutorials.

4.5. Cascade detectors

Cascade detectors are a promising alternative for object-detection and this project proposed a novel implementation of this method in bioimage analysis software. The major difficulty with such detector is the optimization of the model training, especially with small datasets. This section thus reviews critical parameters for the training of a model on custom data, and discuss advantages and drawbacks compared to other object-detection methods.

4.5.1. Advantages and drawbacks

Cascade detectors are widely available object-detectors thanks to their implementation in the OpenCV library, which is even compatible with mobile devices such as smartphones. Neither the training nor the detection requires particularly powerful hardware, and a trained cascade is contained in a lightweight xml file. The training is relatively fast on CPU with medium size dataset, thanks to the readily available Haar-patterns. Moreover, the detection can be performed in real-time on conventional cameras. On microscopy images, we also reported high-detection speed on full resolution images (20-50 ms per image with 2048 x 2048 pixel images). However, cascade classifiers did not benefit from major developments in the last years and tend to be replaced by more robust deep-learning solutions. The training of cascade detectors is also relatively complex with multiple parameters and little documentation available. Here we mostly experimented with the true positive rate, false positive rate and number of stages, while other parameters could be adjusted to influence the structure of the cascade, such as the maximum number of weak classifiers per stage but also the choice of alternative feature sets such as local binary patterns instead of Haar-like patterns (Liao et al. 2007). We often observed a non-negligible number of false positive detections on test sets and supposed that the automatic generation of the negative training set is not particularly efficient and should be completed by manually annotated “harder” negative training instances. Finally, cascade detectors only work for grayscale images, and with a single class at a time, although simultaneous detection of multiple object-classes can be achieved by running the detection with dedicated cascade detectors in parallel.

4.5.2. Training a custom detector with small training sets

In the original paper (Viola and Jones 2001), the author reports a cascade face-detector containing 32 stages, that was trained with about 5000 faces and 10000 non-faces images. For microscopy applications, generating such large training sets is often not achievable, some dozens to a few hundred images is a more reasonable number depending on the setup. Consequently, with such size of training set, the number of stages in the cascade is usually much more limited, as the training quickly runs out of training images. Indeed, with a false positive rate of 50% for the individual classifiers, the number of ground-truth negative instances is reduced by about a half at every stage. Meanwhile, multiple stages would be necessary to compensate for the high false positive rate of the individual classifiers. For instance, a 5-stage cascade with 0.5 false positive rate for each classifier would yield a cascade detector with $0.5^5 = 0.03$ false positive rate, meaning that starting with 100 negative instances, after 5 stages the number of negative instances available for training is given by $100 \text{ x } 0.5^5 = 100 \text{ x } 0.03 \sim 3$ images. A more efficient strategy with small training set is thus to aim for fewer cascade stages (3 to 5), and lower false positive rates for the individual classifiers (e.g. 0.1 instead of 0.5). This results in using more features for the individual classifiers and thus a slightly slower detection but better accuracy.

4.5.3. Importance of the negative training set

The cascade-training implementation available with OpenCV automatically generates a high number of negative image instances by randomly cropping background images multiple times. While this is convenient to automatically generate negative ground-truth data, in practice the resulting training instances seems not challenging enough. For instance, in microscopy images, this routine may yield a majority of plain background regions. As a result, the model training converges quickly, as the negative instances are rapidly discarded by the first classifiers but the detection on test images yields a lot of false-positive hits. It is thus advised to manually annotate challenging negative image regions (e.g. bubbles, edges…) in addition to the automated cropping routine. This should improve the diversity of the negative training set and *in-fine* the performance of the model.

4.6. Deep learning models

Deep learning was used in this project for image-classification and object-detection. This section reviews the advantages and limitations of deep learning methods for such tasks, the implications of transfer learning compared to *de-novo* model training and the choice of a model architecture. The performance of the MXNET/GluonCV library is also briefly discussed and possible alternatives are introduced.

4.6.1. Advantages

Deep learning models have been employed for multiple applications of computer vision with impressive results. One advantage is the flexibility of these models, which architectures can be adapted to the characteristic of the dataset and to custom prediction task. Indeed, while most models were originally designed for RGB colour images, custom architectures can be engineered to work with grayscale, multi-channels or 3-dimensional images (Çiçek et al. 2016). Besides the architecture, optimal features for a given task are directly derived from the training set when a model is trained *de-novo*. This allows the adaptation to new datasets, without having to adjust the parameters for every step of the workflow as in conventional image-processing. Moreover, these features can account for complex image patterns, and thus provide more flexibility than predefined image-features (Pawlowski et al. 2016). Finally for classification, a single deep learning model can be trained to recognize a very large number of classes (Simonyan and Zisserman 2015).

4.6.2. Accessibility

While many programming libraries are available for deep-learning, there is no gold-standard solution when it comes to training a custom deep learning model on original data. Some implementations might be limited to a single network architecture (e.g. YOLO). Others might support multiple architectures but require more efforts to adapt the code or vary in the amount of documentation. For this project, the MXNET (Chen et al. 2015) and GluonCV (Guo et al. 2020) libraries were used in python, as they offer flexibility, good performance, and are well documented. Still with those libraries, training a deep-learning object-detector on custom data required adapting a significant amount of code. By adapting this code to work within KNIME workflows we reduced the complexity of the code to a simpler visual workflow with intuitive graphical user-interfaces. Besides, the workflows facilitate the execution of the program with custom data and offer advanced image-visualization functionalities particularly convenient for large datasets. However, the workflows are only exposing a fraction of selected parameters (number of epochs, batch size…), which should be adapted to new datasets. Other parameters such as learning rate and regularization parameters are set to default values expected to provide satisfactory results in most cases. Advanced users can modify these values in the python code embedded in the workflow or by adapting the workflow to expose these parameters.

However, the lack of dedicated ground-truth annotation tool adapted for the training with GluonCV/MXNET did not facilitate the benchmarking of this framework. We reported here a rather convoluted workflow for the generation of the training set which involves several annotation rounds, intermediate steps and software (Fiji and KNIME). This complexity was partly due to the lack of support for the annotations of different object-categories in ImageJ/Fiji. However, using the novel ROI-group attribute (see section 3.2.1), the generation of the .lst-file could be achieved using a single Fiji script, after annotating all object-classes in a unique annotation round. Finally, one limitation with the object-detection tutorial available for GluonCV (GluonCV Contributors n.d.) is that the model training does not include the evaluation of the model performance on a validation set. Therefore, no indications about the performance of the model on unseen images is provided along the training. Having a validation set is helpful to adjust the training parameters, and to rapidly identify issues with the training such as overfitting on the training set (Shorten and Khoshgoftaar 2019).

4.6.3. Performance

The training of deep-learning models requires loading and processing several images. There are thus 2 potential bottlenecks for the model training: first the access to the data on disk and then the computational power. For the latter, GPU are classically used as they provide large amount of dedicated memory and a pool of specialized and highly parallelized computation threads. Here we indeed observed faster model training on GPU compared to the CPU. However, a major speed-up (16x on CPU and 6x on GPU) was also achieved when using the binary rec format (Figure 43). The rec-file contains both the ground-truth annotations and down-scaled versions of the training images, stored as contiguous memory blocks on disk (GluonCV Contributors 2020a). This promotes a highly optimized access to the training data. In comparison, the lst-file is a text file containing the paths to the original full resolution images, which occupy various locations on disk and are costly to load in memory due to their large sizes. Therefore, even with a powerful GPU, the training will noticeably run slower with the lst format due to the data-access bottleneck. The access to full resolution images, might explain why the training on GPU with the lst file raises an “Out of memory error”. The rec format is however less convenient as it requires an additional conversion step from the lst file and is not human-readable contrary to a simple text file like the lst format.

4.6.4. Transfer learning and fine-tuning

There are different options when working with deep learning: the choice of the model architecture, and the training strategy. While designing a custom model architecture allows full flexibility about the model input, output, and intermediate operations, additional benchmarking is necessary to validate the model. Besides, the training must be started from scratch and thus requires substantial ground-truth data. Using pre-existing model architectures validated for specific purposes allows to save this effort and to potentially reuse pre-trained models with custom data by transfer learning/finetuning (Zhu et al. 2011), thus reducing the training time and the ground-truth annotation effort. Here we adopted this strategy both for the training of an object-detector (section 3.5.2), and of an image-classifier (section 3.6.2). The resulting models showed high-detection and classification accuracy even with small amount of ground-truth data. Especially for classification, the pre-trained model base for feature extraction was frozen, meaning that the pre-existing recognition patterns were not adapted to the custom dataset (transfer learning exclusively). Only the newly added fully connected layers for classification were trained, which still offered high accuracy. Training the model base to further adapt the convolutional layers to custom data (fine-tuning) is possible though and may improve the classification accuracy (Dietz 2020; Rosebrock 2014a).

4.6.5. Model architectures

In terms of architecture, larger models can account for more complex relationships between the model inputs and outputs but are more at risk of overfitting. For the training of a deep learning object detector, only a single model architecture was evaluated (SSD with MobileNet), although the GluonCV/MXNET library supports a diversity of models which can be exchanged almost transparently, provided the image pre-processing is adapted accordingly (see model zoo - GluonCV Contributors n.d.). The SSD architecture tested offered very satisfying results therefore there was no real interest in testing other architectures. Most models supported by GluonCV/MXNET were published with high detection accuracy (>90%) for the detection of various real-life objects classes from standard datasets (Deng et al. 2009; Everingham et al. 2010). They mostly differ in the execution speed and memory footprint, although most of them run in real-time or close to real-time on standard computers. Therefore, one could expect similar performances between those models for object-detection in microscopy images. For instance, (Waithe et al. 2020) reported similar performances for the detection of cells when using either YOLO, Faster-RCNN or RetinaNet. Similarly for the classification of macromolecular crystals, (Bruno et al. 2018) reported little difference between the Inception-v3, Inception-ResNet-v2 and variant of the NASNet architectures.

One important parameter is the input image dimensions expected by the network. Indeed, the pre-processing typically involves downscaling the original images to comply with the model requirements, which depending on the original dimensions might be detrimental, especially for small objects or structures. An alternative when this happens might be to divide the input image in patches that are processed separately.

4.6.6. Alternatives

The ImageAI library (Moses and Olafenwa 2018) was briefly tested as an alternative to MXNET/GluonCV for object-detection. This library is well documented too and allows the training of a custom YOLO object-detector (Redmon et al. 2016) with just a few lines of code. However, the training time was slower than with GluonCV/MXNET, most likely due to less efficient data-loading. More recently, the Auto-Gluon library was released (Erickson et al. 2020), with the goal to make classical deep-learning tasks more accessible, by reducing the need to adapt standard code. As it builds up on MXNET and GluonCV, it could potentially simplify the work reported here.

For the ground-truth annotations of object-position and category, other standard file formats such as the PASCAL VOC format (Everingham et al. 2010) are used instead of the .lst or .rec reported here. Graphical user interfaces readily exist to annotate object-regions in images and generate the corresponding VOC file (Tzutalin 2015; Wada 2016). This format is compatible with GluonCV/MXNET too, however it requires formatting the dataset accordingly (GluonCV Contributors 2020b) and some modifications of the training code.

4.7. Automated microscopy

Automated microscopy has a central role for high-throughput imaging. This section reports important experimental considerations before implementing automated acquisition methods and discusses possible perspectives to generalize their applicability to various imaging setups.

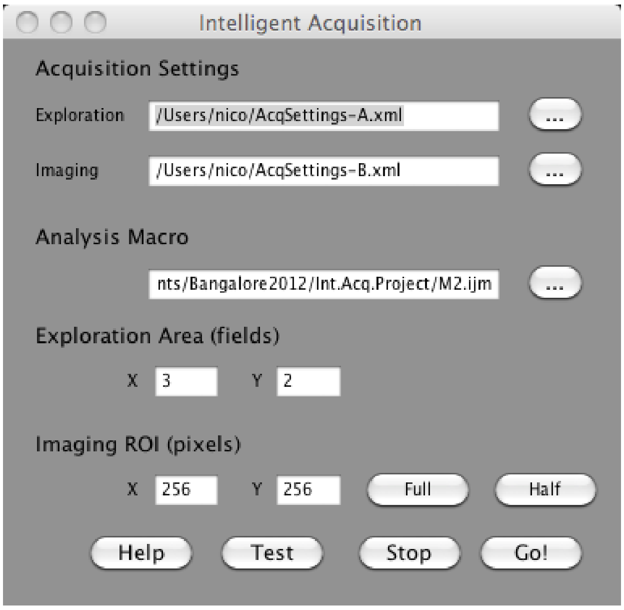
4.7.1. Feedback microscopy vs semi-automated acquisition

While feedback microscopy has a great potential to streamline complex acquisition protocols of large sample collections, full automation also means that not intervention from the user is possible to correct false-positive detections for instance. Therefore, it is primordial to have well calibrated detection algorithms and to systematically control that the acquisition is imaging the right objects. One possibility for feedback microscopy is thus to put a high score-threshold for object-detection or classification to force a selective process and make sure that most imaged regions are correct. If the objects are scarce or variable, it might be necessary to decrease this threshold though to avoid missing some objects. Another risk is a selection bias from the detection algorithm, which might better recognize some phenotypes than others and thus alter the perception of the real object-distribution. Such biases have been observed with CNN face-detectors showing variable performances depending on the skin colour or age (Nagpal et al. 2019). Those risk can be reduced by thoroughly benchmarking the detection algorithms on previously acquired datasets, to confirm that they work as expected for these specific objects and imaging conditions. Indeed, despite extensive benchmarking reported in the literature, one can never reliably predict the behaviour of an existing object-detectors for novel imaging modalities and objects. Notably, for machine-learning algorithms, it is important for the training set to be representative of the effective object variability.

Semi-automated acquisition therefore offers a safer alternative, by giving the user a chance to quickly review the results of the detection before starting a potentially voluminous acquisition. Despite the discontinuous acquisition flow, semi-automated acquisition still has the potential to reduce the user's intervention, by saving the need to manually indicate regions of interest in each image. In the ACQUIFER PlateViewer, semi-automated acquisition is also easier than feedback microscopy, as the PlateViewer takes care of converting pixel coordinates in machine coordinates.

4.7.2. Toward standard protocols

One difficulty which is probably limiting the spread of feedback microscopy is the technical barrier associated to hardware interfacing, and its coupling with image-analysis solutions. Using the ACQUIFER Imaging Machine (IM), writing text files to communicate with the microscope is relatively accessible, but does not allow to have a fine chronological control of the sequence of events, namely because the acquisition software is the master controller. An alternative available with the IM and other commercial systems is to rely on a communication protocol to actively control the system, often programmatically via an API. Classical communication protocols work through a TCP/IP interface (like for the IM) or a COM port. These communication protocols require some technical knowledge though, and differ from one microscope system to the other, which hinder the dissemination of generic feedback microscopy protocols. Therefore, some open-source initiatives such as Micro-Manager (Edelstein et al. 2010) or PyME (PYME contributors 2020) aim at facilitating these communication protocols by providing pre-configured hardware adapters, a standardized API of common functions and an intuitive graphical user interface. Thanks to the hardware adapters and the standardized function API, acquisition protocols could be exchanged between systems, provided equivalent hardware components are available. As an example, those packages would provide a single universal function to move the microscope stage or the objective to a specific position. As part of the standard API, this function would forward this request to the equivalent system-specific function from the API of the microscope manufacturer, according to the hardware configured in the open-source software. Micro-Manager readily allows feedback microscopy experiments with dedicated modules for on-the-fly image-processing and targeted acquisition (Figure 56),(Stuurman 2012). While this module is limited to detection scripts available for ImageJ, the project is open to contributions, and the underlying design could be promoted to a community standard.



Intelligent Acquisition

Acquisition Settings

Exploration /Users/nico/AcqSettings-A.xml

Imaging /Users/nico/AcqSettings-B.xml

Analysis Macro

nts/Bangalore2012/Int.Acq.Project/M2.ijm

Exploration Area (fields)

X 3 Y 2

Imaging ROI (pixels)

X 256 Y 256

Figure 56: MicroManager module for targeted acquisition using feedback microscopy

The module requires to pre-configure a first acquisition protocol (“Exploration” in the GUI above) for the overview of the sample. The images from this first acquisition are then analysed by the Analysis Macro, for the detection of ROIs. The detected regions are then imaged according to a second acquisition protocol (“Imaging” in the GUI above). Reproduced from the Micro-Manager documentation (Stuurman 2012).

4.8. Image-classification

Two strategies were proposed for image-classification: template-matching and deep learning. This section reviews the advantages, drawbacks and critical parameters when applying these methods on custom image-datasets.

4.8.1. Classification with template matching

We demonstrated binary image classification by comparing the images to classify with a single reference template image, and by setting a threshold on the template matching score. Images get either classified as matching the category of the reference template or not. We showed that this strategy allows the identification of organ-morphologies that deviate from a reference morphology. By comparing with multiple templates representative of different classes, multiple class classification can be achieved. We illustrate this second option for the recognition of different organ-morphologies, however for this particular dataset the accuracy of the classification was not as good as the binary classification, since the morphologies only differed slightly in terms of pixel intensity distribution, resulting in similar score between the different classes (Figure 45.B).

This classification strategy effectively corresponds to a nearest-neighbor search (Dhanabal and Chandramathi 2011), as the image is assigned to the class of the most similar template. To account for the variability within a class, here mostly due to variable fluorescence expression within the kidney tissues, the templates were generated by intensity projection from a set of reference images, previously aligned using center of mass. (Cole et al. 2004) applied a similar classification strategy to recognize Lego blocks. As pre-processing they also centered the Lego blocks based on the center of mass and aligned the major axis of the blocks to a fixed orientation. Alternatively to account for more diversity within a class, multiple templates could be provided for each class, and the classification could be formulated as a k-nearest neighbors (with k typically ranging from 3 to 10), for which the k-top matching templates are returned and a majority vote from the represented classes is used to finally classify the image.

For nearest-neighbor searches, the search time is proportional to the number of neighbors (here templates), and the size of the template and images. (Cole et al. 2004) proposed an optimized search strategy using clustering of similar images to avoid the comparison with every single template. Still, like for template matching, the choice of the templates and the variability of the object within a class will greatly influence the outcome of the classification with this method. Importantly, while template matching is advantageous for localization thanks to its robustness to morphological changes (shown with the head region in zebrafish larvae, see section 3.3.5), inversely it is not particularly efficient at distinguishing objects with similar intensity signatures as observed with the long and cystic kidney morphologies (see Figure 45.C).

4.8.2. Classification with deep learning

Deep learning models are particularly efficient at distinguishing multiple object classes (Simonyan and Zisserman 2015). Here we used transfer-learning to re-purpose a model initially trained for real-life photographs classification, to the classification of organ morphology in microscopy images (see section 3.6.3). Despite the high variability of morphology or fluorescence expression within the organ over the dataset, the classification accuracy was close to 100%. Moreover, we did not consider the number of training images when annotating the different classes, which resulted in an imbalanced dataset for the training set, with more images representing the normal phenotype than the cystic phenotype. Although, this distribution could be representative of the reality, with the proposed workflow, it may result in models that are more biased towards predicting one class (Ramyachitra and Manikandan 2014). Indeed, here the misclassified instances are cystic phenotypes classified as normal. It is rather advised to use comparable number of images for each class, either by reducing the size of the over-represented class or by increasing the number of images in the minority classes e.g. using data-augmentation (Rocca 2019; Shorten and Khoshgoftaar 2019). However, data-augmentation can also introduce artifacts (see section 4.2.3). When training instances for a class are scarce (e.g. rare events), a more complex but efficient solution consists in increasing the penalty for misclassification of images belonging to this class (Ramyachitra and Manikandan 2014; Rocca 2019).

Importantly with supervised learning methods, the training set should be representative of the expected samples’ characteristics. With images, this means using similar acquisition modalities and pre-processing between the ground-truth images and images that will be classified. While pre-trained models for specific applications can be found (segmentation, denoising), it might be preferable to retrain or finetune the model on custom data to avoid any artifacts (Weigert et al. 2018).

4.9. Interactive data exploration

This project demonstrated interactive visualization using dimensionality reduction in KNIME and with the TensorFlow embedding projector interface. This section briefly reports the advantage of such visualization methods and the limitations of PCA for this task.

4.9.1. Exploration dashboards

The KNIME workflow generating the dashboard was designed as a generic tool for the exploration of quantitative datasets. It can be executed with any tabular dataset with minimum adaptation. Only for the PCA, the features should cover similar value ranges to avoid bias toward one feature. This can be achieved using normalization, here Z-score normalisation was used. The TensorFlow embedding projector platform offers similar visualization functionalities. It has the advantage that it does not require any installation, and that the visualization can be directly accessed via a web-URL when the data is hosted in a public repository like GitHub (see Figure 55). However, the platform displays less information than the KNIME interactive dashboard, in particular it does not show the original feature values nor the data table.

These interactive visualization solutions greatly facilitate the exploration of large quantitative datasets by biomedical researchers. They can help identifying similar samples or guide following classification strategies. Moreover, they allow the relation between the quantitative phenotypic information and the qualitative metadata such as the name of a chemical compound, a chemical class or molecular target. Other user-oriented platforms are available for the exploration of data using dimensionality reduction such as Orange (Demšar et al. 2013) or Sleepwalk (Ovchinnikova and Anders 2019), but were not explored here.

4.9.2. PCA for dimensionality reduction

Dimensionality reduction is a great method to get a visual overview of complex datasets. However, in some cases it can be misleading. For instance, when comparing a dataset between reduction to 2D and 3D, 2 datapoints might seem close to each other's in 2D but actually more distant when adding the 3rd dimension. This consecutively applies to the following higher dimensions. One reason for that is that dimensionality reduction is an approximation of the original dataset distribution and that there is an intrinsic mathematical difficulty to render the original distances from N-dimensions to only 2 to 3 dimensions (Kuo and Sloan 2005).

In this report, only PCA was explored mostly because it is historically widely used, it is free of any parameters and from a theoretical point of view is simpler than other dimensionality reduction methods such as t-SNE or UMAP. However, according to (Maaten and Hinton 2008) PCA is among the "*linear techniques that focus on keeping the low-dimensional representations of dissimilar datapoints far apart. For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping".* It is thus wise to compare the visualization after PCA with other non-linear dimensionality reduction methods such as t-SNE and UMAP. This is possible with the TensorFlow embedding projector web interface (Figure 55), or using the Sleepwalk web app (Ovchinnikova and Anders 2019), which was explicitly developed to compare different visualizations and combinations of parameters. Importantly, those 2 platforms make the relation between the dimension-reduced visualization and the original pairwise sample-distances in the original feature space, which helps identifying when the visualization is locally representative of the original data-distribution. When using PCA, the percentage of the original dataset variance represented by the low-dimension encoding is also an indicator of reliability.

Finally, it is important to pre-process the data before dimensionality reduction including for instance normalization of the feature values such that they cover similar ranges, to avoid having one feature dominating the low-dimension encoding.

4.10. Benefits for microscopy-based screening of small organisms

The software solutions presented in this dissertation can benefit microscopy-based screening of small organisms at several steps of the experimental workflow and interplay well together. First the ROI-detection algorithms are data-driven, either in the form of user-provided templates or labelled training images. This results in an improved adaptability and versatility towards new image-sets compared to classical image-processing workflows and to a better tolerance to object-variability. Moreover, the selectivity of a given detector can be simply tuned using the score-threshold. These detectors can be used for automated microscopy protocols, or as a pre-processing step in custom analysis workflows. Especially, they facilitate the localization of sub-regions of large specimens such as a specific organ or tissue, which is particularly challenging with intensity-based methods when no specific labelling is available, e.g. in brightfield images of whole organisms.

The annotation tools enable the manual handling, inspection and scoring of image-datasets acquired by screening microscopy. The tools have potential applications in multiple image-analysis scenarios: from ground-truth category annotations for supervised classification of specific phenotypes, to quantitative and qualitative measurements of images or ROIs. They provide a valuable tool set for the straightforward analysis of even large data collections. Finally, the annotations can be exploited for data-exploration and image-classification, to identify samples with similar phenotypes or with deviant responses, and to relate phenotypic scores with the underlying biology, such as the mode of action of a compound or a signalling pathway.

Although illustrated on datasets from zebrafish and medaka embryos, the tools are generic and applicable to various image-datasets including cell cultures. Finally, they provide access to complex computer vision or data-science methods in common user-oriented image-analysis software and enable their reusability in custom workflows.

5. Conclusion

This dissertation reports the design, development and benchmarking of novel research software inspired by the field of computer vision and data-science. The aim was to create versatile and robust solutions tailored to the requirements of microscopy-based phenotypic screening studies in small model organisms. The resulting software tools address various steps of the screening workflow, from manual ground-truth annotations, automated detection of regions of interest, targeted imaging of specific tissues or organs using feedback microscopy, image-classification, and interactive data exploration. Importantly, the tools are generic by design and were benchmarked on several microscopy datasets of zebrafish larvae and medaka embryos. They are particularly suitable for phenotypic screening studies at the tissue- and organ specific level. The software is easy-to-use and readily accessible to biomedical researchers with little to no prior knowledge of computer vision or image-processing. The tools are integrated in common scientific image-analysis packages and accompanied by extensive documentation in the form of articles in academic journals, readme files accompanying the source codes and online video tutorials. To foster their distribution and the inspiration of derived work, most of the underlying source code is available online in open-source repositories. This project demonstrates the translation of methods originating from the field of computer vision into bioimage analysis solutions. Computer vision has the potential to provide more robust alternatives to classical image-processing workflows based on e.g. intensity thresholding.

I propose here innovative software solutions to tackle previously unaddressed limitations of microscopy-based screening studies with small model organisms. The resulting toolsets and documentations will empower biomedical researchers to conduct complex imaging workflows and to extract novel knowledge from phenotypic screening studies.

For this research, only a fraction of the multitude of resources offered by the field of computer vision has been explored. Especially, further development could address segmentation methods to complement the rough localization by rectangular bounding boxes, the tracking of object over time or the detection of object in three-dimensional volumes. Finally, by providing widely available implementations of complex methods from the fields of computer vision in user-oriented software, this research also fosters similar interdisciplinary innovations which could benefit biomedical imaging.

6. Zusammenfassung

Das Thema dieser Dissertation ist das Design, die Entwicklung und das Benchmarking neuartiger wissenschaftlicher Software, welche durch die Fachbereiche Computer Vision und Data-science inspiriert wurde. Das Ziel war das Entwickeln von vielseitigen und robusten Lösungsansätzen, die auf die Anforderungen von mikroskopiebasierten, phänotypischen Screeningstudien in kleinen Modellorganismen zugeschnitten sind. Die hierbei entstandenen Softwares adressieren verschiedene Arbeitsschritte von Screening-Experimenten: die manuelle Ground-Truth-Annotation, die automatische Erkennung von Zielregionen (ROI), gezieltes Mikroskopieren von spezifischen Geweben und Organen mittels Feedback-Mikroskopie, Bildklassifikation and interaktive Datenexploration. Die Methoden wurden dabei generisch designed, an verschiedenen Mikroskopiedatensätzen von Zebrafisch- und Medakaembryonen getestet und sind insbesondere für das gewebe- und organspezifische phänotypische Screening geeignet. Die Software ist benutzerfreundlich und für biomedizinische Forscher mit wenig Programmiervorkenntnissen leicht zugänglich. Sie wurde in gängige wissenschaftliche Bildanalyse-Software integriert, und eine umfassende Dokumentation in Form von wissenschaftlichen Veröffentlichungen, quellcodebegleitenden Nutzungsanleitungen und Online-Video-Tutorials wurde erstellt. Um die Verbreitung der entstandenen Software und ihre Verwendung in neuen Projekten zu erleichtern, ist der gesamte Quellcode in Open-Source-Repositorien hinterlegt. Die hier entwickelte Lösung demonstriert die Methoden der Computer Vision in softwarebasierten BioImaging-Anwendungen. Computer Vision Methoden stellen potentiell robustere Alternativen zu klassischen Bildverarbeitungswerkzeugen dar, z.B. als intensitätsbasiertes Thresholding. In dieser Arbeit werden innovative Softwarelösungsansätze für bisher unbearbeitete methodische Limitierungen im Bereich des mikroskopiebasierten Screening von alternativen Modellorganismen vorgestellt. Die entstandenen Werkzeuge und Dokumentationen ermöglichen biomedizinischen Wissenschaftlern neuartige, komplexe Mikroskopie-Workflows durchzuführen und neues Wissen aus phänotypischen Screeningstudien abzuleiten. In dieser Forschungsarbeit wurde nur ein Bruchteil der vielfältigen Ressourcen aus dem Bereich der Computer Vision untersucht. Insbesondere die Nutzung von Segmentierungsmethoden, welche die relativ grobe Lokalisierung mittels Bounding Boxen ergänzen, das zeitliche Nachverfolgen (Tracking) von Objekten oder die Detektion von Objekten in dreidimensionalen Volumina könnten weiterführende und aufbauende Arbeiten darstellen. Schlussendlich fördert das Bereitstellen von leicht einsetzbaren Implementierungen von nun gekapselten, komplexen Methoden aus dem Bereich der Computer Vision, weitere interdisziplinäre Innovationen im Bereich des biomedizinischen Imaging.

7. References

Aaron, J., Wait, E., DeSantis, M. and Chew, T. (2019). **Practical Considerations in Particle and Object Tracking and Analysis**. Current Protocols in Cell Biology *83*, e88, doi:10.1002/cpcb.88.

Abbe, E. (1873). **Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung**. Archiv f mikrosk Anatomie *9*, 413–468, doi:10.1007/BF02956173.

Abdi, H. and Williams, L. J. (2010). **Principal component analysis: Principal component analysis**. WIREs Comp Stat *2*, 433–459, doi:10.1002/wics.101.

ACQUIFER (2017a). **ACQUIFER smart imaging**, *ACQUIFER website*, URL: https://www.acquifer.de/downloads/AppNote_Smart_Imaging.pdf [Accessed October 2020].

ACQUIFER (2017b). **The ACQUIFER PlateViewer**, *ACQUIFER website*, URL: https://www.acquifer.de/downloads/Clicktoolappnote_ImagingMachine.pdf [Accessed October 2020].

Ahmadi, A. (2017). *Cascade Trainer GUI*, URL: http://amin-ahmadi.com/cascade-trainer-gui/ [Accessed July 2020].

Alessandri, K., Andrique, L., Feyeux, M., Bikfalvi, A., Nassoy, P. and Recher, G. (2017). **All-in-one 3D printed microscopy chamber for multidimensional imaging, the UniverSlide**. Scientific Reports *7*, doi:10.1038/srep42378.

Aleström, P., D'Angelo, L., Midtlyng, P. J., Schorderet, D. F., Schulte-Merker, S., Sohm, F. and Warner, S. (2020). **Zebrafish: Housing and husbandry recommendations**. Lab Anim *54*, 213–224, doi:10.1177/0023677219869037.

Alexe, B., Deselaers, T. and Ferrari, V. (2012). **Measuring the Objectness of Image Windows**. IEEE Transactions on Pattern Analysis and Machine Intelligence *34*, 2189–2202, doi:10.1109/TPAMI.2012.28.

Al-Janabi, S., Huisman, A. and Van Diest, P. J. (2012). **Digital pathology: current status and future perspectives**. Histopathology *61*, 1–9.

Amat, F., Lemon, W., Mossing, D. P., McDole, K., Wan, Y., Branson, K., Myers, E. W. and Keller, P. J. (2014). **Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data**. Nature Methods *11*, 951–958, doi:10.1038/nmeth.3036.

Annila, T., Lihavainen, E., Marques, I. J., Williams, D. R., Yli-Harja, O. and Ribeiro, A. (2013). **ZebIAT, an image analysis tool for registering zebrafish embryos and quantifying cancer metastasis**. BMC bioinformatics *14*, S5, URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-S10-S5 [Accessed October 2017].

Antoni, D., Burckel, H., Josset, E. and Noel, G. (2015). **Three-Dimensional Cell Culture: A Breakthrough in Vivo**. Int J Mol Sci 11.

Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A. and Sebastian Seung, H. (2017). **Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification**. Bioinformatics, doi:10.1093/bioinformatics/btx180.

Aung, S. M., Kanokwiroon, K., Phairatana, T. and Chatpun, S. (2019). *Live and dead cells counting from microscopic trypan blue staining images using thresholding and morphological operation techniques*. *9*, 9.

Bakkers, J. (2011). **Zebrafish as a model to study cardiac development and human cardiac disease**. Cardiovascular Research *91*, 279–288, doi:10.1093/cvr/cvr098.

Ball, J. S., Stedman, D. B., Hillegass, J. M., Zhang, C. X., Panzica-Kelly, J., Coburn, A., Enright, B. P., Tornesi, B., Amouzadeh, H. R., Hetheridge, M., Gustafson, A.-L. and Augustine-Rauch, K. A. (2014). **Fishing for Teratogens: A Consortium Effort for a Harmonized Zebrafish Developmental Toxicology Assay**. Toxicological Sciences *139*, 210–219, doi:10.1093/toxsci/kfu017.

Bao, G., Cai, S., Qi, L., Xun, Y., Zhang, L. and Yang, Q. (2016). **Multi-template matching algorithm for cucumber recognition in natural environment**. Computers and Electronics in Agriculture *127*, 754–762, doi:10.1016/j.compag.2016.08.001.

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. and Kreshuk, A. (2019). **ilastik: interactive machine learning for (bio)image analysis**. Nature Methods, doi:10.1038/s41592-019-0582-9.

Berthold, Michael R, Cebron, N., Dill, F., Gabriel, T. R., Kotter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K. and Wiswedel, B. (2009). *KNIME: The Konstanz Information Miner*. 8, doi:10.1007/978-3-540-78246-9_38.

Berthold, Michael R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K. and Wiswedel, B. (2009). **KNIME - the Konstanz information miner: version 2.0 and beyond**. ACM SIGKDD Explorations Newsletter *11*, 26, doi:10.1145/1656274.1656280.

Betzig, E., Patterson, G. H., Sougrat, R., Lindwasser, O. W., Olenych, S., Bonifacino, J. S., Davidson, M. W., Lippincott-Schwartz, J. and Hess, H. F. (2006). **Imaging intracellular fluorescent proteins at nanometer resolution**. Science *313*, 1642–1645.

Booth, M. J. (2007). **Adaptive optics in microscopy**. Phil Trans R Soc A *365*, 2829–2843, doi:10.1098/rsta.2007.0013.

Boothe, T., Hilbert, L., Heide, M., Berninger, L., Huttner, W. B., Zaburdaev, V., Vastenhouw, N. L., Myers, E. W., Drechsel, D. N. and Rink, J. C. (2017). **A tunable refractive index**

matching medium for live imaging cells, tissues and model organisms. eLife *6*, doi:10.7554/eLife.27240.

Boutros, M., Heigwer, F. and Laufer, C. (2015). **Microscopy-Based High-Content Screening**. Cell *163*, 1314–1325, doi:10.1016/j.cell.2015.11.007.

Bouvrie, J. (2006). *Notes on Convolutional Neural Networks*.

Bradford, Y. M., Toro, S., Ramachandran, S., Ruzicka, L., Howe, D. G., Eagle, A., Kalita, P., Martin, R., Taylor Moxon, S. A., Schaper, K. and Westerfield, M. (2017). **Zebrafish Models of Human Disease: Gaining Insight into Human Disease at ZFIN**. ILAR Journal *58*, 4–16, doi:10.1093/ilar/ilw040.

Brady, C. A., Rennekamp, A. J. and Peterson, R. T. (2016). **Chemical Screening in Zebrafish**. In: **Zebrafish: Methods and Protocols**, Eds Kawakami, K., Patton, E. E., and Orger, M. Springer New York, New York, NY, pp. 3–16, doi:10.1007/978-1-4939-3771-4_1.

Bray, M.-A., Singh, S., Han, H., Davis, C. T., Borgeson, B., Hartland, C., Kost-Alimova, M., Gustafsdottir, S. M., Gibson, C. C. and Carpenter, A. E. (2016). **Cell Painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes**. Nat Protoc *11*, 1757–1774, doi:10.1038/nprot.2016.105.

Bray, M.-A., Vokes, M. S. and Carpenter, A. E. (2015). **Using CellProfiler for Automatic Identification and Measurement of Biological Objects in Images: CellProfiler to Identify Biological Objects in Images**. In: **Current Protocols in Molecular Biology**, Eds Ausubel, F. M., Brent, R., Kingston, R. E., Moore, D. D., Seidman, J. G., Smith, J. A., and Struhl, K. John Wiley & Sons, Inc., Hoboken, NJ, USA, p. 14.17.1-14.17.13, doi:10.1002/0471142727.mb1417s109.

Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A. (1984). Classification and regression trees, CRC press.

Bronson, F., Dagg, C. and Snell, G. (1966). **Biology of the laboratory mouse**. McGraw-Hill, New York 187–204.

Bruno, A. E., Charbonneau, P., Newman, J., Snell, E. H., So, D. R., Vanhoucke, V., Watkins, C. J., Williams, S. and Wilson, J. (2018). **Classification of crystallization outcomes using deep convolutional neural networks** Ed. Hu, J. PLoS ONE *13*, e0198883, doi:10.1371/journal.pone.0198883.

Burgmer, C. (2018). MultiLayerNeuralNetwork english.png, URL: https://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetwork_english.png.

Burns, C. G., Milan, D. J., Grande, E. J., Rottbauer, W., MacRae, C. A. and Fishman, M. C. (2005). **High-throughput assay for small molecules that modulate zebrafish embryonic heart rate**. Nature Chemical Biology *1*, 263–264, doi:10.1038/nchembio732.

Butler, A., Hoffman, P., Smibert, P., Papalexi, E. and Satija, R. (2018). **Integrating single-cell transcriptomic data across different conditions, technologies, and species**. Nat Biotechnol *36*, 411–420, doi:10.1038/nbt.4096.

C. elegans: methods and applications (2006). Ed. Strange, K., Humana Press, Totowa, N.J.

Caicedo, J. C., Cooper, S., Heigwer, F., Warchal, S., Qiu, P., Molnar, C., Vasilevich, A. S., Barry, J. D., Bansal, H. S., Kraus, O., Wawer, M., Paavolainen, L., Herrmann, M. D., Rohban, M., Hung, J., Hennig, H., Concannon, J., Smith, I., Clemons, P. A., Singh, S., Rees, P., Horvath, P., Linington, R. G. and Carpenter, A. E. (2017). **Data-analysis strategies for image-based cell profiling**. Nature Methods *14*, 849–863, doi:10.1038/nmeth.4397.

Caicedo, J. C., Roth, J., Goodman, A., Becker, T., Karhohs, K. W., Broisin, M., Molnar, C., McQuin, C., Singh, S., Theis, F. J. and Carpenter, A. E. (2019). **Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images**. Cytometry *95*, 952–965, doi:10.1002/cyto.a.23863.

Caicedo, J. C., Singh, S. and Carpenter, A. E. (2016). **Applications in image-based profiling of perturbations**. Current Opinion in Biotechnology *39*, 134–142, doi:10.1016/j.copbio.2016.04.003.

Caie, P. D., Walls, R. E., Ingleston-Orme, A., Daya, S., Houslay, T., Eagle, R., Roberts, M. E. and Carragher, N. O. (2010). **High-Content Phenotypic Profiling of Drug Response Signatures across Distinct Cancer Cells**. Molecular Cancer Therapeutics *9*, 1913–1926, doi:10.1158/1535-7163.MCT-09-1148.

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. and Sabatini, D. M. (2006). **CellProfiler: image analysis software for identifying and quantifying cell phenotypes**. Genome Biology 11.

Cha, Y. R. and Weinstein, B. M. (2007). **Visualization and experimental analysis of blood vessel formation using transgenic zebrafish**. Birth Defect Res C *81*, 286–296, doi:10.1002/bdrc.20103.

Chang, T.-Y., Pardo-Martin, C., Allalou, A., Wählby, C. and Yanik, M. F. (2012). **Fully automated cellular-resolution vertebrate screening platform with parallel animal processing**. Lab Chip *12*, 711–716, doi:10.1039/C1LC20849G.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z. (2015). **MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems**. arXiv:151201274 [cs], URL: http://arxiv.org/abs/1512.01274 [Accessed August 2020].

Choi, M.-S. and Kim, W.-Y. (2002). **A novel two stage template matching method for rotation and illumination invariance**. Pattern recognition *35*, 119–129.

Chollet, F. and others (2015). Keras, URL: https://keras.io.

Chou, S.-Y. (2019). **AUC - Insider's Guide to the Theory and Applications**, *Sin-Yi Chou - Data-science enthusiast*, URL: https://sinyi-chou.github.io/classification-auc/ [Accessed October 2020].

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. and Ronneberger, O. (2016). **3D U-Net: learning dense volumetric segmentation from sparse annotation**. In: **International conference on medical image computing and computer-assisted intervention**, 2016, Springer, pp. 424–432.

Cireşan, D. C., Giusti, A., Gambardella, L. M. and Schmidhuber, J. (2013). **Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks**. In: **Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013**, Eds Mori, K., Sakuma, I., Sato, Y., Barillot, C., and Navab, N. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 411–418, doi:10.1007/978-3-642-40763-5_51.

Cole, Luke, Austin, D., Cole, Lance, and others (2004). **Visual object recognition using template matching**. In: **Australian conference on robotics and automation**, 2004, URL: http://www.araa.asn.au/acra/acra2004/papers/cole.pdf [Accessed October 2017].

Conrad, C., Wünsche, A., Tan, T. H., Bulkescher, J., Sieckmann, F., Verissimo, F., Edelstein, A., Walter, T., Liebel, U., Pepperkok, R. and Ellenberg, J. (2011). **Micropilot: automation of fluorescence microscopy–based imaging for systems biology**. Nature Methods *8*, 246–249, doi:10.1038/nmeth.1558.

Copmans, D., Meinl, T., Dietz, C., van Leeuwen, M., Ortmann, J., Berthold, M. R. and de Witte, P. A. (2016). **A KNIME-Based Analysis of the Zebrafish Photomotor Response Clusters the Phenotypes of 14 Classes of Neuroactive Molecules**. Journal of biomolecular screening *21*, 427–436, URL: http://journals.sagepub.com/doi/abs/10.1177/1087057115618348 [Accessed October 2017].

Cordelières, F. P. and Bolte, S. (2014). **Experimenters' guide to colocalization studies**. In: **Methods in Cell Biology**, Elsevier, pp. 395–408, doi:10.1016/B978-0-12-420138-5.00021-5.

Cornet, C., Calzolari, S., Miñana-Prieto, R., Dyballa, S., van Doornmalen, E., Rutjes, H., Savy, T., D'Amico, D. and Terriente, J. (2017). **ZeGlobalTox: An Innovative Approach to Address Organ Drug Toxicity Using Zebrafish**. International Journal of Molecular Sciences *18*, 864, doi:10.3390/ijms18040864.

Cortes, C. and Vapnik, V. (1995). **Support-vector networks**. Mach Learn *20*, 273–297, doi:10.1007/BF00994018.

Costes, S. V., Daelemans, D., Cho, E. H., Dobbin, Z., Pavlakis, G. and Lockett, S. (2004). **Automatic and Quantitative Measurement of Protein-Protein Colocalization in Live Cells**. Biophysical Journal *86*, 3993–4003, doi:10.1529/biophysj.103.038422.

Crispin, A. J. and Rankov, V. (2007). **Automated inspection of PCB components using a genetic algorithm template-matching approach**. Int J Adv Manuf Technol *35*, 293–300, doi:10.1007/s00170-006-0730-0.

Dalal, N. and Triggs, B. (2005). **Histograms of Oriented Gradients for Human Detection**. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**, 2005, IEEE, San Diego, CA, USA, pp. 886–893, doi:10.1109/CVPR.2005.177.

Davis, L. (2012). Basic Methods in Molecular Biology, Elsevier Science, URL: https://books.google.de/books?id=739enqOyW_UC.

Dembrower, K., Lindholm, P. and Strand, F. (2020). **A Multi-million Mammography Image Dataset and Population-Based Screening Cohort for the Training and Evaluation of Deep Neural Networks—the Cohort of Screen-Aged Women (CSAW)**. J Digit Imaging *33*, 408–413, doi:10.1007/s10278-019-00278-0.

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. and Zupan, B. (2013). **Orange: Data Mining Toolbox in Python**. Journal of Machine Learning Research *14*, 2349–2353, URL: http://jmlr.org/papers/v14/demsar13a.html.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). **Imagenet: A large-scale hierarchical image database**. In: **2009 IEEE conference on computer vision and pattern recognition**, 2009, Ieee, pp. 248–255.

Denk, W., Strickler, J. and Webb, W. (1990). **Two-photon laser scanning fluorescence microscopy**. Science *248*, 73–76, doi:10.1126/science.2321027.

Dhanabal, S. and Chandramathi, S. (2011). **A Review of various k-Nearest Neighbor Query Processing Techniques**. International Journal of Computer Applications *31*, 14–22.

Dietz, C. (2020). Fine-tune VGG16, URL: https://kni.me/w/EUWPBdnVuIxuFMGf [Accessed September 2020].

Domínguez, C., Heras, J. and Pascual, V. (2017). **IJ-OpenCV: Combining ImageJ and OpenCV for processing images in biomedicine**. Computers in Biology and Medicine *84*, 189–194, doi:10.1016/j.compbiomed.2017.03.027.

Dougherty, W. M. and Goodwin, P. C. (2011). *3D structured illumination microscopy*. In: Eds Enderlein, J., Gryczynski, Z. K., and Erdmann, R. 10 February 2011, p. 790512, doi:10.1117/12.877595.

Driever, W., Solnica-Krezel, L., Schier, A. F., Neuhauss, S. C., Malicki, J., Stemple, D. L., Stainier, D. Y., Zwartkruis, F., Abdelilah, S., Rangini, Z., Belak, J. and Boggs, C. (1996). **A genetic screen for mutations affecting embryogenesis in zebrafish**. Development *123*, 37, URL: http://dev.biologists.org/content/123/1/37.abstract.

Drosophila: methods and protocols (2008). Ed. Dahmann, C., Humana Press, Totowa, N.J.

Dudko, O. K. and Weiss, G. H. (2005). *Photon Diffusion in Biological Tissues*. 21.

Eberle, J. P., Muranyi, W., Erfle, H. and Gunkel, M. (2017). **Fully Automated Targeted Confocal and Single-Molecule Localization Microscopy**. In: **Super-Resolution Microscopy**, Ed. Erfle, H. Springer New York, New York, NY, pp. 139–152, doi:10.1007/978-1-4939-7265-4_12.

Edelstein, A., Amodaj, N., Hoover, K., Vale, R. and Stuurman, N. (2010). **Computer Control of Microscopes Using µManager**. Current Protocols in Molecular Biology *92*, doi:10.1002/0471142727.mb1420s92.

Eggert, U. S. (2013). **The why and how of phenotypic small-molecule screens**. Nat Chem Biol *9*, 206–209, doi:10.1038/nchembio.1206.

El-Naqa, I., Yongyi Yang, Wernick, M. N., Galatsanos, N. P. and Nishikawa, R. (2002). **Support vector machine learning for detection of microcalcifications in mammograms**. In: **Proceedings IEEE International Symposium on Biomedical Imaging**, 2002, IEEE, Washington, DC, USA, pp. 201–204, doi:10.1109/ISBI.2002.1029228.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M. and Smola, A. (2020). **AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data**. arXiv preprint arXiv:200306505.

Esner, M., Meyenhofer, F. and Bickle, M. (2018). **Live-Cell High Content Screening in Drug Development**. Methods Mol Biol *1683*, 149–164.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., and others (1996). **A density-based algorithm for discovering clusters in large spatial databases with noise.** In: **Kdd**, 1996, pp. 226–231.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. (2010). **The Pascal Visual Object Classes (VOC) Challenge**. Int J Comput Vis *88*, 303–338, doi:10.1007/s11263-009-0275-4.

Fadero, T. C. and Maddox, P. S. (2017). **Live imaging looks deeper**. eLife *6*, e30515, doi:10.7554/eLife.30515.

Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., Böhm, A., Deubner, J., Jäckel, Z., Seiwald, K., Dovzhenko, A., Tietz, O., Dal Bosco, C., Walsh, S., Saltukoglu, D., Tay, T. L., Prinz, M., Palme, K., Simons, M., Diester, I., Brox, T. and Ronneberger, O. (2018). **U-Net: deep learning for cell counting, detection, and morphometry**. Nature Methods, doi:10.1038/s41592-018-0261-2.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. and Ramanan, D. (2010). **Object Detection with Discriminatively Trained Part-Based Models**. IEEE Transactions on Pattern Analysis and Machine Intelligence *32*, 1627–1645, doi:10.1109/TPAMI.2009.167.

Fink, M., Callol-Massot, C., Chu, A., Ruiz-Lozano, P., Belmonte, J. C. I., Giles, W., Bodmer, R. and Ocorr, K. (2009). **A new method for detection and quantification of heartbeat**

parameters in Drosophila, zebrafish, and embryonic mouse hearts. BioTechniques *46*, 101–113, doi:10.2144/000113078.

Fisher, R. A. (1936). **THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS**. Annals of Eugenics *7*, 179–188, doi:10.1111/j.1469-1809.1936.tb02137.x.

Forde, J., Granger, B., Head, T., Holdgraf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B. and Willing, C. (2018). *Binder 2.0 - Reproducible, interactive, sharable environments for science at scale*. In: 2018, Austin, Texas, pp. 113–120, doi:10.25080/Majora-4af1f417-011.

Fredriksson, K., Mäkinen, V. and Navarro, G. (2007). **Rotation and lighting invariant template matching**. Information and Computation *205*, 1096–1113, doi:10.1016/j.ic.2007.03.002.

Fuad, N. M., Kaslin, J. and Wlodkowic, D. (2018). **Lab-on-a-Chip imaging micro-echocardiography (iμEC) for rapid assessment of cardiovascular activity in zebrafish larvae**. Sensors and Actuators B: Chemical *256*, 1131–1141, doi:10.1016/j.snb.2017.10.050.

Futamura, Y., Kawatani, M., Kazami, S., Tanaka, K., Muroi, M., Shimizu, T., Tomita, K., Watanabe, N. and Osada, H. (2012). **Morphobase, an Encyclopedic Cell Morphology Database, and Its Use for Drug Target Identification**. Chemistry & Biology *19*, 1620–1630, doi:10.1016/j.chembiol.2012.10.014.

Gallego, J., Pedraza, A., Lopez, S., Steiner, G., Gonzalez, L., Laurinavicius, A. and Bueno, G. (2018). **Glomerulus Classification and Detection Based on Convolutional Neural Networks**. Journal of Imaging *4*, 20, doi:10.3390/jimaging4010020.

Gallois, B. (2018). FastTrack, URL: https://github.com/FastTrackOrg/FastTrack [Accessed October 2020].

Gatys, L. A., Ecker, A. S. and Bethge, M. (2016). **Image Style Transfer Using Convolutional Neural Networks**. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, June 2016.

Gehrig, J. (2019). *3dpf zebrafish larvae, 96 well plate,Tg(wt1b:EGFP), dorsal view, ACQUIFER Imaging Machine*, doi:10.5281/zenodo.2650162.

Gehrig, J., Pandey, G. and Westhoff, J. H. (2018). **Zebrafish as a Model for Drug Screening in Genetic Kidney Diseases**. Frontiers in Pediatrics *6*, doi:10.3389/fped.2018.00183.

Gehrig, J., Reischl, M., Kalmár, É., Ferg, M., Hadzhiev, Y., Zaucker, A., Song, C., Schindler, S., Liebel, U. and Müller, F. (2009). **Automated high-throughput mapping of promoter-enhancer interactions in zebrafish embryos**. Nature Methods *6*, 911–916, doi:10.1038/nmeth.1396.

Geisler, R., Borel, N., Ferg, M., Maier, J. V. and Strähle, U. (2016). **Maintenance of Zebrafish Lines at the European Zebrafish Resource Center**. Zebrafish *13*, S-19-S-23, doi:10.1089/zeb.2015.1205.

Gierten, J. and Gehrig, J. (2019). *102 hpf medaka embryos in 96 well plate (4 embryo/well) - brightfield - 2X magnification - ACQUIFER Imaging Machine*, doi:10.5281/zenodo.2650147.

Gierten, J., Pylatiuk, C., Hammouda, O. T., Schock, C., Stegmaier, J., Wittbrodt, J., Gehrig, J. and Loosli, F. (2020). **Automated high-throughput heartbeat quantification in medaka and zebrafish embryos under physiological conditions**. Sci Rep *10*, 2046, doi:10.1038/s41598-020-58563-w.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2016). **Region-Based Convolutional Networks for Accurate Object Detection and Segmentation**. IEEE Trans Pattern Anal Mach Intell *38*, 142–158, doi:10.1109/TPAMI.2015.2437384.

GluonCV Contributors (2020a). **Prepare custom datasets for object detection**, *GluonCV documentation*, URL: https://cv.gluon.ai/build/examples_datasets/detection_custom.html [Accessed October 2020].

GluonCV Contributors (2020b). **Prepare custom datasets for object detection - Pascal VOC format**, *GluonCV documentation*, URL: https://cv.gluon.ai/build/examples_datasets/detection_custom.html#derive-from-pascal-voc-format [Accessed October 2020].

GluonCV Contributors (n.d.). **Finetune a pretrained detection model**, *GluonCV documentation*, URL: https://gluon-cv.mxnet.io/build/examples_detection/finetune_detection.html [Accessed August 2020a].

GluonCV Contributors (n.d.). **GluonCV model zoo**, *GluonCV documentation*, URL: https://cv.gluon.ai/model_zoo/detection.html#gluoncv-model-zoo-detection [Accessed October 2020b].

Godinez, W. J., Hossain, I., Lazic, S. E., Davies, J. W. and Zhang, X. (2017). **A multi-scale convolutional neural network for phenotyping high-content cellular images**. Bioinformatics *33*, 2010–2019.

Grigorescu, S., Trasnea, B., Cocias, T. and Macesanu, G. (2020). **A survey of deep learning techniques for autonomous driving**. J Field Robotics *37*, 362–386, doi:10.1002/rob.21918.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. and Chen, T. (2017). **Recent advances in convolutional neural networks**. Pattern Recognition, doi:10.1016/j.patcog.2017.10.013.

Gunkel, M., Eberle, J. P. and Erfle, H. (2017). **Fluorescence-Based High-Throughput and Targeted Image Acquisition and Analysis for Phenotypic Screening**. In: **Light Microscopy**, Eds Markaki, Y. and Harz, H. Springer New York, New York, NY, pp. 269–280, doi:10.1007/978-1-4939-6810-7_17.

Gunnarsson, L., Jauhiainen, A., Kristiansson, E., Nerman, O. and Larsson, D. G. J. (2008). **Evolutionary Conservation of Human Drug Targets in Organisms used for Environmental Risk Assessments**. Environ Sci Technol *42*, 5807–5813, doi:10.1021/es8005173.

Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., and others (2020). **GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing.** Journal of Machine Learning Research *21*, 1–7.

Gustafsson, M. G. L. (2000). **Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy. SHORT COMMUNICATION**. J Microsc *198*, 82–87, doi:10.1046/j.1365-2818.2000.00710.x.

Han, J. W., Breckon, T. P., Randell, D. A. and Landini, G. (2012). **The application of support vector machine classification to detect cell nuclei for automated microscopy**. Machine Vision and Applications *23*, 15–24, doi:10.1007/s00138-010-0275-y.

Han, J. W., Breckon, T., Randell, D. and Landini, G. (2008). *Radicular cysts and odontogenic keratocysts epithelia classification using cascaded Haar classifiers*. 5.

Hanke, N., King, B. L., Vaske, B., Haller, H. and Schiffer, M. (2015). **A fluorescence-based assay for proteinuria screening in larval zebrafish (Danio rerio)**. Zebrafish *12*, 372–376.

Hanke, N., Staggs, L., Schroder, P., Litteral, J., Fleig, S., Kaufeld, J., Pauli, C., Haller, H. and Schiffer, M. (2013). **'Zebrafishing' for Novel Genes Relevant to the Glomerular Filtration Barrier**. BioMed Research International *2013*, 1–12, doi:10.1155/2013/658270.

Haralick, R. M. and Shanmugam, K. (1973). **Textural features for image classification**. IEEE Transactions on systems, man, and cybernetics 610–621.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. (2020). **Array programming with NumPy**. Nature *585*, 357–362, doi:10.1038/s41586-020-2649-2.

Harrison, M. A. and Rae, I. F. (1997). General Techniques of Cell Culture, Cambridge University Press, URL: https://books.google.de/books?id=P_-Mkis9lo0C.

He, K., Gkioxari, G., Dollar, P. and Girshick, R. (2017). **Mask R-CNN**. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, October 2017.

Hedrich, H. J. and Bullock, G. R. (2004). The laboratory mouse, Elsevier Academic Press, London; San Diego, URL: http://www.myilibrary.com?id=111185 [Accessed October 2020].

Heilbronner, R. P. (1992). **The autocorrelation function: an image processing tool for fabric analysis**. Tectonophysics *212*, 351–370, doi:10.1016/0040-1951(92)90300-U.

Hell, S. W. and Wichmann, J. (1994). **Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy**. Opt Lett *19*, 780–782.

Hentschel, D. M., Mengel, M., Boehme, L., Liebsch, F., Albertin, C., Bonventre, J. V., Haller, H. and Schiffer, M. (2007). **Rapid screening of glomerular slit diaphragm integrity in larval zebrafish**. American Journal of Physiology-Renal Physiology *293*, F1746–F1750, doi:10.1152/ajprenal.00009.2007.

Ho, T. K. (1995). **Random decision forests**. In: **Proceedings of 3rd international conference on document analysis and recognition**, 1995, IEEE, pp. 278–282.

Hocking, J. C., Distel, M. and Köster, R. W. (2013). **Studying cellular and subcellular dynamics in the developing zebrafish nervous system**. Experimental Neurology *242*, 1–10, doi:10.1016/j.expneurol.2012.03.009.

Hostetter, C. L., Sullivan-Brown, J. L. and Burdine, R. D. (2003). **Zebrafish pronephros: a model for understanding cystic kidney disease**. Dev Dyn *228*, 514–522.

Howe, K., Clark, M. D., Torroja, C. F., Torrance, J., Berthelot, C., Muffato, M., Collins, J. E., Humphray, S., McLaren, K., Matthews, L., McLaren, S., Sealy, I., Caccamo, M., Churcher, C., Scott, C., Barrett, J. C., Koch, R., Rauch, G.-J., White, S., Chow, W., Kilian, B., Quintais, L. T., Guerra-Assunção, J. A., Zhou, Y., Gu, Y., Yen, J., Vogel, J.-H., Eyre, T., Redmond, S., Banerjee, R., Chi, J., Fu, B., Langley, E., Maguire, S. F., Laird, G. K., Lloyd, D., Kenyon, E., Donaldson, S., Sehra, H., Almeida-King, J., Loveland, J., Trevanion, S., Jones, M., Quail, M., Willey, D., Hunt, A., Burton, J., Sims, S., McLay, K., Plumb, B., Davis, J., Clee, C., Oliver, K., Clark, R., Riddle, C., Elliott, D., Threadgold, G., Harden, G., Ware, D., Begum, S., Mortimore, B., Kerry, G., Heath, P., Phillimore, B., Tracey, A., Corby, N., Dunn, M., Johnson, C., Wood, J., Clark, S., Pelan, S., Griffiths, G., Smith, M., Glithero, R., Howden, P., Barker, N., Lloyd, C., Stevens, C., Harley, J., Holt, K., Panagiotidis, G., Lovell, J., Beasley, H., Henderson, C., Gordon, D., Auger, K., Wright, D., Collins, J., Raisen, C., Dyer, L., Leung, K., Robertson, L., Ambridge, K., Leongamornlert, D., McGuire, S., Gilderthorp, R., Griffiths, C., Manthravadi, D., Nichol, S., Barker, G., Whitehead, S., Kay, M., Brown, J., Murnane, C., Gray, E., Humphries, M., Sycamore, N., Barker, D., Saunders, D., Wallis, J., Babbage, A., Hammond, S., Mashreghi-Mohammadi, M., Barr, L., Martin, S., Wray, P., Ellington, A., Matthews, N., Ellwood, M., Woodmansey, R., Clark, G., Cooper, J. D., Tromans, A., Grafham, D., Skuce, C., Pandian, R., Andrews, R., Harrison, E., Kimberley, A., Garnett, J., Fosker, N., Hall, R., Garner, P., Kelly, D., Bird, C., Palmer, S., Gehring, I., Berger, A., Dooley, C. M., Ersan-Ürün, Z., Eser, C., Geiger, H., Geisler, M., Karotki, L., Kirn, A., Konantz, J., Konantz, M., Oberländer, M., Rudolph-Geiger, S., Teucke, M., Lanz, C., Raddatz, G., Osoegawa, K., Zhu, B., Rapp, A., Widaa, S., Langford, C., Yang, F., Schuster, S. C., Carter, N. P., Harrow, J., Ning, Z., Herrero, J., Searle, S. M. J., Enright, A., Geisler, R., Plasterk, R. H. A., Lee, C., Westerfield, M., de Jong, P. J., Zon, L. I., Postlethwait, J. H., Nüsslein-Volhard, C., Hubbard, T. J. P., Crollius, H. R., Rogers, J.

and Stemple, D. L. (2013). **The zebrafish reference genome sequence and its relationship to the human genome**. Nature *496*, 498–503, doi:10.1038/nature12111.

Huisken, J. (2004). **Optical Sectioning Deep Inside Live Embryos by Selective Plane Illumination Microscopy**. Science *305*, 1007–1009, doi:10.1126/science.1100035.

ImageJ contributors (2005). ImageJA GitHub repository, URL: https://github.com/imagej/ImageJA [Accessed October 2020].

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). **Data clustering: a review**. ACM Comput Surv *31*, 264–323, doi:10.1145/331499.331504.

Jang, Z.-H., Chung, H.-C., Ahn, Y. G., Kwon, Y.-K., Kim, J.-S., Ryu, J.-H., Ryu, D. H., Kim, C.-H. and Hwang, G.-S. (2012). **Metabolic profiling of an alcoholic fatty liver in zebrafish (Danio rerio)**. Mol BioSyst *8*, 2001–2009, doi:10.1039/C2MB25073J.

Jenkins, J. L. and Urban, L. (2010). **Fishing for neuroactive compounds**. Nat Chem Biol *6*, 172–173, doi:10.1038/nchembio.320.

Jiang, H., Li, S., Liu, W., Zheng, H., Liu, J. and Zhang, Y. (2020). **Geometry-Aware Cell Detection with Deep Learning** Ed. Gilbert, J. A. mSystems *5*, e00840-19, /msystems/5/1/msys.00840-19.atom, doi:10.1128/mSystems.00840-19.

Karlsson, J., von Hofsten, J. and Olsson, P. E. (2001). **Generating transparent zebrafish: a refined method to improve detection of gene expression during embryonic development**. Mar Biotechnol (NY) *3*, 522–527.

Kato, S., Nakagawa, T., Ohkawa, M., Muramoto, K., Oyama, O., Watanabe, A., Nakashima, H., Nemoto, T. and Sugitani, K. (2004). **A computer image processing system for quantification of zebrafish behavior**. Journal of Neuroscience Methods *134*, 1–7, doi:10.1016/j.jneumeth.2003.09.028.

Kaufman, C. K., White, R. M. and Zon, L. (2009). **Chemical genetic screening in the zebrafish embryo**. Nature protocols *4*, 1422, doi:10.1038/nprot.2009.144.

Khotanzad, A. and Hong, Y. H. (1990). **Invariant image recognition by Zernike moments**. IEEE Transactions on Pattern Analysis and Machine Intelligence *12*, 489–497.

Kim, H. Y. and De Araújo, S. A. (2007). **Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast**. In: **Pacific-Rim Symposium on Image and Video Technology**, 2007, Springer, pp. 100–113.

Kim, J., Koo, B.-K. and Knoblich, J. A. (2020). **Human organoids: model systems for human biology and medicine**. Nat Rev Mol Cell Biol *21*, 571–584, doi:10.1038/s41580-020-0259-3.

Kimmel, C. B., Ballard, W. W., Kimmel, S. R., Ullmann, B. and Schilling, T. F. (1995). **Stages of embryonic development of the zebrafish**. Dev Dyn *203*, 253–310, doi:10.1002/aja.1002030302.

Kirk, R. G. W. (2018). **Recovering *The Principles of Humane Experimental Technique*: The 3Rs and the Human Essence of Animal Research**. Science, Technology, & Human Values *43*, 622–648, doi:10.1177/0162243917726579.

KNIME (n.d.). **Setup the python integration**, *KNIME 4.2 documentation*, URL: https://docs.knime.com/latest/python_installation_guide/index.html#setup_python_integration [Accessed October 2020].

Kokel, D., Bryan, J., Laggner, C., White, R., Cheung, C. Y. J., Mateus, R., Healey, D., Kim, S., Werdich, A. A., Haggarty, S. J., MacRae, C. A., Shoichet, B. and Peterson, R. T. (2010). **Rapid behavior-based identification of neuroactive small molecules in the zebrafish**. Nature Chemical Biology *6*, 231–237, doi:10.1038/nchembio.307.

Koster, R. and Sassen, W. A. (2015). **A molecular toolbox for genetic manipulation of zebrafish**. AGG 151, doi:10.2147/AGG.S57585.

Kotsiantis, S. B., Zaharakis, I. and Pintelas, P. (2007). **Supervised machine learning: A review of classification techniques**. Emerging artificial intelligence applications in computer engineering *160*, 3–24.

Krull, A., Buchholz, T.-O. and Jug, F. (2019). **Noise2void-learning denoising from single noisy images**. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2019, pp. 2129–2137.

Kumar, D., Wong, A. and Clausi, D. A. (2015). **Lung Nodule Classification Using Deep Features in CT Images**. In: **2015 12th Conference on Computer and Robot Vision**, June 2015, IEEE, Halifax, NS, Canada, pp. 133–138, doi:10.1109/CRV.2015.25.

Kuo, F. Y. and Sloan, I. H. (2005). *Lifting the Curse of Dimensionality*. *52*, 9.

Lagache, T., Sauvonnet, N., Danglot, L. and Olivo-Marin, J.-C. (2015). **Statistical analysis of molecule colocalization in bioimaging: Statistical Analysis of Molecule Colocalization in Bioimaging**. Cytometry *87*, 568–579, doi:10.1002/cyto.a.22629.

Lakhmi Prasanna, P., Rajeswara Rao, D., Meghana, Y., Maithri, K. and Dhinesh, T. (2017). **Analysis of supervised classification techniques**. IJET *7*, 283, doi:10.14419/ijet.v7i1.1.9486.

Lam, S. H. and Gong, Z. (2006). **Modeling Liver Cancer Using Zebrafish: A Comparative Oncogenomics Approach**. Cell Cycle *5*, 573–577, doi:10.4161/cc.5.6.2550.

Langenau, D. M., Traver, D., Ferrando, A. A., Kutok, J. L., Aster, J. C., Kanki, J. P., Lin, S., Prochownik, E., Trede, N. S., Zon, L. I. and Look, A. T. (2003). **Myc-induced T cell leukemia in transgenic zebrafish**. Science *299*, 887–890.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). **Gradient-based learning applied to document recognition**. Proc IEEE *86*, 2278–2324, doi:10.1109/5.726791.

Lecun, Yann, Cortes, C. and Burges, C. J. C. (1998). *The MNIST database of handwritten digits*, URL: http://yann.lecun.com/exdb/mnist/ [Accessed October 2020].

Legland, D., Arganda-Carreras, I. and Andrey, P. (2016). **MorphoLibJ: integrated library and plugins for mathematical morphology with ImageJ**. Bioinformatics btw413, doi:10.1093/bioinformatics/btw413.

Lehmann, R., Lee, C. M., Shugart, E. C., Benedetti, M., Charo, R. A., Gartner, Z., Hogan, B., Knoblich, J., Nelson, C. M. and Wilson, K. M. (2019). **Human organoids: a new dimension in cell biology** Ed. Drubin, D. G. MBoC *30*, 1129–1137, doi:10.1091/mbc.E19-03-0135.

Leifer, A. M., Fang-Yen, C., Gershow, M., Alkema, M. J. and Samuel, A. D. T. (2011). **Optogenetic manipulation of neural activity in freely moving Caenorhabditis elegans**. Nat Methods *8*, 147–152, doi:10.1038/nmeth.1554.

Letamendia, A., Quevedo, C., Ibarbia, I., Virto, J. M., Holgado, O., Diez, M., Izpisua Belmonte, J. C. and Callol-Massot, C. (2012). **Development and Validation of an Automated High-Throughput System for Zebrafish In Vivo Screenings** Ed. Nebert, D. PLoS ONE *7*, e36690, doi:10.1371/journal.pone.0036690.

Li, H., Soto-Montoya, H., Voisin, M., Valenzuela, L. F. and Prakash, M. (2019). Octopi: Open configurable high-throughput imaging platform for infectious disease diagnosis in the field, Bioengineering, doi:10.1101/684423.

Liao, S., Zhu, X., Lei, Z., Zhang, L. and Li, S. Z. (2007). **Learning Multi-scale Block Local Binary Patterns for Face Recognition**. In: **Advances in Biometrics**, Eds Lee, S.-W. and Li, S. Z. 2007, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 828–837.

Liaw, A. and Wiener, M. (2002). *Classification and Regression by randomForest*. *2*, 6.

Lieschke, G. J. and Currie, P. D. (2007). **Animal models of human disease: zebrafish swim into view**. Nat Rev Genet *8*, 353–367.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2018). **Focal Loss for Dense Object Detection**. arXiv:170802002 [cs], URL: http://arxiv.org/abs/1708.02002 [Accessed October 2020].

Lippmann, R. P. (1989). **Pattern classification using neural networks**. IEEE Communications Magazine *27*, 47–50.

Lischik, C. Q., Adelmann, L. and Wittbrodt, J. (2019). **Enhanced in vivo-imaging in medaka by optimized anaesthesia, fluorescent protein selection and removal of pigmentation** Ed. Winkler, C. PLoS ONE *14*, e0212956, doi:10.1371/journal.pone.0212956.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016). **SSD: Single Shot MultiBox Detector**. arXiv:151202325 [cs] *9905*, 21–37, doi:10.1007/978-3-319-46448-0_2.

Ljosa, V., Caie, P. D., ter Horst, R., Sokolnicki, K. L., Jenkins, E. L., Daya, S., Roberts, M. E., Jones, T. R., Singh, S., Genovesio, A., Clemons, P. A., Carragher, N. O. and Carpenter, A. E. (2013). **Comparison of Methods for Image-Based Profiling of Cellular**

Morphological Responses to Small-Molecule Treatment. J Biomol Screen *18*, 1321–1329, doi:10.1177/1087057113503553.

Ljosa, V. and Carpenter, A. E. (2009). **Introduction to the quantitative analysis of two-dimensional fluorescence microscopy images for cell-based screening**. PLoS computational biology *5*, e1000603.

Loddo, A., Di Ruberto, C., Kocher, M. and Prod'Hom, G. (2019). **MP-IDB: The Malaria Parasite Image Database for Image Processing and Analysis**. In: **Processing and Analysis of Biomedical Information**, Eds Lepore, N., Brieva, J., Romero, E., Racoceanu, D., and Joskowicz, L. Springer International Publishing, Cham, pp. 57–65, doi:10.1007/978-3-030-13835-6_7.

Loo, L.-H., Wu, L. F. and Altschuler, S. J. (2007). **Image-based multivariate profiling of drug responses from single cells**. Nat Methods *4*, 445–453, doi:10.1038/nmeth1032.

Lukonin, I., Serra, D., Challet Meylan, L., Volkmann, K., Baaten, J., Zhao, R., Meeusen, S., Colman, K., Maurer, F., Stadler, M. B., Jenkins, J. and Liberali, P. (2020). **Phenotypic landscape of intestinal organoid regeneration**. Nature *586*, 275–280, doi:10.1038/s41586-020-2776-9.

Luo, J. Y., Irisson, J.-O., Graham, B., Guigand, C., Sarafraz, A., Mader, C. and Cowen, R. K. (2018). **Automated plankton image analysis using convolutional neural networks: Automated plankton image analysis using CNNs**. Limnol Oceanogr Methods *16*, 814–827, doi:10.1002/lom3.10285.

Lupi, E., Mercader, N. and Gehrig, J. (2017). **High Throughtput Screening Assay for the identification of compound enhancing the Epicardium formation**, *4D Heart - Horizon 2020 research project*, URL: http://www.4dheart.eu/wp-content/uploads/2018/12/Eleonora-Lupi.pdf.

Maaten, L. van der and Hinton, G. (2008). **Visualizing data using t-SNE**. Journal of machine learning research *9*, 2579–2605.

MacRae, C. A. and Peterson, R. T. (2015). **Zebrafish as tools for drug discovery**. Nature Reviews Drug Discovery *14*, 721–731, doi:10.1038/nrd4627.

Mandrell, D., Truong, L., Jephson, C., Sarker, M. R., Moore, A., Lang, C., Simonich, M. T. and Tanguay, R. L. (2012). **Automated Zebrafish Chorion Removal and Single Embryo Placement: Optimizing Throughput of Zebrafish Developmental Toxicity Screens**. Journal of Laboratory Automation *17*, 66–74, doi:10.1177/2211068211432197.

Marcato, D., Alshut, R., Breitwieser, H., Mikut, R., Strahle, U., Pylatiuk, C. and Peravali, R. (2015). *An automated and high-throughput Photomotor Response platform for chemical screens*. In: August 2015, IEEE, pp. 7728–7731, doi:10.1109/EMBC.2015.7320183.

Marimon, D. and Ebrahimi, T. (2007). **Efficient rotation-discriminative template matching**. Progress in Pattern Recognition, Image Analysis and Applications 221–230.

Marques, I. J., Lupi, E. and Mercader, N. (2019). **Model systems for regeneration: zebrafish**. Development *146*, dev167692, doi:10.1242/dev.167692.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, URL: https://www.tensorflow.org/.

Masud, S., Torraca, V. and Meijer, A. H. (2017). **Modeling Infectious Diseases in the Context of a Developing Immune System**. Curr Top Dev Biol *124*, 277–329.

Maulana Budiman, R. A., Achmad, B., Faridah, Arif, A., Nopriadi and Zharif, L. (2016). **Localization of white blood cell images using Haar Cascade classifiers**. In: **2016 1st International Conference on Biomedical Engineering (IBIOMED)**, October 2016, IEEE, Yogyakarta, Indonesia, pp. 1–5, doi:10.1109/IBIOMED.2016.7869822.

McInnes, L. (2018). **Basic UMAP parameters**, *UMAP readthedoc Documentation*.

McInnes, L., Healy, J. and Melville, J. (2020). **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction**. arXiv:180203426 [cs, stat], URL: http://arxiv.org/abs/1802.03426 [Accessed October 2020].

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. In: 2010, Austin, Texas, pp. 56–61, doi:10.25080/Majora-92bf1922-00a.

Meiniel, W., Olivo-Marin, J.-C. and Angelini, E. D. (2018). **Denoising of Microscopy Images: A Review of the State-of-the-Art, and a New Sparsity-Based Method**. IEEE Trans on Image Process *27*, 3842–3856, doi:10.1109/TIP.2018.2819821.

Meyvantsson, I. and Beebe, D. J. (2008). **Cell Culture Models in Microfluidic Systems**. Annual Rev Anal Chem *1*, 423–449, doi:10.1146/annurev.anchem.1.031207.113042.

Mikut, R., Dickmeis, T., Driever, W., Geurts, P., Hamprecht, F. A., Kausler, B. X., Ledesma-Carbayo, M. J., Marée, R., Mikula, K., Pantazis, P., Ronneberger, O., Santos, A., Stotzka, R., Strähle, U. and Peyriéras, N. (2013). **Automated Processing of Zebrafish Imaging Data: A Survey**. Zebrafish *10*, 401–421, doi:10.1089/zeb.2013.0886.

Ming-Kuei Hu (1962). **Visual pattern recognition by moment invariants**. IEEE Transactions on Information Theory *8*, 179–187, doi:10.1109/TIT.1962.1057692.

Minsky, M. (1988). **Memoir on inventing the confocal scanning microscope**. Scanning *10*, 128–138, doi:10.1002/sca.4950100403.

Mi-Sun Kang, Hye-Ryun Kim, Sudong Kim, Gyu Ha Ryu, and Myoung-Hee Kim (2016). *Analysis of cancer cell morphology in fluorescence microscopy image exploiting shape descriptor*. In: 6 April 2016, doi:10.1117/12.2213829.

Mobahi, H., Collobert, R. and Weston, J. (2009). **Deep learning from temporal coherence in video**. In: **Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09**, 2009, ACM Press, Montreal, Quebec, Canada, pp. 1–8, doi:10.1145/1553374.1553469.

Model, M. A. and Burkhardt, J. K. (2001). **A standard for calibration and shading correction of a fluorescence microscope**. Cytometry *44*, 309–316, doi:10.1002/1097-0320(20010801)44:4<309::AID-CYTO1122>3.0.CO;2-3.

Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M. and Van Valen, D. (2019). **Deep learning for cellular image analysis**. Nat Methods *16*, 1233–1246, doi:10.1038/s41592-019-0403-1.

Moses and Olafenwa, J. (2018). ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities, URL: https://github.com/OlafenwaMoses/ImageAI.

Müller, A. C. and Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists, First edition., O'Reilly Media, Inc, Sebastopol, CA.

Nagpal, S., Singh, M., Singh, R. and Vatsa, M. (2019). **Deep Learning for Face Recognition: Pride or Prejudiced?** arXiv:190401219 [cs], URL: http://arxiv.org/abs/1904.01219 [Accessed October 2020].

Neumann, B., Held, M., Liebel, U., Erfle, H., Rogers, P., Pepperkok, R. and Ellenberg, J. (2006). **High-throughput RNAi screening by time-lapse imaging of live human cells**. Nat Methods *3*, 385–390, doi:10.1038/nmeth876.

Neumann, B., Walter, T., Hériché, J.-K., Bulkescher, J., Erfle, H., Conrad, C., Rogers, P., Poser, I., Held, M., Liebel, U., Cetin, C., Sieckmann, F., Pau, G., Kabbe, R., Wünsche, A., Satagopam, V., Schmitz, M. H. A., Chapuis, C., Gerlich, D. W., Schneider, R., Eils, R., Huber, W., Peters, J.-M., Hyman, A. A., Durbin, R., Pepperkok, R. and Ellenberg, J. (2010). **Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes**. Nature *464*, 721–727, doi:10.1038/nature08869.

Newell, A., Yang, K. and Deng, J. (2016). **Stacked Hourglass Networks for Human Pose Estimation**. arXiv:160306937 [cs], URL: http://arxiv.org/abs/1603.06937 [Accessed October 2020].

NG, A. (2018). Non-max Suppression video tutorial - Coursera. 2018, URL: https://www.coursera.org/lecture/convolutional-neural-networks/non-max-suppression-dvrjH [Accessed July 2020].

Nguyen, C. T., Lu, Q., Wang, Y. and Chen, J.-N. (2008). **Zebrafish as a model for cardiovascular development and disease**. Drug Discov Today Dis Models *5*, 135–140.

Noble, W. S. (2006). **What is a support vector machine?** Nat Biotechnol *24*, 1565–1567, doi:10.1038/nbt1206-1565.

Nüsslein-Volhard, C. (2012). **The zebrafish issue of <em>Development</em>**. Development *139*, 4099, doi:10.1242/dev.085217.

Nusslein-Volhard, C. and Dahm, R. (2002). Zebrafish, Oxford University Press.

Oke, S., Ookado, M. and Nakamura, Y. (1995). **Recognition of Fruits by Image Processing - Application of Template Matching-**. IFAC Proceedings Volumes *28*, 129–134, doi:10.1016/S1474-6670(17)47172-2.

Olaode, A., Naghdy, G. and Todd, C. (2014). *Unsupervised Classification of Images: A Review*. 19.

OpenCV contributors (2000). Open Source Computer Vision Library, URL: https://opencv.org/ [Accessed July 2020].

OpenCV contributors (2018). *Training of a Haar-cascade detector with OpenCV*, URL: https://docs.opencv.org/3.4.2/dc/d88/tutorial_traincascade.html [Accessed July 2020].

OpenCV team (2020). OpenCV releases, URL: https://opencv.org/releases/ [Accessed October 2017].

Ou, G. and Murphey, Y. L. (2007). **Multi-class pattern classification using neural networks**. Pattern Recognition *40*, 4–18, doi:10.1016/j.patcog.2006.04.041.

Ovchinnikova, S. and Anders, S. (2019). Exploring dimension-reduced embeddings with Sleepwalk, Bioinformatics, doi:10.1101/603589.

Pandey, G., Gehrig, J. and Thomas, L. (2020). *Fluorescently-labelled zebrafish pronephroi + ground truth classes (normal/cystic) + trained CNN model*, doi:10.5281/ZENODO.3997728.

Pandey, G., Westhoff, J., Schaefer, F. and Gehrig, J. (2019). **A Smart Imaging Workflow for Organ-Specific Screening in a Cystic Kidney Zebrafish Disease Model**. International Journal of Molecular Sciences *20*, 1290, doi:10.3390/ijms20061290.

Pantanowitz, L., Evans, A., Pfeifer, J., Collins, L., Valenstein, P., Kaplan, K., Wilbur, D. and Colgan, T. (2011). **Review of the current state of whole slide imaging in pathology**. J Pathol Inform *2*, 36, doi:10.4103/2153-3539.83746.

Papageorgiou, C. P., Oren, M. and Poggio, T. (1998). **A general framework for object detection**. In: **Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)**, 1998, Narosa Publishing House, Bombay, India, pp. 555–562, doi:10.1109/ICCV.1998.710772.

Pardo-Martin, C., Chang, T.-Y., Koo, B. K., Gilleland, C. L., Wasserman, S. C. and Yanik, M. F. (2010). **High-throughput in vivo vertebrate screening**. Nat Methods *7*, 634–636, doi:10.1038/nmeth.1481.

Pawlowski, N., Caicedo, J. C., Singh, S., Carpenter, A. E. and Storkey, A. (2016). Automating Morphological Profiling with Generic Deep Convolutional Networks, Bioinformatics, doi:10.1101/085118.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). **Scikit-learn: Machine Learning in Python**. Journal of Machine Learning Research *12*, 2825–2830.

Peravali, R., Gehrig, J., Giselbrecht, S., Lütjohann, D. S., Hadzhiev, Y., Müller, F. and Liebel, U. (2011). **Automated feature detection and imaging for high-resolution screening of zebrafish embryos**. BioTechniques *50*, 319–324, doi:10.2144/000113669.

Perlman, Z. E. (2004). **Multidimensional Drug Profiling By Automated Microscopy**. Science *306*, 1194–1198, doi:10.1126/science.1100709.

Peterson, R. T. and Fishman, M. C. (2011). **Designing Zebrafish Chemical Screens**. In: **Methods in Cell Biology**, Elsevier, pp. 525–541, doi:10.1016/B978-0-12-381320-6.00023-0.

Peterson, R. T., Link, B. A., Dowling, J. E. and Schreiber, S. L. (2000). **Small molecule developmental screens reveal the logic and timing of vertebrate development**. Proceedings of the national academy of sciences *97*, 12965–12969.

Petrie, T. A., Strand, N. S., Yang, C.-T., Rabinowitz, J. S. and Moon, R. T. (2014). *Macrophages modulate adult zebrafish tail fin regeneration*. 12.

Petsiuk, A. L. and Pearce, J. M. (2020). **Open source computer vision-based layer-wise 3D printing analysis**. Additive Manufacturing *36*, 101473, doi:10.1016/j.addma.2020.101473.

Pirvu, A. (2019). Perceptron, URL: https://www.codeproject.com/Articles/5161043/From-Machine-Learning-to-Machine-Cognition.

Popova, A. A., Marcato, D., Peravali, R., Wehl, I., Schepers, U. and Levkin, P. A. (2018). **Fish-Microarray: A Miniaturized Platform for Single-Embryo High-Throughput Screenings**. Advanced Functional Materials *28*, 1703486, doi:10.1002/adfm.201703486.

Pylatiuk, C., Sanchez, D., Mikut, R., Alshut, R., Reischl, M., Hirth, S., Rottbauer, W. and Just, S. (2014). **Automatic Zebrafish Heartbeat Detection and Analysis for Zebrafish Embryos**. Zebrafish *11*, 379–383, doi:10.1089/zeb.2014.1002.

PYME contributors (2020). PYME: The PYthon Microscopy Environment, URL: https://www.python-microscopy.org/ [Accessed October 2020].

Quwailid, M. M., Hugill, A., Dear, N., Vizor, L., Wells, S., Horner, E., Fuller, S., Weedon, J., McMath, H., Woodman, P., Edwards, D., Campbell, D., Rodger, S., Carey, J., Roberts, A., Glenister, P., Lalanne, Z., Parkinson, N., Coghill, E. L., McKeone, R., Cox, S., Willan, J., Greenfield, A., Keays, D., Brady, S., Spurr, N., Gray, I., Hunter, J., Brown, S. D. M. and Cox, R. D. (2004). **A gene-driven ENU-based approach to generating an allelic series in any gene**. Mamm Genome *15*, 585–591, doi:10.1007/s00335-004-2379-z.

Rabut, G. and Ellenberg, J. (2004). **Automatic real-time three-dimensional cell tracking by fluorescence microscopy**. J Microsc *216*, 131–137, doi:10.1111/j.0022-2720.2004.01404.x.

Ramyachitra, D. D. and Manikandan, P. (2014). **Imbalanced dataset classification and solutions: a review**. International Journal of Computing and Business Research *5*, 29.

Rasband, W. and Ferreira, T. (2012). ImageJ Tools, URL: https://imagej.nih.gov/ij/docs/guide/146-19.html#toc-Section-19 [Accessed October 2020].

Rasband, W. and ImageJ contributors (n.d.). **API overview**, *Image API documentation*, URL: https://imagej.nih.gov/ij/developer/api/overview-summary.html [Accessed October 2020].

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). **You Only Look Once: Unified, Real-Time Object Detection**. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, June 2016, IEEE, Las Vegas, NV, USA, pp. 779–788, doi:10.1109/CVPR.2016.91.

Rennekamp, A. J. and Peterson, R. T. (2015). **15 years of zebrafish chemical screening**. Current Opinion in Chemical Biology *24*, 58–70, doi:10.1016/j.cbpa.2014.10.025.

Renshaw, S. A., Loynes, C. A., Trushell, D. M. I., Elworthy, S., Ingham, P. W. and Whyte, M. K. B. (2006). **A transgenic zebrafish model of neutrophilic inflammation**. Blood *108*, 3976–3978, doi:10.1182/blood-2006-05-024075.

Rho, V., Yi, A., Channavajjala, B., McPhillips, L., Nathan, S. W., Focht, R., Ohene, N., Adorno, W., Durieux, M. and Brown, D. (2020). **Digitization of Perioperative Surgical Flowsheets**. In: **2020 Systems and Information Engineering Design Symposium (SIEDS)**, April 2020, IEEE, Charlottesville, VA, USA, pp. 1–6, doi:10.1109/SIEDS49339.2020.9106679.

Rhode, D. S. (2015). *ZEN Open Application Development (OAD) Tools for Smart Microscopy*. 26, URL: http://eubias.org/eubias2015/wp-content/uploads/2015/01/ZEN.pdf [Accessed October 2020].

Richter, S., Schulze, U., Tomançak, P. and Oates, A. C. (2017). **Small molecule screen in embryonic zebrafish using modular variations to target segmentation**. Nat Commun *8*, 1901, doi:10.1038/s41467-017-01469-5.

Rieger, S., Wang, F. and Sagasti, A. (2011). **Time-lapse imaging of neural development: Zebrafish lead the way into the fourth dimension**. Genesis *49*, 534–545, doi:10.1002/dvg.20729.

Rines, D. R., Gomez-Ferreria, M., Zhou, Y., DeJesus, P., Grob, S., Batalov, S., Labow, M., Huesken, D., Mickanin, C., Hall, J., Reinhardt, M., Natt, F., Lange, J., Sharp, D. J., Chanda, S. K. and Caldwell, J. S. (2008). **Whole genome functional analysis identifies novel components required for mitotic spindle integrity in human cells**. Genome Biol *9*, R44, doi:10.1186/gb-2008-9-2-r44.

Rocca, B. (2019). **Handling imbalanced datasets in machine learning**, *Towards data science*, URL: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28 [Accessed November 2020].

Ronneberger, O., Fischer, P. and Brox, T. (2015). **U-Net: Convolutional Networks for Biomedical Image Segmentation**. In: **Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015**, Eds Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F. Springer International Publishing, Cham, pp. 234–241, doi:10.1007/978-3-319-24574-4_28.

Rosebrock, A. (2014a). **Fine-tuning with Keras and Deep-Learning**, *PyImageSearch*, URL: https://www.pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/.

Rosebrock, A. (2014b). **Non-Maximum Suppression for Object Detection in Python**, *PyImageSearch*, URL: https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/ [Accessed July 2020].

Rothe, R., Guillaumin, M. and Van Gool, L. (2015). **Non-maximum Suppression for Object Detection by Passing Messages Between Windows**. In: **Computer Vision -- ACCV 2014**, Eds Cremers, D., Reid, I., Saito, H., and Yang, M.-H. Springer International Publishing, Cham, pp. 290–306, doi:10.1007/978-3-319-16865-4_19.

Rowley, H. A., Baluja, S. and Kanade, T. (1998). **Neural network-based face detection**. IEEE Trans Pattern Anal Machine Intell *20*, 23–38, doi:10.1109/34.655647.

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

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2015). **ImageNet Large Scale Visual Recognition Challenge**. International Journal of Computer Vision (IJCV) *115*, 211–252.

Russell, W. M. S. (1995). **The development of the three Rs concept.** Altern Lab Anim *23*, 298–304.

Rust, M. J., Bates, M. and Zhuang, X. (2006). **Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)**. Nature Methods *3*, 793–796, doi:10.1038/nmeth929.

Safavian, S. R. and Landgrebe, D. (1991). **A survey of decision tree classifier methodology**. IEEE Transactions on Systems, Man, and Cybernetics *21*, 660–674.

Sampurna, B., Audira, G., Juniardi, S., Lai, Y.-H. and Hsiao, C.-D. (2018). **A Simple ImageJ-Based Method to Measure Cardiac Rhythm in Zebrafish Embryos**. Inventions *3*, 21, doi:10.3390/inventions3020021.

Santi, P. A. (2011). **Light Sheet Fluorescence Microscopy: A Review**. J Histochem Cytochem *59*, 129–138, doi:10.1369/0022155410394857.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W. and Lin, C.-T. (2017). **A review of clustering techniques and developments**. Neurocomputing *267*, 664–681, doi:10.1016/j.neucom.2017.06.053.

Saydmohammed, M., Vollmer, L. L., Onuoha, E. O., Vogt, A. and Tsang, M. (2011). **A high-content screening assay in transgenic zebrafish identifies two novel activators of fgf signaling**. Birth Defects Research Part C: Embryo Today: Reviews *93*, 281–287, doi:10.1002/bdrc.20216.

Scherf, N. and Huisken, J. (2015). **The smart and gentle microscope**. Nature Biotechnology *33*, 815–818, doi:10.1038/nbt.3310.

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. and Cardona, A. (2012). **Fiji: an open-source platform for biological-image analysis**. Nature Methods *9*, 676–682, doi:10.1038/nmeth.2019.

Schmidt, U., Weigert, M., Broaddus, C. and Myers, G. (2018). **Cell Detection with Star-Convex Polygons**. In: **Medical Image Computing and Computer Assisted Intervention – MICCAI 2018**, Eds Frangi, A. F., Schnabel, J. A., Davatzikos, C., Alberola-López, C., and Fichtinger, G. 2018, Springer International Publishing, Cham, pp. 265–273.

Schneider, C. A., Rasband, W. S. and Eliceiri, K. W. (2012). **NIH Image to ImageJ: 25 years of image analysis**. Nature Methods *9*, 671–675, doi:10.1038/nmeth.2089.

Schorb, M. and Sieckmann, F. (2017). **Matrix MAPS—an intuitive software to acquire, analyze, and annotate light microscopy data for CLEM**. In: **Methods in Cell Biology**, Elsevier, pp. 321–333, doi:10.1016/bs.mcb.2017.03.012.

Schutera, M., Dickmeis, T., Mione, M., Peravali, R., Marcato, D., Reischl, M., Mikut, R. and Pylatiuk, C. (2016). **Automated phenotype pattern recognition of zebrafish for high-throughput screening**. Bioengineered *7*, 261–265, doi:10.1080/21655979.2016.1197710.

SciPy 1.0 Contributors, Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F. and van Mulbregt, P. (2020). **SciPy 1.0: fundamental algorithms for scientific computing in Python**. Nat Methods *17*, 261–272, doi:10.1038/s41592-019-0686-2.

Shah, K., Bedi, R., Rogozhnikov, A., Ramkumar, P., Tong, Z., Rash, B., Stanton, M., Sorokin, J., Apaydin, C., Batarse, A., Bergamaschi, J., Blattner, R., Brown, S., Bosshardt, A., Castrillo, C., Dang, B., Drusinsky, S., Enriquez, L., Grayson, D., Hilliard, J., Hsu, P.-K., Johnson, C., Jones, R., Lash, A., Lee, E., Li, K., McKay, A., Mount, E., Nicola, J., Oumzil, I., Paek, J., Pascoe, D., Piepho, A., Poust, S., Quang, D., Schultz, M., Sims, J., Taylor, P., Treiman, G., Wueseke, O., Young, N., Pollen, A. A., Flanzer, D., Chao, D., Skibinski, G., Kato, S. and Escola, S. (2020). Optimization and scaling of patient-derived brain organoids uncovers deep phenotypes of disease, Neuroscience, doi:10.1101/2020.08.26.251611.

Shang, S., Long, L., Lin, S. and Cong, F. (2019). **Automatic Zebrafish Egg Phenotype Recognition from Bright-Field Microscopic Images Using Deep Convolutional Neural Network**. Applied Sciences *9*, 3362, doi:10.3390/app9163362.

Shaw, P. J. (1995). **Comparison of Wide-Field/Deconvolution and Confocal Microscopy for 3D Imaging**. In: **Handbook of Biological Confocal Microscopy**, Ed. Pawley, J. B. Springer US, Boston, MA, pp. 373–387, doi:10.1007/978-1-4757-5348-6_23.

Shorten, C. and Khoshgoftaar, T. M. (2019). **A survey on Image Data Augmentation for Deep Learning**. J Big Data *6*, 60, doi:10.1186/s40537-019-0197-0.

Simm, J., Klambauer, G., Arany, A., Steijaert, M., Wegner, J. K., Gustin, E., Chupakhin, V., Chong, Y. T., Vialard, J., Buijnsters, P., Velter, I., Vapirev, A., Singh, S., Carpenter, A. E., Wuyts, R., Hochreiter, S., Moreau, Y. and Ceulemans, H. (2018). **Repurposing High-Throughput Image Assays Enables Biological Activity Prediction for Drug Discovery**. Cell Chemical Biology *25*, 611-618.e3, doi:10.1016/j.chembiol.2018.01.015.

Simonyan, K. and Zisserman, A. (2015). **Very Deep Convolutional Networks for Large-Scale Image Recognition**. arXiv:14091556 [cs], URL: http://arxiv.org/abs/1409.1556 [Accessed April 2020].

Singh, S., Bray, M.-A., Jones, T. R. and Carpenter, A. E. (2014). **Pipeline for illumination correction of images for high-throughput microscopy: ILLUMINATION CORRECTION FOR HIGH-THROUGHPUT IMAGES**. Journal of Microscopy *256*, 231–236, doi:10.1111/jmi.12178.

Sönnichsen, B., Koski, L. B., Walsh, A., Marschall, P., Neumann, B., Brehm, M., Alleaume, A.-M., Artelt, J., Bettencourt, P., Cassin, E., Hewitson, M., Holz, C., Khan, M., Lazik, S., Martin, C., Nitzsche, B., Ruer, M., Stamford, J., Winzi, M., Heinkel, R., Röder, M., Finell, J., Häntsch, H., Jones, S. J. M., Jones, M., Piano, F., Gunsalus, K. C., Oegema, K., Gönczy, P., Coulson, A., Hyman, A. A. and Echeverri, C. J. (2005). **Full-genome RNAi profiling of early embryogenesis in Caenorhabditis elegans**. Nature *434*, 462–469, doi:10.1038/nature03353.

Spanhol, F. A., Oliveira, L. S., Cavalin, P. R., Petitjean, C. and Heutte, L. (2017). **Deep features for breast cancer histopathological image classification**. In: **2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**, October 2017, IEEE, Banff, AB, pp. 1868–1873, doi:10.1109/SMC.2017.8122889.

Spomer, W., Pfriem, A., Alshut, R., Just, S. and Pylatiuk, C. (2012). **High-Throughput Screening of Zebrafish Embryos Using Automated Heart Detection and Imaging**. Journal of Laboratory Automation *17*, 435–442, doi:10.1177/2211068212464223.

Stacey, G. (2001). **Primary cell cultures and immortal cell lines**. e LS.

Stelzer, E. H. K. (2002). **Beyond the diffraction limit?** Nature *417*, 806–807, doi:10.1038/417806a.

Stern, O., Marée, R., Aceto, J., Jeanray, N., Muller, M., Wehenkel, L. and Geurts, P. (2011a). **Automatic localization of interest points in zebrafish images with tree-based methods**. Pattern Recognition in Bioinformatics 179–190.

Stern, O., Marée, R., Aceto, J., Jeanray, N., Muller, M., Wehenkel, L. and Geurts, P. (2011b). **Zebrafish Skeleton Measurements using Image Analysis and Machine Learning Methods**. Benelearn 2011 113.

Stirman, J. N., Crane, M. M., Husson, S. J., Wabnig, S., Schultheis, C., Gottschalk, A. and Lu, H. (2011). **Real-time multimodal optical control of neurons and muscles in freely behaving Caenorhabditis elegans**. Nat Methods *8*, 153–158, doi:10.1038/nmeth.1555.

Stringer, C., Wang, T., Michaelos, M. and Pachitariu, M. (2020). Cellpose: a generalist algorithm for cellular segmentation, Bioinformatics, doi:10.1101/2020.02.02.931238.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W. M., Hao, Y., Stoeckius, M., Smibert, P. and Satija, R. (2019). **Comprehensive Integration of Single-Cell Data**. Cell *177*, 1888-1902.e21, doi:10.1016/j.cell.2019.05.031.

Stuurman, N. (2012). **Intelligent Acquisition**, *Micro-Manager wiki documentation*, URL: https://micro-manager.org/wiki/Intelligent_Acquisition [Accessed October 2020].

Sun, Z., Amsterdam, A., Pazour, G. J., Cole, D. G., Miller, M. S. and Hopkins, N. (2004). **A genetic screen in zebrafish identifies cilia genes as a principal cause of cystic kidney**. Development *131*, 4085–4093.

Swedlow, J. R. and Platani, M. (2002). **Live Cell Imaging Using Wide-Field Microscopy and Deconvolution**. Cell Structure and Function *27*, 335–341, doi:10.1247/csf.27.335.

Szegedy, C., Toshev, A. and Erhan, D. (2013). **Deep neural networks for object detection**. In: **Advances in neural information processing systems**, 2013, pp. 2553–2561.

Tarasco, M., Cordelières, F. P., Cancela, M. L. and Laizé, V. (2020). **ZFBONE: An ImageJ toolset for semi-automatic analysis of zebrafish bone structures**. Bone *138*, 115480, doi:10.1016/j.bone.2020.115480.

Teague, M. R. (1980). **Image analysis via the general theory of moments**. Journal of the Optical Society of America *70*, 920, doi:10.1364/JOSA.70.000920.

Teixidó, E., Kießling, T. R., Krupp, E., Quevedo, C., Muriana, A. and Scholz, S. (2018). **Automated morphological feature assessment for zebrafish embryo developmental toxicity screens**. Toxicological Sciences, doi:10.1093/toxsci/kfy250.

Teknomo, K. (2006). **K-means clustering tutorial**. Medicine *100*, 3.

Thomas, L. (2019a). Multi-Template Matching - GitHub repositories, URL: https://multi-template-matching.github.io/Multi-Template-Matching/ [Accessed September 2020].

Thomas, L. (2019b). Multi-Template-Matching - YouTube tutorials. 13 September 2019, URL: https://www.youtube.com/playlist?list=PLHZOgc1s26MJ8QjYau7NcG5k0zh9SjHpo [Accessed October 2020].

Thomas, L. (2020a). Cascade classifiers estimations - excel sheet, URL: https://osf.io/bh4eg [Accessed November 2020].

Thomas, L. (2020b). Classification using template matching - KNIME workflow, URL: https://kni.me/w/DtQb80kyBESKsfsT [Accessed October 2020].

Thomas, L. (2020c). Custom phenotypic data for tensorflow embedding projector - GitHub, URL: https://github.com/LauLauThom/interactive-embedding-kidney [Accessed November 2020].

Thomas, L. (2020d). Fiji qualitative annotation plugins - YouTube tutorials. 15 September 2020, URL: https://www.youtube.com/playlist?list=PLbBgXlYof3_YVqR80jhFPCkc0M3GQMAq4 [Accessed October 2020].

Thomas, L. (2020e). **ImageJA Pull Request: Roi-group attribute**, *ImageJA GitHub repository*, URL: https://github.com/imagej/ImageJA/pull/18 [Accessed October 2020].

Thomas, L. (2020f). KNIME example workflows for the exploitation of qualitative image annotations, URL: https://hub.knime.com/l.thomas/spaces/Public/latest/Qualitative-Annotations/ [Accessed October 2020].

Thomas, L. (2020g). KNIME exploration dashboard - YouTube tutorial. 29 April 2020, URL: https://youtu.be/A4Eqe1Aju4A [Accessed November 2020].

Thomas, L. (2020h). KNIME features exploration dashboard, URL: https://kni.me/w/O3Q36l_xNLQ85IRg [Accessed October 2020].

Thomas, L. (2020i). Multi-Template-Matching - KNIME workflow, URL: https://kni.me/w/9i0_HPPQlbNzW598 [Accessed October 2020].

Thomas, L. (2020j). Qualitative annotation plugins for Fiji - GitHub repository, URL: https://github.com/LauLauThom/Fiji-QualiAnnotations [Accessed September 2020].

Thomas, L. (2020k). Roi 1-click tools - GitHub repository, URL: https://github.com/LauLauThom/Fiji-RoiClickTools [Accessed October 2020].

Thomas, L. (2020l). Roi 1-click tools - YouTube tutorials. 2020, URL: https://www.youtube.com/playlist?list=PLbBgXlYof3_aEXBmXkkuLyQ78aVj_xdan [Accessed October 2020].

Thomas, L. (2020m). *Using ROI categories for annotations in ImageJ/Fiji*., doi:10.7490/F1000RESEARCH.1117827.1.

Thomas, Laurent S.V. and Gehrig, J. (2020). **ImageJ/Fiji ROI 1-click tools for rapid manual image annotations and measurements**. microPublication Biology, doi:10.17912/micropub.biology.000215.

Thomas, Laurent S. V. and Gehrig, J. (2020). **Multi-template matching: a versatile tool for object-localization in microscopy images**. BMC Bioinformatics *21*, 44, doi:10.1186/s12859-020-3363-7.

Thomas, L. S. V., Schaefer, F. and Gehrig, J. (2020). **Fiji plugins for qualitative image annotations: routine analysis and application to image classification**. F1000Res *9*, 1248, doi:10.12688/f1000research.26872.1.

Thomas, Laurent, Gehrig, Jochen, Heigwer, Jana, Westhoff, Jens, Steenbergen, Petrus and Cooper, Ledean (2020). *Interactive feature exploration with KNIME - application to Zebrafish phenotyping*., doi:10.17605/OSF.IO/WPN2M.

Tinevez, J.-Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E., Bednarek, S. Y., Shorte, S. L. and Eliceiri, K. W. (2017). **TrackMate: An open and extensible platform for single-particle tracking**. Methods *115*, 80–90, doi:10.1016/j.ymeth.2016.09.016.

Tischer, C., Hilsenstein, V., Hanson, K. and Pepperkok, R. (2014). **Adaptive fluorescence microscopy by online feedback image analysis**. In: **Methods in Cell Biology**, Elsevier, pp. 489–503, doi:10.1016/B978-0-12-420138-5.00026-4.

Tseng, Q., Wang, I., Duchemin-Pelletier, E., Azioune, A., Carpi, N., Gao, J., Filhol, O., Piel, M., Théry, M. and Balland, M. (2011). **A new micropatterning method of soft substrates reveals that different tumorigenic signals can promote or reduce cell contraction levels**. Lab Chip *11*, 2231, doi:10.1039/c0lc00641f.

Turk, M. and Pentland, A. (1991). **Eigenfaces for Recognition**. Journal of Cognitive Neuroscience *3*, 71–86, doi:10.1162/jocn.1991.3.1.71.

Tzutalin (2015). LabelImg, URL: https://github.com/tzutalin/labelImg [Accessed September 2020].

Uc-Cetina, V., Brito-Loeza, C. and Ruiz-Piña, H. (2015). **Chagas Parasite Detection in Blood Images Using AdaBoost**. Computational and Mathematical Methods in Medicine *2015*, 1–13, doi:10.1155/2015/139681.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. and Smeulders, A. W. M. (2013). **Selective Search for Object Recognition**. Int J Comput Vis *104*, 154–171, doi:10.1007/s11263-013-0620-5.

Vaillant, R. (1994). **Original approach for the localisation of objects in images**. IEE Proc, Vis Image Process *141*, 245, doi:10.1049/ip-vis:19941301.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E. and Yu, T. (2014). **scikit-image: image processing in Python**. PeerJ *2*, e453, doi:10.7717/peerj.453.

VanderPlas, J. (2016). Python data-science handbook, O'Reilly Media, Inc, URL: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html [Accessed September 2020].

Viola, P. and Jones, M. (2001). **Rapid object detection using a boosted cascade of simple features**. In: **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001**, 2001, IEEE Comput. Soc, Kauai, HI, USA, p. I-511-I–518, doi:10.1109/CVPR.2001.990517.

Viola, P. and Jones, M. J. (2004). **Robust Real-Time Face Detection**. International Journal of Computer Vision *57*, 137–154, doi:10.1023/B:VISI.0000013087.49260.fb.

Wada, K. (2016). labelme: Image Polygonal Annotation with Python, URL: https://github.com/wkentaro/labelme.

Wagner, B. K. and Schreiber, S. L. (2016). **The Power of Sophisticated Phenotypic Screening and Modern Mechanism-of-Action Methods**. Cell Chemical Biology *23*, 3–9, doi:10.1016/j.chembiol.2015.11.008.

Wagner, J. and Gehrig, J. (2019). ACQUIFER - Manual Annotation Tool, Zenodo, doi:10.5281/ZENODO.3367366.

Wagner, T. and Lipinski, H.-G. (2013). **IJBlob: an ImageJ library for connected component analysis and shape analysis**. Journal of Open Research Software *1*.

Waithe, D., Brown, J. M., Reglinski, K., Diez-Sevilla, I., Roberts, D. and Eggeling, C. (2020). **Object detection networks and augmented reality for cellular detection in fluorescence microscopy**. Journal of Cell Biology *219*, e201903166, doi:10.1083/jcb.201903166.

Wakamatsu, Y., Pristyazhnyuk, S., Kinoshita, M., Tanaka, M. and Ozato, K. (2001). **The see-through medaka: a fish model that is transparent throughout life**. Proc Natl Acad Sci U S A *98*, 10046–10050.

Wang, K., Milkie, D. E., Saxena, A., Engerer, P., Misgeld, T., Bronner, M. E., Mumm, J. and Betzig, E. (2014). **Rapid adaptive optical recovery of optimal resolution over large volumes**. Nat Methods *11*, 625–628, doi:10.1038/nmeth.2925.

Wattenberg, M., Viégas, F. and Johnson, I. (2016). **How to Use t-SNE Effectively**. Distill *1*, 10.23915/distill.00002, doi:10.23915/distill.00002.

Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., Wilhelm, B., Schmidt, D., Broaddus, C., Culley, S., Rocha-Martins, M., Segovia-Miranda, F., Norden, C., Henriques, R., Zerial, M., Solimena, M., Rink, J., Tomancak, P., Royer, L., Jug, F. and Myers, E. W. (2018). **Content-aware image restoration: pushing the limits of fluorescence microscopy**. Nat Methods *15*, 1090–1097, doi:10.1038/s41592-018-0216-7.

Weigert, M., Schmidt, U., Haase, R., Sugawara, K. and Myers, G. (2020). **Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy**. In: **2020 IEEE Winter Conference on Applications of Computer Vision (WACV)**, March 2020, IEEE, Snowmass Village, CO, USA, pp. 3655–3662, doi:10.1109/WACV45572.2020.9093435.

Westhoff, J. H., Steenbergen, P. J., Thomas, L. S. V., Heigwer, J., Bruckner, T., Cooper, L., Tönshoff, B., Hoffmann, G. F. and Gehrig, J. (2020). **In vivo High-Content Screening in Zebrafish for Developmental Nephrotoxicity of Approved Drugs**. Front Cell Dev Biol *8*, 583, doi:10.3389/fcell.2020.00583.

White, R. M. (2015). **Cross-species oncogenomics using zebrafish models of cancer**. Current Opinion in Genetics & Development *30*, 73–79, doi:10.1016/j.gde.2015.04.006.

White, R., Rose, K. and Zon, L. (2013). **Zebrafish cancer: the state of the art and the path forward**. Nat Rev Cancer *13*, 624–636.

Wittbrodt, J. N., Liebel, U. and Gehrig, J. (2014). **Generation of orientation tools for automated zebrafish screening assays using desktop 3D printing**. BMC Biotechnology *14*, 36, doi:10.1186/1472-6750-14-36.

Wold, S., Esbensen, K. and Geladi, P. (1987). **Principal component analysis**. Chemometrics and Intelligent Laboratory Systems *2*, 37–52, doi:10.1016/0169-7439(87)80084-9.

Wu, Z., Wang, X., Jiang, Y.-G., Ye, H. and Xue, X. (2015). **Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification**. In: **Proceedings of the 23rd ACM international conference on Multimedia - MM '15**, 2015, ACM Press, Brisbane, Australia, pp. 461–470, doi:10.1145/2733373.2806222.

Xia, S., Zhu, Y., Xu, X. and Xia, W. (2013). **Computational techniques in zebrafish image processing and analysis**. Journal of Neuroscience Methods *213*, 6–13, doi:10.1016/j.jneumeth.2012.11.009.

Yang, H., Deng, R., Lu, Y., Zhu, Z., Chen, Y., Roland, J. T., Lu, L., Landman, B. A., Fogo, A. B. and Huo, Y. (2020). **CircleNet: Anchor-free Detection with Circle Representation**. arXiv:200602474 [cs], URL: http://arxiv.org/abs/2006.02474 [Accessed June 2020].

Young, D. W., Bender, A., Hoyt, J., McWhinnie, E., Chirn, G.-W., Tao, C. Y., Tallarico, J. A., Labow, M., Jenkins, J. L., Mitchison, T. J. and Feng, Y. (2008). **Integrating high-content screening and ligand-target prediction to identify mechanism of action**. Nat Chem Biol *4*, 59–68, doi:10.1038/nchembio.2007.53.

Young, Glasbey, Gray, and Martin (1998). **Towards automatic cell identification in DIC microscopy**. Journal of Microscopy *192*, 186–193, doi:10.1046/j.1365-2818.1998.00397.x.

Zheng, W., Thorne, N. and McKew, J. C. (2013). **Phenotypic screens as a renewed approach for drug discovery**. Drug Discovery Today *18*, 1067–1073, doi:10.1016/j.drudis.2013.07.001.

Zhou, W., Boucher, R. C., Bollig, F., Englert, C. and Hildebrandt, F. (2010). **Characterization of mesonephric development and regeneration using transgenic zebrafish**. American Journal of Physiology-Renal Physiology *299*, F1040–F1047, doi:10.1152/ajprenal.00394.2010.

Zhou, X., Wang, D. and Krähenbühl, P. (2019a). **Objects as Points**. arXiv:190407850 [cs], URL: http://arxiv.org/abs/1904.07850 [Accessed October 2020].

Zhou, X., Zhuo, J. and Krahenbuhl, P. (2019b). **Bottom-Up Object Detection by Grouping Extreme and Center Points**. In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**, June 2019, IEEE, Long Beach, CA, USA, pp. 850–859, doi:10.1109/CVPR.2019.00094.

Zhu, Y., Chen, Y., Lu, Z., Pan, S. J., Xue, G.-R., Yu, Y. and Yang, Q. (2011). *Heterogeneous Transfer Learning for Image Classification*. 7.

Zitnick, C. L. and Dollár, P. (2014). **Edge Boxes: Locating Object Proposals from Edges**. In: **Computer Vision – ECCV 2014**, Eds Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. Springer International Publishing, Cham, pp. 391–405, doi:10.1007/978-3-319-10602-1_26.

8. Contributions and publications

This research project was part of the Horizon2020/Marie Skłodowska-Curie *ImageInLife* research program for the training of early stage researchers (PhD students) as experts in microscopy and bioimage analysis. The research reported here was carried out at ACQUIFER under the supervision of Jochen Gehrig, with collaborations from the medical university of Heidelberg.

Below is a list of peer-reviewed publications describing software tools developed as part of this research (1-3); and illustrating the previously described interactive exploration dashboard with a concrete phenotypic screening study (4).

1. Thomas, L. S. V. and Gehrig, J. (2020). **Multi-template matching: a versatile tool for object-localization in microscopy images**. BMC Bioinformatics *21*, 44, doi:10.1186/s12859-020-3363-7.
   See sections 2.4 and 3.3.

2. Thomas, L. S. V. and Gehrig, J. (2020). **ImageJ/Fiji ROI 1-click tools for rapid manual image annotations and measurements**. microPublication Biology, doi:10.17912/micropub.biology.000215.
   See sections 2.3.1 and 3.2.2.

3. Thomas, L. S. V., Schaefer, F. and Gehrig, J. (2020). **Fiji plugins for qualitative image annotations: routine analysis and application to image classification**. F1000Res *9*, 1248, doi:10.12688/f1000research.26872.1.
   See sections 2.3.2 and 3.2.3.

4. Westhoff, J. H., Steenbergen, P. J., Thomas, L. S. V., Heigwer, J., Bruckner, T., Cooper, L., Tönshoff, B., Hoffmann, G. F. and Gehrig, J. (2020). **In vivo High-Content Screening in Zebrafish for Developmental Nephrotoxicity of Approved Drugs**. Front Cell Dev Biol *8*, 583, doi:10.3389/fcell.2020.00583.
   See section 3.8.

9. Acknowledgements

I am first very grateful to Jochen Gehrig, my daily supervisor at ACQUIFER for his precious guidance, feedback and suggestions, and the time spent reviewing manuscripts, this thesis and extensively testing the software tools resulting from this research. I am glad to have the opportunity to continue working in this team after my PhD as an employee of ACQUIFER.

I also benefited from the help of many dedicated members of the open-source software community, some of them whom I actually never met! They took the time to answer my questions on the online forums, keep fixing bugs, implementing new functionalities, and making sure that everything sticks together. I am thus particularly thankful to Wayne Rasband for still improving ImageJ after so many years, Curtis Rueden and Jan Eglinger for maintaining the Fiji ecosystem and their precious help on the forum, Gabriel Einsdorf, Stefan Helfrich, Christian Dietz and other KNIMErs, Jonathan Heras for IJ-OpenCV and many others who offered some help on the forum or contributed with new functionalities.

I am also grateful to Gunjan Pandey, Jakob Gierten, Eleonora Lupi, Jana Heigwer, Jens Westhoff, Jochen Wittbrodt and Franz Schaefer for contributing directly or indirectly to this project. I am hoping to see you around Heidelberg or at conferences in the future.

I would like to address my recognition to the coordinators and members of the *ImageInLife* Innovative Training Network who dedicated time and efforts to make this research happen and to organize original training events. I am similarly grateful to my colleagues at Ditabis and ACQUIFER for dealing with the administrative matters and providing a friendly working environment.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme to Ditabis under the Marie Sklodowska-Curie grant agreement No 721537 “ImageInLife”. I am glad to count among the Marie-Curie fellows, and I hope that Europe will remain a place of choice for original research and will drive novel changes to face the challenges of the future.

10. Eidesstattliche Versicherung

1. Bei der eingereichten Dissertation zu dem Thema “*Computer vision and data-analysis solutions for phenotypic screening of small model organisms*” handelt es sich um meine eigenständig erbrachte Leistung.

2. Ich habe nur die angegebenen Quellen und Hilfsmittel benutzt und mich keiner unzulässigen Hilfe Dritter bedient. Insbesondere habe ich wörtlich oder sinngemäß aus anderen Werken übernommene Inhalte als solche kenntlich gemacht.

3. Die Arbeit oder Teile davon habe ich bislang nicht an einer Hochschule des In- oder Auslands als Bestandteil einer Prüfungs- oder Qualifikationsleistung vorgelegt.

4. Die Richtigkeit der vorstehenden Erklärungen bestätige ich.

5. Die Bedeutung der eidesstattlichen Versicherung und die strafrechtlichen Folgen einer unrichtigen oder unvollständigen eidesstattlichen Versicherung sind mir bekannt.
Ich versichere an Eides statt, dass ich nach bestem Wissen die reine Wahrheit erklärt und nichts verschwiegen habe.

Ort und Datum
Mannheim, 19/11/2020

Unterschrift