

Aus dem Bereich der Molekularen Zellbiologie
Theoretische Medizin und Biowissenschaften
der Medizinischen Fakultät
der Universität des Saarlandes, Homburg / Saar

Development of a System for Optical High-Resolution Screening of Primary Cultured Cells

*Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Medizinischen Fakultät
der UNIVERSITÄT DES SAARLANDES
2010*

vorgelegt von: Oliver Müller
geb. am: 04.10.1977 in 66440 Blieskastel

Oliver Müller
Institut für Molekulare Zellbiologie
Kirrbergerstr. Geb. 61
66421 Homburg / Saar
Germany
E-Mail: oliver.mueller@uniklinik-saarland.de

Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater/innen oder anderer Personen) in Anspruch genommen.

Außer den Angegebenen hat niemand von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form in einem anderen Verfahren zur Erlangung des Doktorgrades einer anderen Prüfungsbehörde vorgelegt.

Ich versichere an Eides statt, dass ich nach bestem Wissen die Wahrheit gesagt und nichts verschwiegen habe.

Vor Aufnahme der vorstehenden Versicherung an Eides statt wurde ich über die Bedeutung einer eidesstattlichen Versicherung und die strafrechtlichen Folgen einer unrichtigen oder unvollständigen eidesstattlichen Versicherung belehrt.

Ort, Datum

Unterschrift der/des Promovierenden

Unterschrift der die Versicherung an Eides statt aufnehmenden

Beamtin bzw. des aufnehmenden Beamten

Zusammenfassung

Kommerzielle High-Content Screening Systeme kommen in der pharmazeutischen Industrie immer häufiger zum Einsatz. Sie sind meist auf eine bestimmte Art von Problem zugeschnitten und werden in Verbindung mit Zelllinien verwendet, da diese leicht zu handhaben sind und in der Regel reproduzierbare Ergebnisse liefern. Im Gegensatz dazu werden primär kultivierte Zellen in diesem Zusammenhang kaum verwendet (unter anderem, da sie schlechter verfügbar sind, besonders in größeren Mengen). Darüber hinaus sind sie häufig von Problemen wie beispielsweise einer hohen Fragilität und einer daraus resultierenden komplizierten Handhabung, Dedifferenzierung während der Kultivierung oder Donor-abhängigen Qualitätsunterschieden betroffen. Da diese Zellen näher an ihrem in vivo-Pendant sind als Zelllinien, sind realistischere Erkenntnisse von zellulären Prozessen unter quasi-physiologischen Bedingungen möglich. Ein Anwendungsgebiet für die Untersuchung von adulten, primär isolierten Kardiomyozyten ist die Entschlüsselung zellulärer und molekularer Ursachen von Herzrhythmusstörungen und anderer Herzkrankheiten.

Daher war mein primäres Ziel – basierend auf Prototyp-Komponenten von Industriepartnern und neu entworfenen Modulen – die Entwicklung eines automatisierten High-Content Screening Systems, welches die Analyse adulter, primär kultivierter Herzmuskelzellen mit Hilfe hochauflösender Bildgebung erlaubt. Um Flexibilität und Skalierbarkeit zu gewährleisten, habe ich das Gesamtsystem in eigenständige Module für experimentelle Kontrolle, Datenanalyse, -speicherung und -archivierung sowie Datenkommunikation und Koordination unterteilt. Für jedes Modul habe ich entsprechende Hardware und / oder Software entwickelt.

Insgesamt konnte ich den Machbarkeitsnachweis in Hinblick auf die oben genannte biologische Fragestellung erbringen. Die Flexibilität des Systems wurde durch Anwendung auf ein weiteres biologisches Problem, welches die Untersuchung von Erythrozyten zum Gegenstand hatte, demonstriert.

Abschließend möchte ich festhalten, dass sowohl auf Hardware- als auch auf Softwareseite die Modularität ein Schlüsselkonzept bei der Entwicklung einer flexiblen, skalierbaren und erweiterbaren High-Content Screening Umgebung ist. Ein offenes Design unterstützt dabei die Eingliederung selbstentwickelter, aber auch kommerziell erhältlicher Komponenten mit einer offenen Programmierschnittstelle. Das in dieser Arbeit vorgestellte automatisierte System zur Analyse primär kultivierter Zellen dient als Grundlage für schnellere und umfangreichere Untersuchungen von Herzkrankheiten wie Herzrhythmusstörungen auf zellulärer Ebene.

Abstract

Commercial high-content screening systems are increasingly used in pharmaceutical industry. Mostly designed for a specific type of problem, such systems are applied to cell lines, which usually are easy to handle and enable reproducible results. In contrast, primary cultured cells are hardly used in such applications, since they are more difficult to get, especially in large numbers. Additionally, working with primary cells is often accompanied by challenges such as high fragility and complicated handling, dedifferentiation during the culture period or donor-dependent variations in quality. Since these cells are closer to their *in vivo* counterpart than cell lines, more realistic insights into cellular processes under quasi-physiological conditions are possible. An application area is the study of cellular and molecular causes of cardiac diseases such as arrhythmia.

Therefore, my primary goal was – based on prototype components of industrial partners and newly designed modules – the development of an automated high-content screening system. This enables high-resolution imaging of primary cultured cardiac muscle cells. To maintain flexibility and scalability, I subdivided the overall system into self-contained modules for experimental control, data analysis, data storage and archiving as well as data communication and coordination. For each module, I developed additional hardware and / or software. Altogether, a proof of concept for the above mentioned biological aim was achieved. The flexibility of the system was demonstrated by application to a different biological problem: studies on erythrocytes.

I conclude, that the modularity of both, the hardware and the software side is a key concept for maintaining flexibility, scalability and expandability of a high-content screening environment. An open design fosters the incorporation of self-developed as well as commercially available components with an open application programming interface. As an automated tool suitable for primary cultured cells, the system presented in this thesis is the basis for faster and more comprehensive investigations of cardiac diseases such as arrhythmia on a cellular level.

List of Figures

1.1	Electrical pathways of the human heart.	2
1.2	Cross-section of a cardiomyocyte.	3
1.3	Coherence between calcium concentration and cell length of a rat ventricular myocyte.	4
1.4	Schematic representation of the principle of fluorescence microscopy.	7
1.5	Scheme of an HCS procedure.	12
1.6	Flowchart of the typical steps involved in image-based HCS data generation.	14
1.7	Different types of storage media with regard to access time and persistency.	18
2.1	The HCS stimulation plate.	24
2.2	The pin assignment of the two connector blocks NI SCC-68.	27
2.3	Voltage traces of the relay types OMRON G3DZ and Panasonic AQY225R1S.	29
2.4	more™ connection scheme of the different components.	31
2.5	Schematic drawing of the digital microscope more™.	32
2.6	Example program written in LabVIEW.	37
2.7	Scheme of the OMERO client-server infrastructure.	45
2.8	Architecture of the OME server and the OMERO platform.	46
2.9	Schematic drawing of the OMERO server design.	47
2.10	Structure of an XML-TIFF header.	49
2.11	Distributed computing environment of our HCS system.	50

3.1	Information flow diagram of the setup running the TriggerIT software.	58
3.2	GUI of the TriggerIT software.	59
3.3	Flowchart of the TriggerIT software.	60
3.4	GUI of the ClickIT software.	63
3.5	The PROTOzoon setup.	65
3.6	GUI of the PROTOzoon software.	66
3.7	Scheme displaying the generation of the alternating negative / positive pulse train.	66
3.8	A snippet of LabVIEW code from the PROTOzoon software.	67
3.9	Arrangement of the different hardware components for electrical field stimulation.	69
3.10	Pulsing pattern of the 24-well plate.	70
3.11	Scheme of the circuit diagram used for each well of the plate.	71
3.12	3D-model of the circuit of the SSRD.	72
3.13	Construction plans of the SSRD.	73
3.14	Blueprint of the circuit paths and drill-holes of the lid.	75
3.15	GUI of the Pulsing Device software.	77
3.16	Architecture of the Pulsing Device software.	79
3.17	Wiring scheme of the perfusion system.	81
3.18	Overview of the software concept of CellAnalyser.	87
3.19	CellAnalyser: resulting $\frac{\Delta F}{F_0}$ plot for 25 erythrocytes.	93
3.20	Flowchart of the ATOM software.	95
3.21	Java class dependencies of the ATOM software.	99
3.22	Schematic drawing of our network architecture.	102
3.23	Scheme of the messaging framework.	104
3.24	Flowchart of the Messaging Server software.	105
3.25	C++ class hierarchy of the Messaging Server software.	107
3.26	Experimental protocol of the GCaMP2 experiment.	111
3.27	Hardware for electrical and chemical stimulation.	111
3.28	Data manager of the OMERO.insight client.	112
3.29	CellAnalyser - GUI with labelled ROI's.	113

3.30	Table of intensity values for each ROI.	114
3.31	Fluorescence transients for each ROI.	115
3.32	Project overview.	116

List of Abbreviations

{2, 3, 4, 5}D	{two, three, four, five} dimensional
ABMI	Arivis Core Mathworks MATLAB™ Interface
ACC	Advanced Cell Classifier
AF	Atrial fibrillation
ANN	Artificial neural network
ANSI	American National Standards Institute
API	Application programming interface
ATOM	AuTO.iMporter
CCD	Charge coupled device
CLI	Command line interface
CPU	Central processing unit
CVI	LabWindows / C for virtual instrumentation
DLL	Dynamic-link library
DMZ	Demilitarised zone
GND	Ground
GPIB	General purpose interface bus
GUI	Graphical user interface
HCS	High-content screening

HTS	High-throughput screening
ICU	Imaging control unit
I/O	Input / output
IP	Internet protocol
JRE	Java runtime environment
JVM	Java virtual machine
LA	TILL Live Acquisition
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
LED	Light-emitting diode
LTO	Linear tape-open
MAX	Measurement & Automation Explorer
NA	Numerical aperture
OME	Open Microscopy Environment
OMERO	OME Remote Objects
PC	Personal computer
PCI	Peripheral component interconnect
PFI	Programmable function interface
RAID	Redundant array of independent discs
RAM	Random access memory
RGB	Red, green, blue
RNA	Ribonucleic acid
ROI	Region of interest
SCP	Secure copy
SIL	Single in line
SIS	Single image stack
SR	Sarcoplasmic reticulum

SSR	Solid state relay
SSRD	SSR device
TCP	Transmission control protocol
TIFF	Tagged image file format
TTL	Transistor-transistor logic
UI	Usable Image
USB	Universal serial bus
VI	Virtual instrument
WWW	World wide web
XML	Extensible markup language

List of Tables

2.1	Summary of characteristics of three different types of solid state relays.	28
2.2	Summary of characteristics of the dual power supply TOE 8732-2.	30
2.3	more™: equipment of our objective slider.	33
2.4	Summary of specifications of the eServer and the pServer.	35
2.5	Summary of specifications of the disc shelves.	35
2.6	Summary of specifications of the tape library.	36
3.1	I/O channel setup of the data acquisition board DT335.	60
3.2	Overview of command line parameters of the ATOM software.	96
B.1	Overview of implemented XML commands and their interrelation to the buttons of the GUI.	151
B.2	Components of the SSRD.	152
B.3	Pin layout of the control voltage circuit (NI SCC-68), patch bay routing and of the HCS stimulation plate.	153
B.4	Patch bay components.	154
B.5	Connector pin assignment of the control unit.	154

Contents

1	Introduction	1
1.1	The Mammalian Heart	1
1.1.1	Structure and Function of the Heart	1
1.1.2	Cardiomyocytes	2
1.1.3	Excitation-Contraction Coupling	3
1.1.4	Cardiac Arrhythmia	5
1.2	Optical Imaging Techniques in Cell Biology	5
1.2.1	Fluorescence Microscopy	6
1.2.2	Video and Confocal Microscopy	7
1.2.3	The Digital Image	8
1.2.4	Live Cell Imaging	9
1.3	Image-based High-Content Screening	10
1.3.1	High-Throughput Screening	10
1.3.2	High-Content Screening	11
1.3.3	Screening of Primary Cultured Cells	12
1.4	Data	13
1.4.1	Raw Image Data	13
1.4.2	Metadata	15
1.4.3	Experimental Results and Data Management	16
1.5	Objectives of the Study	18
2	Materials and Methods	20
2.1	Hardware for Electrical Field Stimulation	20

2.1.1	HCS Stimulation Plate	20
2.1.2	Data Acquisition Boards for Digital Input / Output	25
2.1.3	I/O Connector Blocks	26
2.1.4	Solid State Relays	28
2.1.5	Voltage Source	29
2.2	Hardware for Application of Test Substances	30
2.2.1	Mounting Plate: Pinch Valves	30
2.2.2	Control Unit: Universal Serial Bus Relay Modules	30
2.3	Hardware for Image Data Acquisition	31
2.3.1	Automated Light Microscopy Platform more™	32
2.3.2	Polychrome V	34
2.3.3	Imaging Control Unit	34
2.4	Hardware for Accessing, Storing and Archiving Data	34
2.4.1	Servers	35
2.4.2	Disc Shelves	35
2.4.3	Tape Library	36
2.5	Programming Languages and Development Environments	36
2.5.1	Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW)	36
2.5.2	LabWindows / C for Virtual Instrumentation (CVI)	39
2.5.3	C++	39
2.5.4	Boost C++ Library	40
2.5.5	Java	40
2.5.6	Eclipse Integrated Development Environment	40
2.5.7	Matrix Laboratory (MATLAB)	41
2.6	Software for Computer Aided Design of Circuit Boards	41
2.6.1	KiCad	41
2.6.2	GC-Prevue	42
2.7	Software for Managing, Storing and Archiving Data	42
2.7.1	OME and OMERO	42
2.7.2	Bacula®	49

2.8	Software for Distributed Computing	50
2.8.1	arivis Grid	51
2.8.2	arivis Browser	52
2.8.3	arivis Core Mathworks MATLAB™ Interface	52
2.8.4	TILL Live Acquisition Software	53
2.9	Approaches for Image Processing and Image Analysis	53
2.9.1	Watershed Transformation	54
2.9.2	Edge Detection	54
2.9.3	Circular Hough Transform	56
2.10	Biological Specimen	56
2.10.1	Cardiomyocytes	56
2.10.2	Erythrocytes	56
3	Results	57
3.1	Preparatory Work	57
3.1.1	Software for Triggering Laboratory Equipment: TriggerIT	57
3.1.2	Enhancement of the TriggerIT Software: ClickIT	61
3.1.3	A Software Prototype for Electrical Field Stimulation of a Single Well: PROTOzoon	63
3.1.4	Long-term Material Testing	67
3.2	Experimental Control	68
3.2.1	General Setup	68
3.2.2	Settings of the Voltage Source	68
3.2.3	Solid State Relay Device	70
3.2.4	Patch Bay	73
3.2.5	HCS Stimulation Plate	74
3.2.6	Software for Electrical Field Stimulation: Pulsing Device	75
3.2.7	Perfusion: Mounting Plate and Control Unit	81
3.2.8	Software for Application of Test Substances: Perfusion System	82

3.3	Data Processing and Analysis	82
3.3.1	An Open Image Analysis Toolbox for Fluorescence and White Light Imaging: CellAnalyser	82
3.3.2	Use Case I: Automated Detection and Analysis of Cardiomy- ocytes	86
3.3.3	Use Case II: Automated Detection and Analysis of Erythro- cytes	91
3.4	Data Storage, Archiving and Management	93
3.4.1	Software for Automated Import of Images into OMERO: AuTO.iMporter (ATOM)	93
3.4.2	Network Security and Backup Solution	98
3.5	Data Communication and Coordination	102
3.5.1	A Multi-Client Communication Framework: Messaging Server	102
3.5.2	Use Case: Example Screening Run	108
3.5.3	Practical Application: Screening Inside a Single Well	110
3.5.4	General Outline of the HCS Project	116
4	Discussion	117
4.1	Hardware	117
4.1.1	Electrical Stimulation Involving Individual Wells	117
4.1.2	Elastic Coating and Oleophobic Foil	119
4.2	Software	120
4.2.1	Image Analysis Software for Automated Cell Detection and Classification	120
4.2.2	Potential Improvements of the CellAnalyser Framework	123
4.2.3	ATOM and OMERO.dropbox	125
4.3	Overall System	126
4.4	Future Perspectives	127
4.4.1	Workflow Essentials	127
4.4.2	Add-ons	128
4.5	Conclusion and Outlook	129

Bibliography	131
A Documentations	142
A.1 Pulsing Device API	142
A.2 Perfusion System API	148
B Tables	151
C Programs	155

Chapter 1

Introduction

1.1 The Mammalian Heart

1.1.1 Structure and Function of the Heart

The main task of the heart is to perform rhythmic contractions and relaxations in an alternating, steady and reliable manner, thus pumping blood through the body. This allows supplying each single cell of the body with oxygen and nutritive substances, while at the same time, metabolic wastes are being removed. The heart is a muscular, hollow organ basically consisting of two kinds of chambers: two upper chambers (the atria) and two lower chambers (the ventricles). To ensure an autonomous and reliable heartbeat, the heart is equipped with an intrinsic conductive system, which consists of specialised myocardium. Pacemaker cells generate rhythmic impulses, which are transmitted through the conductive system. These rhythmic impulses stimulate the working myocardium, thus leading to contractions of the heart muscle. The electrical pathways during normal sinus rhythm in the human heart are shown in Figure 1.1(a). Abnormal electrical pathways as they occur during atrial fibrillation, a heart disease that is further described in Section 1.1.4, are shown in Figure 1.1(b).

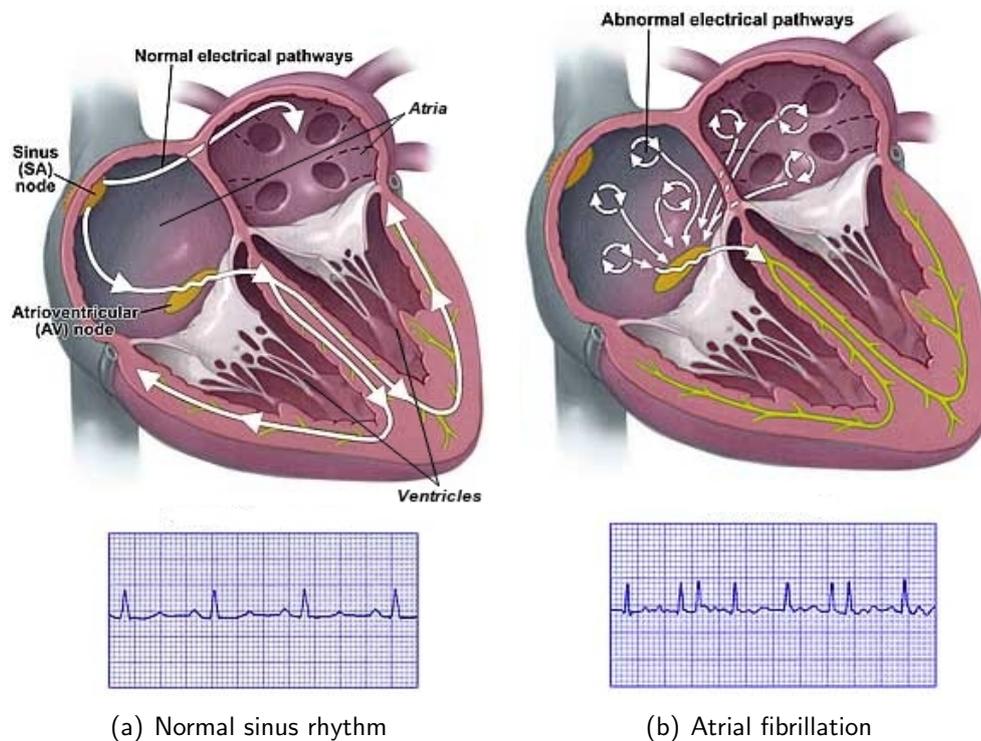


Figure 1.1: Electrical pathways of the human heart. (a) Shows the electrical pathways during normal sinus rhythm. (b) Shows the abnormal electrical pathways occurring during atrial fibrillation. The corresponding electrocardiogram recordings are displayed below the cross-section of the heart (modified from [74]).

1.1.2 Cardiomyocytes

Heart muscle cells, which are referred to as cardiac myocytes or cardiomyocytes, represent the principal constituents of the heart muscle tissue. The myocardium shares properties with the skeletal muscle such as rapid contraction and relaxation cycles. But in contrast to the skeletal muscle, the heart muscle cannot be contracted deliberately. Furthermore, a tetanic contraction of the heart muscle is not possible. In order to provide the required amount of energy, the myocardium is rich in mitochondria and sarcoplasmic reticulum (SR), a special type of smooth endoplasmic reticulum. Cardiomyocytes of the working myocardium

have a length of about $100\ \mu\text{m}$ and a diameter of $10\text{--}20\ \mu\text{m}$ (values for human cardiomyocytes [106]). They can contain one to three nuclei [106] (two nuclei are common). The cells are arranged in an end-to-end manner sharing a basal lamina. Connections between adjacent cells are called intercalated disc. Gap junctions allow for communication between neighbouring cells and thus for electrical synchronisation of contractions. A cross-section of a cardiomyocyte is shown in Figure 1.2.

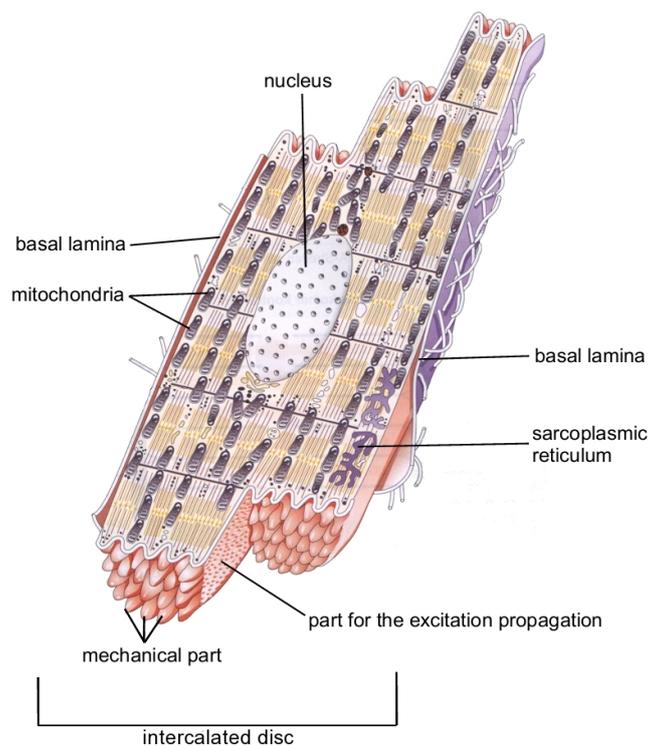


Figure 1.2: Cross-section of a cardiomyocyte (modified from [114]).

1.1.3 Excitation-Contraction Coupling

Each cardiomyocyte of the working myocardium performs rhythmic contractions and relaxations, thus contributing to the pumping performance of the heart. On

the cellular level, these contractions are initiated by electrical pulses that evolve calcium influx and trigger calcium-induced calcium release from the SR, which leads to contraction of the cell [14]. Figure 1.3 depicts the concomitant occurrence of increases in calcium concentration (upper trace) and shortenings of the cardiomyocyte (lower trace). In our institute standard procedures for isolation and

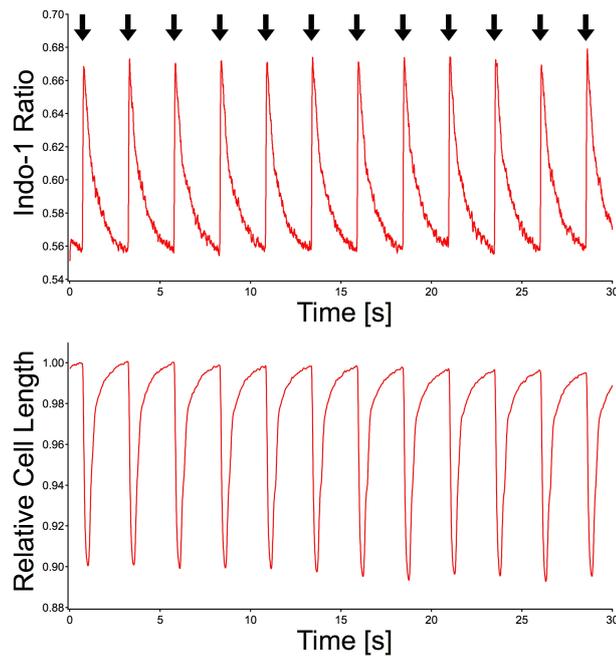


Figure 1.3: Coherence between calcium concentration and cell length of a rat ventricular myocyte. The cell has been loaded with the calcium indicator Indo-1/AM. Each electrical pulse (black arrows) is accompanied by a calcium transient (upper plot) and a shortening of the cell length (lower plot). (Figure adapted with permission from Qinghai Tian).

long-term culture of primary adult cardiomyocytes have been developed [61, 130]. During their time in culture, the properties of the cells are largely preserved, which allows an investigation of the cells for a period of up to one week. Automated methods for examination of calcium signals in cardiomyocytes with a high throughput rate would allow for capturing, analysing and storing the information content of a large number of cells in parallel. This could potentially reveal new insight into

cellular mechanisms involved in cardiac pathology and thus foster the understanding as well as the development of novel pharmacological strategies for the diseased heart.

1.1.4 Cardiac Arrhythmia

The most common cardiac arrhythmia is atrial fibrillation (AF) [67,84]. The existence of AF became known in 1874, when Edemé Félix Alfred Vulpian¹ reported, that a strong and continuous current directly applied to the atria of a dog's heart induces fibrillation of the atria [38]. This fibrillation is an unsteady, rapid and uncontrolled contraction of the atrial heart muscle tissue. AF has been observed in patients that suffered from a myocardial infarction (secondary AF) [39] as well as in patients without a detectable heart disease (lone AF) [57,132]. Considering the acute situation, AF is a non-life-threatening dysfunction (in contrast to ventricular fibrillation). It can be either transient or permanent and increases in prevalence with age [39,43]. Electrical stimulation of the atria with a high frequency can lead to an almost vanishing movement of the atrial muscle tissue, which results in a disturbed blood flow inside the atria, fostering blood clumping and atrial thrombus formation. This bears the risk of a pulmonary embolism (if the right atrium is affected) or a stroke (if the left atrium is affected) [21,133]. Electrophysiological and structural changes of the atrial myocardium caused by AF have been observed. These changes are referred to as atrial remodelling [46,72]. AF can be determined in an electrocardiogram, as depicted in the lower panel of Figure 1.1(b) – the upper panel shows the abnormal electrical pathways during AF.

1.2 Optical Imaging Techniques in Cell Biology

In order to better understand physiological and pathophysiological processes inside cells, special techniques, which allow for visualising the cell's subcellular structures and internal processes, have been developed. The term optical imaging encom-

¹A French physiologist and neurologist.

passes all processes from resolving the details of an object by means of light microscopy to capturing these details into a digital image and making them available for analyses. One of the first microscopes has been used by the English physicist and mathematician Robert Hooke². In 1665, Hooke coined the term “cell”, while he was studying the structure of cork with the aid of his microscope [28]. Since then, light microscopy has evolved. In 1908, August Köhler³ introduced luminescence microscopy as an application in microscopy [23]. In the 1920s, fluorescence microscopy emerged in the field of biological / medical science [23].

1.2.1 Fluorescence Microscopy

The visualisation of some cellular components requires the application of dyes. Using transmission light microscopes, absorption dyes come into operation, while fluorescent microscopes require special fluorescent probes. The basic principle of fluorescence microscopy is depicted in Figure 1.4. It starts with the excitation of the fluorescent dye. According to the excitation spectrum of the dye, an excitation filter allows light with a certain excitation wavelength to pass from a light source to a dichroic mirror. This mirror reflects the excitation light towards the objective, which focuses it onto the specimen.

Following excitation of the dye molecules, fluorescent light is collected by the objective. The wavelength of the emitted fluorescence light is usually longer than the wavelength of the excitation light. Thus, the dichroic mirror does not reflect the emitted fluorescence light. Instead, it allows the light to pass onto an emission filter, which ensures, that only the fluorescent light is collected by the detector.

In order to collect as much fluorescent light as possible, an objective with a high numerical aperture (NA) is desired. The higher the NA of an objective, the better is its ability to collect light from the specimen. The NA can be computed by Equation 1.1 taken from [102]:

²Robert Hooke: *18th of July 1635, †3rd of March 1703.

³ August Köhler: *4th of March 1866, †12th of March 1948.

$$NA = n \cdot \sin\alpha \quad (1.1)$$

where

NA : numerical aperture

n : index of refraction

α : aperture angle

It can be increased by either increasing the aperture angle α of the objective or by applying a medium with a higher index of refraction n . The higher the NA of an objective, the higher the resolution an object can be displayed with. Oil immersion objectives can have a higher NA than water immersion or air objectives.

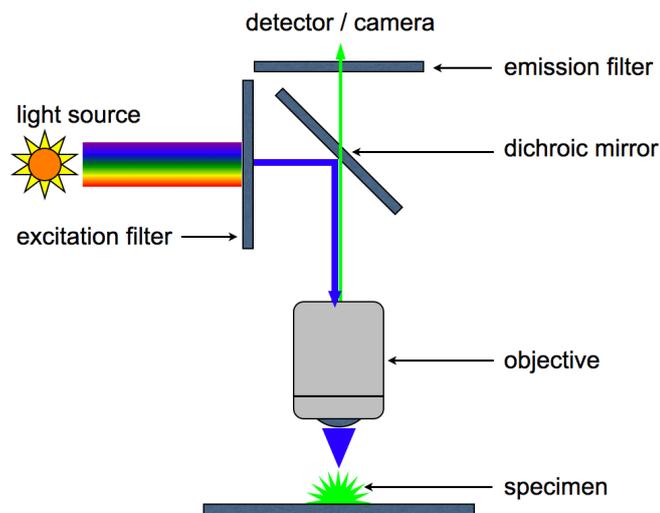


Figure 1.4: Schematic representation of the principle of fluorescence microscopy.

1.2.2 Video and Confocal Microscopy

Modern image processing and analysis approaches require the image to be present in digital form consisting of pixels (see Section 1.2.3). Although electron mi-

croscopy enables the acquisition of high-resolution images, sample preparation methods such as cryofixation or freeze-fracture do not allow live cell imaging and therefore disable studying the dynamics of a cell.

Recording a dynamic cellular event in real-time can be achieved by attaching a video camera to a microscope. Depending on the acquisition frequency, even fast cellular processes can be captured. Video microscopy can be combined with confocal microscopy. A confocal pinhole placed in front of the detector / camera rejects out of focus light and thus only allows light from the focal plane of the objective to be captured. Hence, individual planes of densely packed structures or thick objects can be imaged. This is referred to as optical sectioning. With the aid of image processing software, collections of planes can be realigned and rendered in order to generate a real three dimensional (3D) image with a high depth of focus. This technique gives insight into the morphology and the dynamic of single cells as well as of united cell structures.

1.2.3 The Digital Image

At the very beginning of microscopy, the first images of objects examined under the microscope were detailed drawings, made by hand. Nowadays, solid state cameras and charge coupled device (CCD) cameras attached to the microscope are able to record single images or a fast sequence of images, which can be combined to a movie displaying a certain progression over time. A two dimensional (2D) digital image is spanning in x and y direction. It consists of $x \times y$ individual picture elements, which are called pixels⁴. Each pixel has a discrete value, representing its intensity. Dark pixels have a low intensity, while bright pixels have a high intensity. The information content of the image is encoded by these discrete values, which also represent the starting point of most image processing and analysis software. These values are also called grey values. The number of different grey values a pixel can obtain is called the bit-depth. Usually, the human eye can distinguish

⁴The 3D analogue is called volume element (voxel). The dimensionality of an image has almost no limits. For instance, a 3D volume (x, y, z) observed over time t has the four dimensions (x, y, z, t) . If the volume is observed at different colour channels C , the image has five dimensions (x, y, z, t, C) , as it is common in multi-dimensional imaging.

50 different grey values [28]. Thus, for visual inspection, a bit-depth of 6 bit ($2^6 = 64$) is sufficient. For numerical operations, the bit-depth should be higher. Typical values are 8, 12, 16, 24 or 48 bit [28].

Usually, the image coming from the camera is uncompressed, meaning it is obtained without a loss of quality. This is important in order to avoid potential artefacts from compression, which might falsify the result. A major drawback of uncompressed image data is its size in terms of disc space it requires if stored as a file. Considering high-throughput screening or high-content screening (see Section 1.3), the amount of image data written to the hard disc can require multiple gigabytes per experiment. Thus, lossless compression algorithms are required, particularly with regard to permanent data storage.

1.2.4 Live Cell Imaging

Understanding the processes and functions which occur inside a living cell is the first step, if one wants to tackle a medical problem on the cellular level. Examining living cells is different from examining fixed samples. Living cells have to be treated with caution in order to keep them alive. Providing quasi-physiological conditions such as a specific temperature or pH minimises the risk of unwanted cell death or non-interpretable results due to improper conditions of the cell's environment.

Another important point is the application of adequate dyes. In the case of cardiomyocytes, one has found out that spontaneous release of calcium from the SR is a trigger for contractile waves [14,15,26]. Therefore, special fluorescent dyes that are sensitive to calcium such as Indo-1 or Fura-2 are used.

To obtain statistically reliable results, more than one cell has to be taken into consideration. At the same time, it can be quite challenging to find suitable cells. Although treated with care, cells can die or may be rejected because they are not active enough or too small. Furthermore, to obtain objective results a biased cell selection should be avoided. This suggests an automated process, allowing to get as much information as possible out of suited cells, in order to save both, time and resources.

1.3 Image-based High-Content Screening

Manual inspection of cells and evaluation of imaging data by human experts is both, time-consuming and cost-intensive. Furthermore, the results of the evaluation are potentially error-prone to a certain degree. This may happen due to a biased cell selection or if the number of examined cells is too low. In order to overcome these hurdles, the need for an automation of these processes arose. To be feasible, an automated approach has to meet some requirements [63]. It should allow to examine cells individually in a rapid and reproducible manner. Classification of cells according to predefined parameters as well as storage and combination of acquired data should be possible. In principle, the entire process should be faster than its manual pendant, without a decrease in its quality.

1.3.1 High-Throughput Screening

The request for automated screening of large chemical libraries in the pharmacological and biotechnological field was the main driving force for the development of high-throughput screening (HTS). Accompanied by the development of faster computer systems in the mid-1990s, HTS has been developed for cost and time reduction during the identification of lead structures in drug discovery [49]. HTS allows the screening of 10.000–100.000 compounds per day, reading out a single data point per compound [45]. These measurements are referred to as end-point measurements. When the HTS rate reaches or even exceeds 100.000 compounds per day, this is also referred to as ultra high-throughput screening. This high number of compounds is necessary, since only very few compounds lead to a hit. Usually, second-screening or follow-up screening runs are performed on these hits, leading, if at all, to even fewer hits. Indeed, the pyramid in the drug discovery cascade has a very wide base and a small tip: in order to obtain one successful outcome, usually the screening of 1.000.000 compounds is necessary [45]. Due to the large number of compounds, a single HTS run can last from days to weeks.

HTS can also be applied to cell-based assays. To get reproducible and thus reliable results, usually cell lines are applied (also see Section 1.3.3). While only

end-points are generated, the corresponding trajectories remain unclear. To detect this additional information, high-content screening (HCS), an approach which came up in the late 1990s, can be used.

1.3.2 High-Content Screening

Although HCS sometimes is described as the evolution of HTS [32], a more parallel history of development has to be taken into account. While HTS has been developed initially to automate manual laboratory processes, HCS has evolved from light microscope imaging methods [120]. The main difference between HTS and HCS lies in the way data is acquired. While HTS is usually used to read out a single data quantity, HCS allows the rapid acquisition of spatial and temporal information in parallel and thus provides an insight into the morphology as well as into biochemical dynamics on the cellular and subcellular level. Furthermore, HCS allows the physical (i.e. electrical) and chemical (i.e. application of biochemical substances like hormones or drug molecules) stimulation of cells during the measurement. Typical components of an HCS system thus comprise automated microscopes and robotic systems. The latter are used for assay preparation, liquid handling and transport of the microtitre plates. Images are recorded with CCD cameras and stored on a computer. Once the data has been acquired and is available in digital form, numerical operations can be carried out on the data, in order to access the information content of the images. This can either be done online, during the measurement, or offline, after data acquisition. Analysis can be carried out on single cells as well as on a selected subpopulation [49]. The different kinds of data that are generated in HCS are described in Section 1.4. The principle of an HCS cycle and its subdivision into a data acquisition, analysis and archiving part is shown in Figure 1.5.

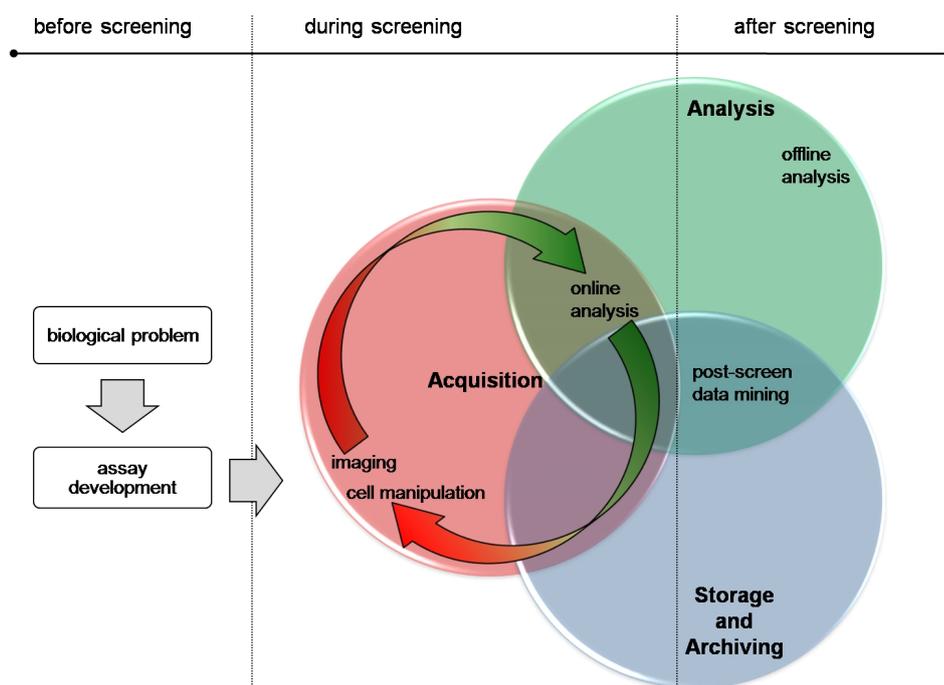


Figure 1.5: Scheme of an HCS procedure. Before the actual screening run, an HCS assay suitable to answer the biological problem has to be developed. During the screening run, acquired images are evaluated online, in order to take influence on the following screening steps. Acquired images are stored for post-screen analysis and archived for post-screen data mining.

1.3.3 Screening of Primary Cultured Cells

Compared to primary cultured cells, cell lines provide the following advantages: (i) they are easy to handle; (ii) genetic manipulation is easy; (iii) cell-to-cell variability is low. But cell lines are often virally transformed and their genotype and phenotype may be far away from their physiological counterpart. Therefore, primary cultured cells experience an increasing popularity in clinical and basic research [40]. Although they closer resemble their *in vivo* counterpart, the use of primary cultured cells is accompanied by several challenges: (a) most cell types are difficult to maintain in culture; (b) some cell types cannot grow *in vitro*; (c) during their time in culture, most cell types tend to dedifferentiate; (d) potential donor-

dependent variations in quality of the cells may occur. Due to these problems, HCS is hardly used together with primary cultured cells.

1.4 Data

Image-based HCS systems produce huge amounts and different types of data [44]. Spatial (x -, y - and z -direction) and temporal resolution of a sample recorded at different spectral channels creates a five-dimensional (5D) image, which nowadays constitutes the basis of most modern imaging experiments [112]. Such multi-dimensional images can be large with respect to their data volume and their information content may be related to metadata describing the context. Thus, sophisticated data management strategies are required. But before suited strategies can be developed, the different types of data as well as the typical steps during image acquisition, analysis and storage have to be identified. The flowchart in Figure 1.6 illustrates these steps. After an image has been acquired, it is stored on a file system. Application of image processing tools may transform the original image to a processed image, which again can be stored together with the results arising from quantitative analysis. During the whole process, the data volume can easily double or triple. The different types of data will be explained in the following subsections.

1.4.1 Raw Image Data

The definition of the term raw image is variable, since a developer of image acquisition hardware may have another point of view than a developer of imaging software. For me, images that are read out from the chip of a CCD camera and stored on a file system without any further processing are referred to as raw images. Hereby, I neglect the fact that a first processing may already occur on the hardware side (i. e. CCD chip and camera electronics).

Numerous raw image formats are in use. Most of them are not compatible with each other, because besides the actual image data, metadata such as light intensity, exposure settings, date and time, camera and objective model and various other

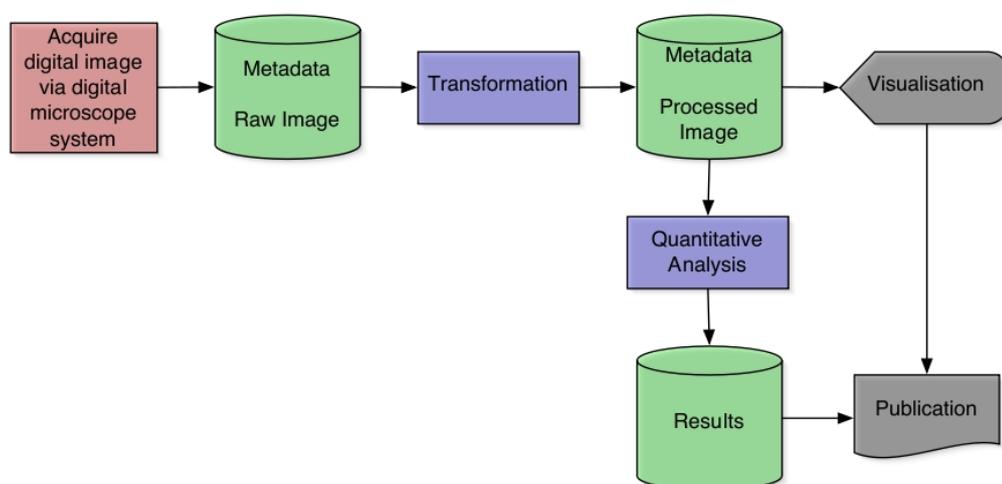


Figure 1.6: Flowchart of the typical steps involved in image-based HCS data generation (red: acquisition, blue: processing and analysis, green: storage and archiving, grey: data output).

information can be stored. Parameters of a raw image such as contrast, sharpness or compression ratio can be controlled during conversion into another file format. Depending on the output file format, the conversion can be done without any loss of information. This is especially useful during dark frame subtraction, removal of systematic noise and image compression, which otherwise may lead to unwanted artefacts in the resulting image file. A widely used file format is the Tagged Image File Format (TIFF)⁵. It allows the storing of multiple images within one file (multi-page TIFF) as well as the corresponding metadata. Furthermore, TIFF images may be compressed using lossless compression algorithms.

A lossless compression is important in two senses: (i) the loss of information could falsify the result of the data analysis; (ii) uncompressed images are larger with respect to data volume. This may lead to longer data transfer times and data management may thus become inconvenient. Since most raw image formats are proprietary, commercial / system specific imaging software is required in order

⁵TIFF was originally created by the company Aldus (Seattle, USA), which has been acquired by Adobe Systems (San Jose, USA).

to allow further processing of raw images. This often constrains the flexibility in freely choosing appropriate image processing software. Furthermore, most proprietary raw image formats are undocumented on purpose. Although in some fields standardised data formats, like DICOM [31] (for medical imaging data), FCS [52] (for flow cytometry data) and SBML [118] (for systems biology data) exist [117], up to now, no raw image standard has been established. Despite these hurdles, Bio-Formats [18], an open-source Java library for reading and writing file formats found in life sciences, has been successfully developed. The Bio-Formats library will be further addressed in Chapter 2.

1.4.2 Metadata

In general, metadata are data which contain information about other data. Considering a computer's file system, metadata can comprise information like file names, access rights, creation and modification date. In database applications, metadata might contain information about tables that are stored in the database. With respect to image data, metadata can provide descriptive and contextual information about the image such as recording conditions and settings of the imaging hardware. In addition to information originating from the imaging system, information and annotations provided by the experimenter can also be stored together with the actual image data. Both, system and user specific information is then referred to as metadata. Depending on the underlying data format and data model, metadata can be either stored internally or externally. Internal metadata is stored within the same file as the actual image data. External metadata is stored separately and then linked to the image data. An example of the latter case may be a protocol file containing the algorithm of an image acquisition procedure. This protocol file can be attached to the image file as external metadata. In both cases, the syntax of the metadata is often described by a markup language such as the extensible markup language (XML) [37]. XML enables a hierarchical and structured representation of metadata. Since this representation allows the use of text-based tags, XML can be read and interpreted by humans as well as by computers. The use of metadata alleviates understanding and management of the image data. Examples

of projects, that are developing specifications and tools for defining experimental metadata are OBO [125] (biomedical ontologies), MGED [90] (ontology for describing samples used in microarray experiments), MIFISHIE [79] (specification for in situ hybridisation and immunohistochemistry experiments) and MIACA [78] (information guideline and modular cellular assay model) [44].

Since metadata enables filtering of the content of a large image set by searching for key words, browsing image data can be tremendously sped up. Due to the above mentioned amenities of TIFF images, the Open Microscopy Environment (OME) [126] consortium has developed an extension of the TIFF specification, called OME-TIFF. Metadata is described using an extension of XML called OME-XML. Together, both data formats are applied in their server and client software which is called OME-Remote Objects (OMERO). OMERO, which will be further described in Chapter 2, allows visualisation, management and annotation of microscope images and metadata.

1.4.3 Experimental Results and Data Management

Analysis is predominantly performed on raw image data, in order to avoid potential artefacts from processed images. The resulting data can either be numerical data such as statistical values or again image data, which might be modified in a sense, that the structures of interest are emphasised in order to make them analysable. This adds to the already existing data volume originating from the aforementioned data types and imposes additional challenges on data storage and archiving strategies.

Data handling and consistency

The different types of related data have to be stored to allow retrieving data without losing the relationship. Therefore, an appropriate syntactic data model, which describes the relation between different data is required.

Data storage and archiving

When working with imaging data, two requirements have to be reconciled amongst others: fast data access and long-term archiving. During the data acquisition and evaluation period, fast handling of image data is essential. Therefore, modern imaging software keeps the image data in the computer's random access memory (RAM) during processing. But since a computer's RAM is volatile, the data would be lost after switching off the computer. Hence, for persistent short-term storage data is written to either the computer's built in hard disc or to an external redundant array of independent discs (RAID). Both storage media allow reasonably fast retrieval of data. Once the evaluation period has been completed, the need for fast data access no longer exists. At the same time, the need for free disc space for new experimental data arises. Since the data has to be kept for reasons of validation and reproducibility and, moreover, since it still might be valuable for future analyses, it has to be archived. Furthermore, scientists are required by law to archive their data for a period of ten years using durable and safe data media. Therefore, the data may be transferred from the RAID to a tape, for instance. Although tape access is slow, tapes are more economic than server hard discs usually utilised in a RAID system. Figure 1.7 outlines the differences between the aforementioned storage media with respect to their specific access times and their suitability for long-term data storage. In addition to storage space, a database management system is required to maintain and utilise large collections of data [100]. Since image-based HCS produces large amounts of data, data management by manually maintaining files and folders on a file system becomes impossible. A relational database system can be used to preserve relations between linked data. Furthermore, it enables the combination of search criteria and thus efficient querying of the data.

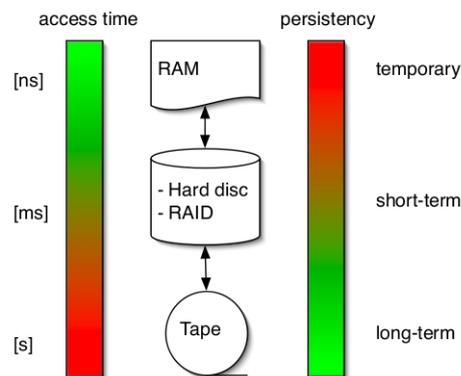


Figure 1.7: Different types of storage media with regard to access time and persistency (green: good, red: bad). Arrows denote the possible dataflow.

1.5 Objectives of the Study

Understanding the causes and effects of AF on the cellular level may support the development of effective pharmacological disease management. But until now, *in vivo* studies of single cardiomyocytes are not possible due to the movement of the entire heart and due to a lack of sophisticated examination methods. The investigation of isolated primary cultured cardiomyocytes represents an alternative, since properties of primary cells are close to their *in vivo* counterpart. Furthermore, isolated cells can be handled applying standardised imaging techniques such as confocal microscopy. Cellular events, such as calcium signalling cascades during excitation-contraction coupling, can be recorded and analysed utilising HCS.

Depending on the configuration, prices for commercial HCS systems range from less than €125.000 to about €650.000 [44]. Mainly developed for the pharmaceutical industry, these systems are mostly tailored to a special problem. But since biological questions in life science research often change, commercial HCS systems are not always practical. In the academic field, an HCS system has to be flexible as well as expandable. Due to the mentioned requirements, the development of an open, modular HCS system suitable for primary cultured cells was aspired. Therefore, my thesis covers the following goals:

- development of hardware and software for electrical and chemical / hormonal stimulation of isolated primary cultured cardiomyocytes
- implementation and programming of software for management, storage and archiving of image data
- programming of image processing and image analysis algorithms for automated cell detection and calcium imaging
- programming of software for inter-process communication and distributed computing

Altogether, the development of an automated HCS system for high-resolution image acquisition of primary cultured cells was achieved as a proof of concept.

Chapter 2

Materials and Methods

2.1 Hardware for Electrical Field Stimulation

2.1.1 HCS Stimulation Plate

The HCS stimulation plate can be divided into two components: the multi-well plate and the lid (see Figure 2.1(a)). Its task was to provide multiple “test tubes”, which enabled HCS of adult primary cultured cardiomyocytes. These test tubes are referred to as compartments or wells. To adapt commercial designs of multi-well plates for the prerequisites of live cell imaging, various design modifications had to be implemented.

Size and dimensions of the multi-well plate

One of the major goals of this development was to design HCS hardware that is compatible with live cell imaging. Moreover, to ensure seamless integration of commercially available standard hardware, it was an imperative to adapt the dimensions of our multi-well plate to fit the industry standards (for details see [1–4]).

In addition to the outer dimensions of the multi-well plate, we also optimised the particular design of each well. Usually, these wells have a spherical geometry that makes it difficult to incorporate standard electrodes for stimulation of cardiomyocytes. We thus redesigned the individual well for a maximised area between

the electrodes, that ensures homogeneous electric fields. The result was a rectangular design. In the end, we fitted 24 rectangular wells into a standard multi-well plate. As described in the introduction, our HCS approach requires high-resolution image acquisition that cannot be performed with air objectives. Immersion objectives are an essential requirement. Standard plastic bottoms do not permit the use of high NA objectives. We therefore used a specialised foil with optical properties that matched those of glass coverslips without the fragility of the glass material. The dimensions and the arrangement of the wells are shown in Figure 2.1(b).

Properties of the body of the lid

Acquisition of high-resolution images of cells requires an even illumination of the imaging area. To enable recordings with transmitted light, the body of the lid had to be transparent. Furthermore, for sterile handling, the material had to be autoclavable. Since polycarbonate is heat resistant to at least 130 °C, transparent and easy to handle, it was used for the body of the lid.

Properties of the electrodes

Two electrodes per well served to conduct the stimulation current to the sample. Since sterile handling is essential in order to avoid contamination of the cells during long-term experiments, the electrodes as well as all parts of the electrical circuit in the lid had to be autoclavable. The electrodes themselves had to be chemically inert and should not release substances that are poisonous to the cells. Therefore, we used customised carbon electrodes (Schunk Kohlenstofftechnik GmbH [111], Heuchelheim, Germany) with a size of 25 × 15 × 1 mm. The distance between two electrodes was 12 mm. The distance between the electrodes and the wall of the well was at least 1 mm in order to minimise capillary forces in between.

Conducting paths

In an early version of the lid, each pair of electrodes was individually wired by using a plastic coated platinum wire, which was manually soldered to a 50 pin

ribbon cable interface. Constant use and repeated autoclavation cycles required a scheduled service of the soldering points and the wires. In order to increase the ease of maintenance of the lid, the plastic coated platinum wires were replaced by conducting paths consisting of a titanium-gold alloy (Naturwissenschaftliches und Medizinisches Institut [85], Reutlingen, Germany). They can be printed directly onto the polycarbonate material. Furthermore, printed paths can be better protected against mechanical damage. These paths connected each single electrode to a standard cable connector, thus enabling stimulation protocols for each individual well. The pin assignment of the connector is shown in Table B.3 (see appendix). Pins 1 and 2 were connected to electrical mass, all other pins were assigned to wells pair-wise. Figure 2.1(c) demonstrates how electronic jumpers can be used to combine any number of wells by using jumpers in the socket of the ribbon cable. This is especially useful if the number of available outputs on the control hardware is limited.

Electrical isolation

To protect the conducting paths against mechanical damage, moisture and short-circuits, the polycarbonate lid has been electrically isolated using the transparent acrylic resin Cramolin Plastik (ITW Chemische Produkte GmbH & Co. KG [56], Mühlacker, Germany). This protective paint is resistant to elevated temperatures up to 100–120 °C that are necessary for autoclavation.

Mounting of carbon electrodes

The carbon electrodes were electrically connected to the conducting paths by custom made clamps consisting of a stainless steel foil with a thickness of 0.1 mm. The clamps were fixed onto the lid by using sheet-metal screws.

Plastic foil

The plastic foil, which was bonded to the bottom of the 24-well plate without using glue, is available from ibidi GmbH [50] (Martinsried, Germany). The additional

oleophobic coating of the bottom of the foil minimised adhesion between the foil and the oil drop on top of the microscope's objective. This minimised putative loss of oil attached to the foil while moving the multi-well plate during the screening process (for details, see [80]).

Liquid handling

Cardiomyocytes can be electrically stimulated by an electric design as described above. Besides this, HCS often requires specific liquid handling steps such as application of test substances or constant flow of extracellular medium. To allow both experimental interventions, the lid offers two different paths for access.

For sole substance application, there is a central opening in the lid with a diameter of 3 mm for each well. These openings can be covered either by a Parafilm® (Pechiney Plastic Packaging Company [93], Chicago, USA) clamped on top of the lid or by an adhesive sealing film. Access by a pipette is possible by film penetration or removal, respectively.

In order to perform constant perfusion of individual wells, two pairs of 1 mm holes in the lid have been added. This allows to enable a laminar flow at the bottom of the well and thus permits simultaneous liquid addition and removal.

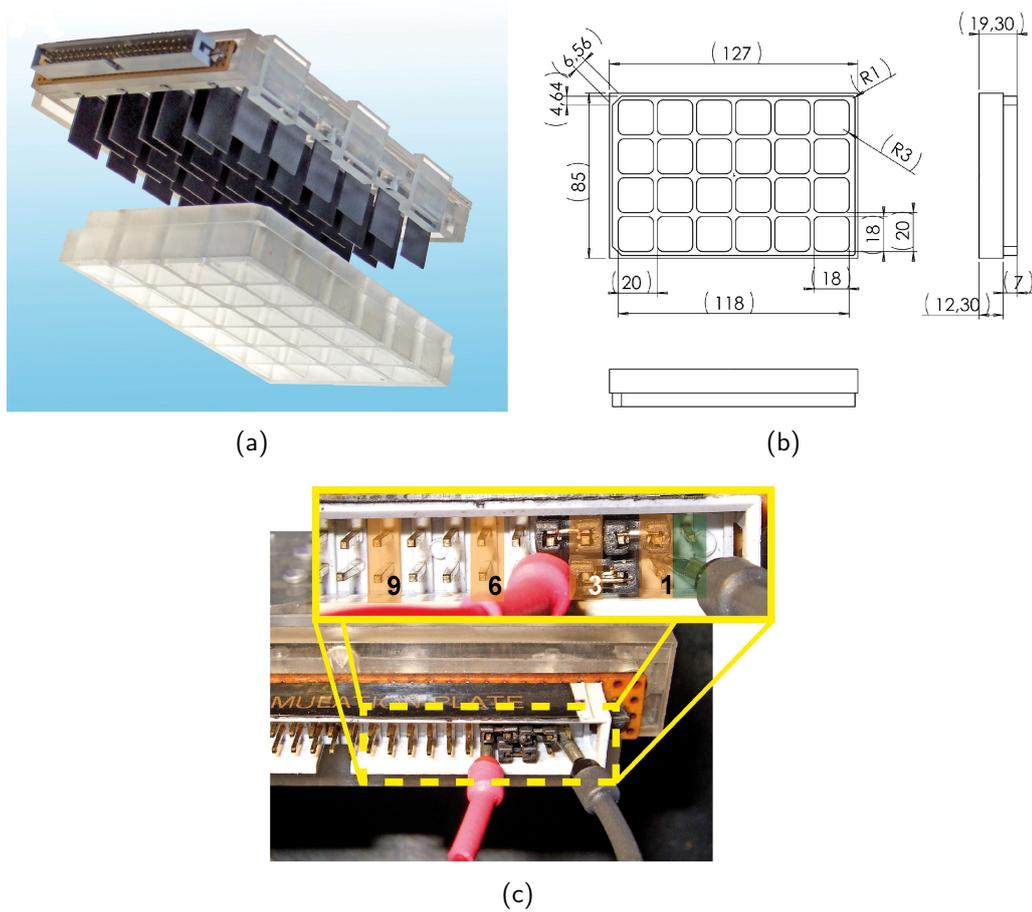


Figure 2.1: The HCS stimulation plate. (a) The lid, containing the carbon electrodes (top) and the 24-well plate (bottom). (b) Technical drawing displaying the size and dimensions of the 24-well plate. (c) The insert shows that three jumpers are used to connect a column of four wells. The green transparent block marks pins connected to the electrical mass, while the orange blocks display examples for single well contacts and are labelled with corresponding numbers.

2.1.2 Data Acquisition Boards for Digital Input / Output

DT335 data acquisition board

Initially, a digital input / output (I/O) card (DT335, Datatranslation [29], Marlboro, USA), which was available in our laboratory, was used. This card provided 32 digital I/O lines for controlling laboratory equipment. The DT335 had a peripheral component interconnect (PCI) interface. Connection to an external connector block was realised by a 68-pin connector. The maximum output of a digital I/O channel was 5 V with transistor-transistor logic (TTL) level.

NI PCI-6229 multifunction data acquisition board

As described above, the HCS stimulation plate consisted of 24 wells, each containing two carbon electrodes. In order to avoid accumulation of electrolytic products at the electrodes, the polarity of consecutive pulses was alternated. To independently control the 24 wells, 48 digital trigger lines in total were required. Additionally, an analogue line for external control of the stimulation voltage was needed. The digital I/O card NI PCI-6229 (National Instruments [81], Austin, USA) met our requirements. It also had a PCI interface. Supported operating systems were Microsoft Windows [77], Linux and Mac OS [7]. The NI PCI-6229 was able to make use of real-time capabilities of the underlying operating system. It was equipped with four 16-bit analogue outputs, 48 digital I/O channels and 32-bit counters and timers for digital triggering. The analogue outputs had a maximum voltage range from -10 V to $+10$ V, which allowed for externally controlling the dual power supply described in Section 2.1.5. The digital I/O channels had a maximum I/O level of $+5$ V (TTL). Thus, they were suitable to control the solid state relays described in Section 2.1.4.

Each of the two connector interfaces was plugged to a connector block (NI SCC-68, National Instruments) using a shielded 2 m cable (NI SHC68-68-EPM, National Instruments).

2.1.3 I/O Connector Blocks

A major requirement of the electric design was to independently provide alternating pulses (as described in Section 2.1.2) to each individual well. This decoupling of the wells was necessary to enable concurrent application of different stimulation modes inside multiple wells. Therefore, the electrical current had to be continuously switched for each well, resulting in individual alternating pulse trains. Switching was implemented by using solid state relays (SSR's) as described in Section 2.1.4. In order to conduct the triggers from the digital I/O-card described in the previous section to the SSR's, two NI SCC-68 (National Instruments) connector blocks were required. The NI SCC-68 featured a general breadboard area for I/O signal connection.

The pin assignments of the two connector blocks are shown in Figure 2.2. Eight digital I/O channels and 16 programmable function interfaces (PFI's) of connector 0 were used to conduct the triggers for the positive pulse. The PFI's were set to digital output, thus connector 0 provided 24 digital output channels in total. Additionally, one analogue output of connector 0 was used for external voltage control of the dual power supply (see Section 2.1.5). 24 digital output channels of connector 1 were used to conduct the triggers for the negative pulse.

2.1.4 Solid State Relays

Electrical stimulation of cardiomyocytes typically is done by applying short electrical pulses with a pulse width of about 5 ms. To generate pulses with a pulse width in this range, relays with short switch-on and switch-off times were required. Furthermore, a low leakage current was fundamental, in order to avoid electrolytic products which might harm the cells. These properties were best met by SSR's. Since an SSR does not have any moving parts, short cycle times are possible. In an SSR switching is implemented by using an optocoupler, which eliminates attrition and thus accounts for long-term reliability. The pins for control and load current are galvanically separated, which adds to the low leakage current. Table 2.1 contains information about three different relays that have been taken into account for this project. All SSR's taken into consideration displayed the required properties but showed differences as detailed below.

Table 2.1: Summary of characteristics of three different types of solid state relays.

Vendor	COSMO Electronics Corporation	OMRON Corporation	Panasonic Corporation
Type	KAQY212s	G3DZ	AQY225R1S
T_{on} (max)	1.5 ms	6.0 ms	0.75 ms
T_{off} (max)	1.5 ms	10.0 ms	0.2 ms
Leakage current (max)	10^{-6} A	10^{-5} A	10^{-8} A
Load voltage (max)	60 V	100 V	80 V
Load current (max)	400 mA	600 mA	350 mA
Minimum order quantity	2000	1	1
Cost per relay excl. VAT	€0,50	€12,50	€5,90

Due to the minimum order quantity of 2000 units of the SSR type KAQY212s (COSMO Electronics Corporation [27], Taipei Hsien, Taiwan), it was not feasible to test this type of relay. Figure 2.3 shows voltage traces of the relay types G3DZ (OMRON Corporation [89], Kyoto, Japan) and AQY225R1S (Panasonic Corporation [92], Osaka, Japan). The signals were recorded using a four channel digital storage oscilloscope (TDS 2024B, Tektronix [121], Beaverton, USA). Its

front-panel USB host port allowed exporting displayed waveforms. As can be seen in Table 2.1, the relay type AQY225R1S was approximately one order of magnitude better than the relay type G3DZ. This held true for both, the switching times (verified by the output of the oscilloscope) and the leakage current. In conclusion, the relay type AQY225R1S was chosen for the HCS platform.

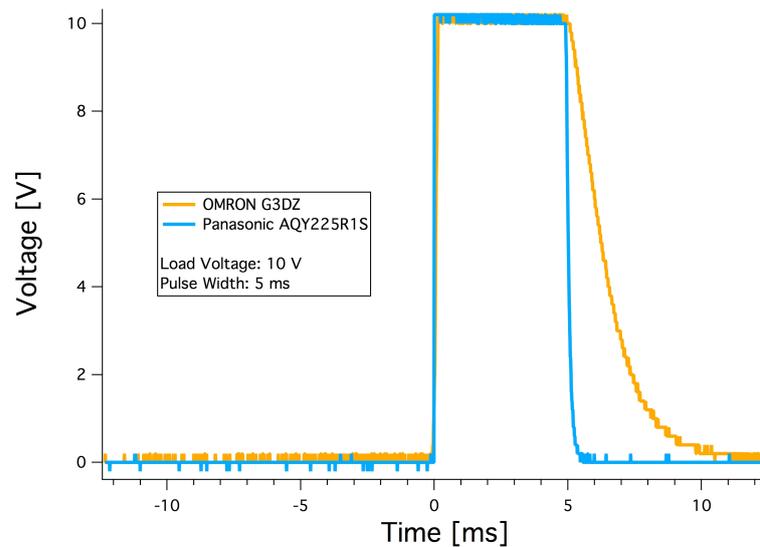


Figure 2.3: Voltage traces of the relay types OMRON G3DZ (orange) and Panasonic AQY225R1S (cyan). In both cases, the load voltage was set to 10 V direct current. As can be seen, both relay types were able to generate an electrical pulse with a width of 5 ms. Due to faster switching times, rising and decay phase of the Panasonic AQY225R1S relay are steeper than of the OMRON G3DZ relay. This results in a more rectangular shape of the trigger signal and thus allows for a more precise rectangular pulse generation.

2.1.5 Voltage Source

For pulse trains with alternating negative / positive currents, the stabilised dual power supply TOE 8732-2 (TOELLNER Electronic Instrumente GmbH [129], Herdecke, Germany) was chosen. Its ten-turn potentiometers allowed precise adjustment of voltage and current values with very good reproducibility. Alternatively,

current and voltage values could be controlled externally, by using an analogue standard industrial control voltage of 0–10 V, which translated to 0–32 V and 0–1 A per output. In order to control the two outputs of the TOE 8732-2 concurrently and to thus make the stimulation voltage software-adaptable, the tracking mode could be used. Table 2.2 shows a summary of the specification of the TOE 8732-2.

Table 2.2: Summary of characteristics of the dual power supply TOE 8732-2.

Output voltage		$2 \times 0\text{--}32 \text{ V}$
Setting resolution		2×10^{-4}
Output current		$2 \times 0\text{--}1 \text{ A}$
Setting resolution		4×10^{-4}
External voltage and current control	Input resistance	10 k Ω
	Control voltage	0–10 V (for 0–32 V / 0–1 A output)
Operating temperature		0–35 °C

2.2 Hardware for Application of Test Substances

The perfusion system was composed of a mounting plate holding the reservoirs and pinch valves and the control unit.

2.2.1 Mounting Plate: Pinch Valves

The mounting plate consisted of a polyvinyl carbonate plate with milled openings for 24 reservoirs and 24 pinch valves (type 075P2NC, Bio-Chem Fluidics [17], Boonton, USA).

2.2.2 Control Unit: Universal Serial Bus Relay Modules

The core of the control unit comprised three universal serial bus (USB) relay modules (type USBREL8, Quancom Informationssysteme GmbH [99], Wesseling, Germany), each containing eight decoupled dual in-line package relays that were

controllable by the Quancom driver library QLIB. This library provided the interface to the programming environment LabVIEW (see Section 2.5.1). The three USB modules were connected to one USB port of a personal computer (PC) via an USB hub.

2.3 Hardware for Image Data Acquisition

For image data acquisition, the automated light microscopy platform more™ (TILL Photonics [128], Gräfelfing, Germany) was used. This software controlled scientific imaging platform was based on a modular architecture and thus permitted the adaption of the system to all major fluorescence methods applied in live cell imaging. The different components used for this project are depicted in Figure 2.4 and described in the following.

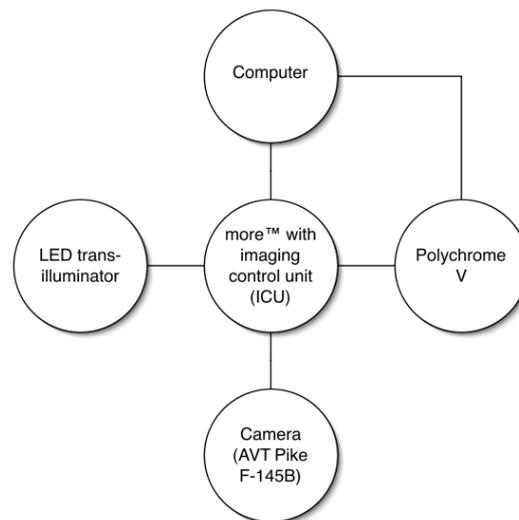


Figure 2.4: more™ connection scheme of the different components.

2.3.1 Automated Light Microscopy Platform more™

The core of the imaging platform was the computer controlled digital microscope more™. Applying the components shown in Figure 2.4, the microscope could be used to perform epifluorescence imaging, Förster (also: fluorescence) resonance energy transfer (commonly referred to as FRET) ratio imaging and time-lapse studies (3D / 4D imaging) in an automated manner. Figure 2.5 shows a schematic drawing of the more™. Its different components are described below.

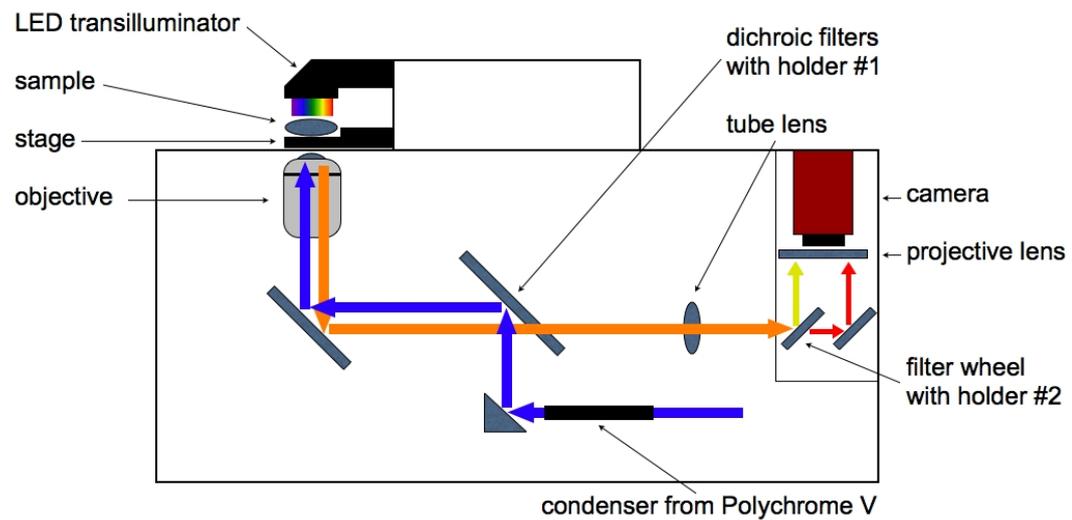


Figure 2.5: Schematic drawing of the digital microscope more™. The figure shows the optical path for light-emitting diode (LED)- and Polychrome V illumination as well as the location of the different components inside the microscope. A distortion free tube lens guides the fluorescence emission light from the microscope frame towards the entrance aperture. The light beam is split into two colours and finally detected by the camera.

Stage

The more™ was equipped with a stage holder for microtitre plates. This vibration damping optical table minimised oscillations that occur during the movement of the stage and thus contributed to more stable image recordings. The X/Y-movement

(speed (max): 10 mm/s; acceleration (max): 1000 mm/s²) was software controlled.

Objectives

The more™ provided a software controlled objective changer with a capacity of four objectives. Z-movement of the objectives was performed by a voice coil motor, a device which combined the advantages of mechanical Z-drives (large travelling distance) and piezo-Z stages (high precision). Our system was adjusted for a maximal movement of 9.49 mm along the Z-axis with nanometer resolution. Table 2.3 depicts the three available objectives.

Table 2.3: more™: equipment of our objective changer.

position	objective
1	10/0.25 air (Olympus PlanC N)
2	60/1.49 oil (Olympus Apo N)
3	20/0.85 oil (Olympus UPlanSApo)
4	empty

Filters

The acquisition of calcium signals as depicted in Figure 1.3 requires the loading of the cells with a fluorescent dye. To enable the detection of such fluorescence signals, our system contained filter-sets for calcium imaging with Fluo-4 / Fura-2 as well as for dual emission Indo-1 imaging.

Camera

Images were captured by a camera (Pike F-145B, Allied Vision Technologies GmbH (AVT) [6], Stadtroda, Germany). This monochrome CCD camera offered a resolution of 1.45 megapixels (1388 × 1038 pixels) at an acquisition rate of 30 frames per second.

LED transilluminator

The LED transilluminator was a multi-LED array with red, green and blue (RGB) monochrome LED's. Since the LED's were gateable and dimmable, each colour within the RGB colour model could be generated by additive colour mixing.

2.3.2 Polychrome V

A second light source for the more™ was the Polychrome V (TILL Photonics). This ultra fast switching monochromator could be used for fluorescence ratio imaging. It allowed a variable wave length selection from 320–680 nm by using a galvanometer driven defraction grating.

2.3.3 Imaging Control Unit

The imaging control unit (ICU) was the main interface between the PC and the more™. It translated protocols executed on the PC into commands that were usable by the microscope. This kind of mediator allowed high time resolution and real-time protocol capability independently of the workload of the PC. For communication, the ICU provided digital I/O interfaces as well as analogue output interfaces.

2.4 Hardware for Accessing, Storing and Archiving Data

Reliability and redundancy are two major requirements of computer hardware involved in data storage. Since most systems have to run 24 hours a day, 7 days a week, a high level of quality was essential. Therefore, the hardware for accessing, storing and archiving data has been purchased from the well-known International Business Machines Corporation (IBM [51], Armonk, USA). The different devices and their specifications are listed below.

2.4.1 Servers

Table 2.4: Summary of specifications of the eServer and the pServer.

IBM xSeries 346 (eServer)	Hard discs Number of processors Processor type RAM	3 × 73,4 GB arranged as RAID 1 + 1 hot spare disc 4 Intel® Xeon™ central processing unit (CPU) 3.20 GHz (64-bit) 4 GB
IBM System P5 (pServer)	Hard discs Number of processors Processor type RAM	2 × 73,4 GB + 6 × 300 GB 2 IBM POWER5 CPU 1.9 GHz (64-bit) 4 GB

2.4.2 Disc Shelves

Table 2.5: Summary of specifications of the disc shelves.

IBM DS4100 Midrange Disc System	Hard discs	14 × 400 GB
IBM DS4000 EXP100 Storage Expansion Unit	Hard discs (2 units)	2 × 14 × 400 GB
Total disc space		3 × 14 × 400 GB = 16800 GB = 16,8 TB

2.4.3 Tape Library

Table 2.6: Summary of specifications of the tape library IBM Total Storage 3584-L52.

Number of tape drives	3
Tape drive type	Linear tape-open (LTO) Ultrium 3
Data throughput	80–160 megabytes per second
Current number of tapes	114 + 1 cleaning tape
Number of free slots	14
Total number of tapes	129
Tape capacity (each)	400 GB (native)
Current total tape capacity	45,6 TB
Maximal stage of expansion	1878 TB

2.5 Programming Languages and Development Environments

2.5.1 Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW)

The cross-platform graphical development environment LabVIEW [69] (National Instruments) allows the development of software in the fields of data acquisition and instrument control as well as for signal processing and analysis. The programming language used in LabVIEW is called “G”. It is a graphical dataflow programming language, where different function-nodes, represented by graphical symbols that describe programming actions, are connected by drawing lines (wires). The source code is stored as a block diagram. This is exemplified in Figure 2.6, which shows the front panel and the block diagram containing G source code.

For this thesis, LabVIEW 2009, the LabVIEW Real-Time Module (National Instruments) and the Measurement & Automation Explorer (MAX) (National Instruments) were used. MAX is a tool, which allows the configuration e.g. of a general

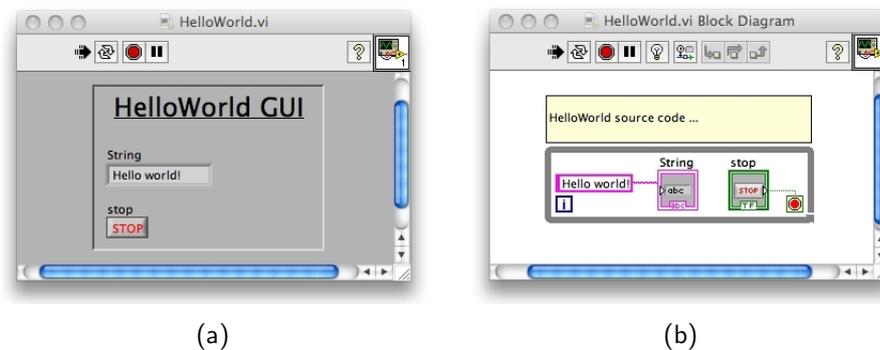


Figure 2.6: Example program written in LabVIEW. (a) Shows the front panel editor of the LabVIEW development environment. Here, the elements of the graphical user interface are arranged. (b) Shows the block diagram editor containing exemplified source code. The string constant ‘Hello world!’ is connected to an indicator with a wire (the pink line between the two elements). The Boolean stop button is connected to the loop condition of the while loop (grey structure) with a green wire. If the loop condition receives the Boolean value TRUE, the while loop terminates and the program exits.

purpose interface bus (GPIB) interface. The data acquisition board NI PCI-6229 (see Section 2.1.2) was found on the GPIB bus and configured with MAX.

The reasons for choosing LabVIEW as the development environment are outlined below. Although LabVIEW is very complex and powerful, it allows for rapid prototyping as well as for developing graphical user interfaces (GUI's). This enables programmers to easily implement their ideas, leading to a reduced development time. LabVIEW allows to access instrumentation hardware, which can be controlled directly from the GUI. A LabVIEW program is referred to as a virtual instrument (VI). Stand-alone programs can be distributed without the need of a LabVIEW installation. Only the provided runtime engine has to be installed. Instrumentation hardware can be software-simulated. This allows VI development independently of the actual data acquisition hardware. With this, software maintenance and development can coincide with experimental work using the control and acquisition hardware. Furthermore, hardware components do not have to be moved from the measurement computer to the software development site. The included compiler generates native code for the underlying CPU platform, which ensures fast

execution speed. LabVIEW programs can also use multithreading and real-time capabilities. From version 8.2 onwards, LabVIEW supports object-oriented programming. This feature alleviates modular programming. Interestingly, LabVIEW supports math nodes, which are compatible with MATLAB (see Section 2.5.7), a widespread numerical computing environment. LabVIEW was used to control the NI PCI-6229 digital I/O-card, which is described in Section 2.1.2, as well as the three USB modules described in Section 2.2.2.

Producer-consumer design pattern

Since LabVIEW supports multithreading – the ability of a process to fork into two or more tasks that are running concurrently – techniques enabling synchronisation of multiple processes had to be developed. One of these techniques, the producer-consumer design pattern [82], was applied to LabVIEW programs presented in this thesis and is described below.

While one task, such as a loop, may be responsible for producing and providing data, a second loop may depend on this data and require the provided information for further processing. Thereby, the second loop consumes the data that is produced by the first loop. Since the two loops may produce and consume data at different rates, a synchronisation may be necessary to enable an unhindered data flow. For instance, if the consumer loop processes data at a lower rate than the producer loop, the latter would have to wait until the consumer loop has finished processing, leading to a locking of the producer loop.

To avoid such potential synchronisation issues, LabVIEW provides the producer-consumer design pattern, a programming technique that allows the decoupling of two connected loops running in parallel. It uses data queues to enable communication between the two loops. Since a data queue can be considered as a data buffer between producer and consumer loops, the producer loop can write its data into the queue, while the consumer loop can take out data from the queue. Hereby, both loops can write and read data at their specific rates, thus avoiding a blocking of the producer loop.

2.5.2 LabWindows / C for Virtual Instrumentation (CVI)

The full version of the LabVIEW development environment also contains the event-driven, American National Standards Institute (ANSI) C programming environment LabWindows/CVI [70], which is commonly referred to as CVI. Since CVI can use the same libraries and data acquisition modules as LabVIEW, it can be considered as an alternative for ANSI C programmers, who want to access instrumentation hardware using text-based C source code. As a result, CVI allows the implementation of ANSI C software interfaces, which can be accessed by external software supporting ANSI C software interfaces. For this thesis, CVI 9.0 has been used. It supports Windows, Linux and real-time platforms and allows the optimisation for multi-core systems.

2.5.3 C++

C++ is a text-based programming language, developed by Bjarne Stroustrup¹ at Bell Laboratories [13] in 1979 [107]. Starting as an object-oriented enhancement of the text-based programming language C, C++ has become a self-contained programming language. In 1998, an International Organisation for Standardisation (short: ISO) and ANSI standard for C++ has been ratified [107]. One design criterion was to keep it as compatible to C as possible, thus most C source code can be made to compile correctly in C++. In contrast to G, C++ does not need a sophisticated programming environment. Various free and proprietary compilers for creating machine code on target hardware are available for nearly all platforms. For this thesis, the following C++ compilers have been used:

- Mac OS X: gcc version 4.0.1 (Apple Inc. build 5490)
- Linux: gcc version 4.1.2 20061115 (prerelease) (SUSE Linux)

¹Danish computer scientist, born in 1950.

2.5.4 Boost C++ Library

The Boost C++ Library [20] is a collection of free sub-libraries, running on Unix, Linux, Mac OS and Windows. By using the sub-library Boost.Thread², portable C++ code, which allows the execution of multiple threads with shared data was implemented. Boost.Thread has been applied in the Messaging Server software (see Section 3.5.1).

2.5.5 Java

Another text-based and object-oriented programming language is called Java. It has been developed by James Gosling³ at Sun Microsystems [115] in 1995 [108].

Although the syntax of Java is very similar to the syntax of C/C++, there are some distinct differences. For instance, Java's object-model is simpler, multiple inheritance and pointer arithmetic is not supported and memory management is automated. In contrast to C/C++ source code, Java source code is not compiled directly into machine code. Instead, it is first translated into bytecode, which is executed in a Java runtime environment (JRE) and then interpreted by the Java virtual machine (JVM). Since the JRE is available for nearly all platforms and since the JVM is integrated into the JRE, Java programs do not have to be adjusted or modified in order to run on different computer platforms. For this thesis, the following Java version has been used:

- Mac OS X: java version "1.5.0_16" / javac 1.5.0_16

2.5.6 Eclipse Integrated Development Environment

C++ and Java programming has been performed using the open source multi-language development platform Eclipse [34] developed by a free software community. Written in Java, Eclipse originally was developed as a Java programming environment. Nowadays, an integrated plug-in system allows to extend

²Author: Anthony Williams (originally William Kempf), First Release: 1.25.0.

³Canadian software developer, born in 1955.

Eclipse and enables e.g. the development of C/C++ source code. For this thesis, Eclipse 3.4 Ganymede has been used.

2.5.7 Matrix Laboratory (MATLAB)

The numerical computing environment MATLAB has been developed by Cleve Barry Moler in the late 1970s. Since 1984, it is available from The MathWorks [122] (Natick, USA), a software vendor for technical computing software. Its cross-platform capability allows the use of MATLAB scripts on computers running Microsoft Windows, Mac OS and Linux. MATLAB is mainly used for solving mathematical problems numerically. It can be used for matrix manipulation, implementation of algorithms, graphical display of results and GUI development. MATLAB scripts can be compiled and stored as executable files. The target system does not require an installed MATLAB in order to execute the scripts – installing the provided runtime engine is sufficient. The functionality of MATLAB can be enhanced by installing additional MATLAB packages called toolboxes. In this thesis, MATLAB was used for image processing and image analysis. Since our aim was to develop an automated HCS environment, MATLAB's capability of interfacing with programs developed in other programming languages allowed us to establish an interconnection between different modules of our HCS system and thus to foster automation. For this dissertation, MATLAB 2009a, the Image Processing Toolbox (The MathWorks) and the MatVis interactive visualisation toolkit [98] were used.

2.6 Software for Computer Aided Design of Circuit Boards

2.6.1 KiCad

KiCad [66] is an open source computer aided design software suite for designing electronic circuit boards. It has been developed by the French researcher Jean-Pierre Charras. KiCad allows to export data in the Gerber file format, which is

widely used by manufacturing machines for printed circuit boards. In addition to information for printing the layout of electrical connections, Gerber files also contain information for drilling and milling machines and thus enable the complete production of an electronic circuit board.

2.6.2 GC-Prevue

GC-Prevue (GraphiCode [41], Snohomish, USA) is a software for viewing and printing electronic manufacturing data. It is free of charge and standard in electronics industry. GC-Prevue was used to verify the Gerber files.

2.7 Software for Managing, Storing and Archiving Data

2.7.1 OME and OMERO

The OME consortium comprises members from academia as well as industry. Their goal is to bridge the gap between these two areas and to collaborate in order to produce open source software tools in the field of data management for biological light microscopy. Currently, the OME consortium provides the following software tools and data formats, which will be discussed below:

- OME server
- OMERO server and client software
- Bio-Formats library
- OME-XML
- OME-TIFF

OME server

The original OME server is a Perl⁴-based system for visualising, analysing and managing microscope images and their corresponding metadata. Started in the year 2000, the OME server is still being developed. It is controlled using a web browser interface and provides an interface for MATLAB as well as support for the Bio-Formats library (see below). The strengths of the OME server are its server-side analysis and its infrastructure for dynamic typing, a scheme where the type of a variable is deduced from its value during run-time.

The lack of advanced visualisation and a non-user-friendly design made it complicated to use for people without a background in informatics. To overcome these problems, the OME consortium started the development of a remote data client called Shoola in 2003. Shoola was meant to improve visualisation and management of image data with the OME server. Because the remote interfaces available in the Perl-based OME server did not offer sufficient data throughput, the functionality of Shoola was limited. Therefore, the development of Shoola was stopped in 2005. The aforementioned drawbacks of the OME server have led to the development of a second, self-contained software system called OMERO.

OMERO server and client software

The development of the OMERO software started in 2004, when the OME consortium became aware of the drawbacks of the OME server. The main attention during the development of OMERO was focused on a fast client-server communication, advanced visualisation and improved usability. The data transfer issues of the original OME server was overcome by switching from Perl to Java as the underlying programming language, that enabled more powerful remote interfaces. A more user-friendly design of the software was achieved by the aid of the Usable Image (UI) team [127], which started its work on OMERO in 2006. The UI team applies user-centred design techniques to an on-going software development project and thus allows the adjustment of the software according to the user's

⁴Perl is an interpreted programming language developed in 1987 by the American programmer Larry Wall.

needs already in an early stage of the development process. The aim of OMERO is similar to the aim of the original OME server: visualising, managing and annotating microscope images and their corresponding metadata. Although OMERO's server-side analysis features are still in their infancies, other features like remote access, visualisation and platform independency already outperform the abilities of the original OME server. Furthermore, OMERO allows development of client software. Due to different underlying data models, the two servers are not compatible. Nevertheless, the software developers provide a tool for migrating OME data into the OMERO data format. OMERO consists of the following separate but linked subsystems:

OMERO.server

OMERO.server is the central application of the OMERO software system. As depicted in Figure 2.7, client applications can communicate with the server over a network. Image data and metadata are managed with a relational database. Currently, OMERO supports PostgreSQL [95], a free object-relational open source database system. Metadata and its relations to the underlying image data are stored in the database, while the actual image data are stored in an image repository that can be remote.

OMERO.importer

OMERO.importer is a Java-based client that uses the Bio-Formats library to read various image file formats and upload them to the OMERO server.

OMERO.insight

OMERO.insight has been developed from the above mentioned OME client Shoola. It can be used to view, categorise and annotate images from an OMERO server.

OMERO.webadmin

OMERO.webadmin is a web browser-based server administration tool.

OMERO.editor

OMERO.editor is a stand-alone tool for recording experimental metadata.

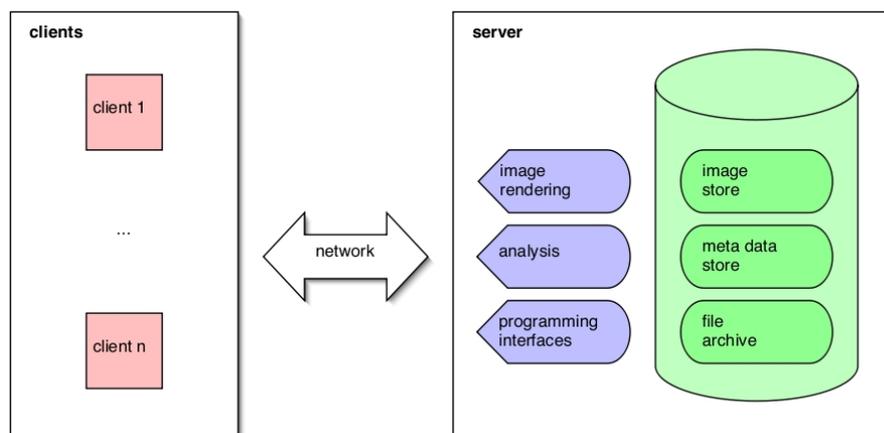
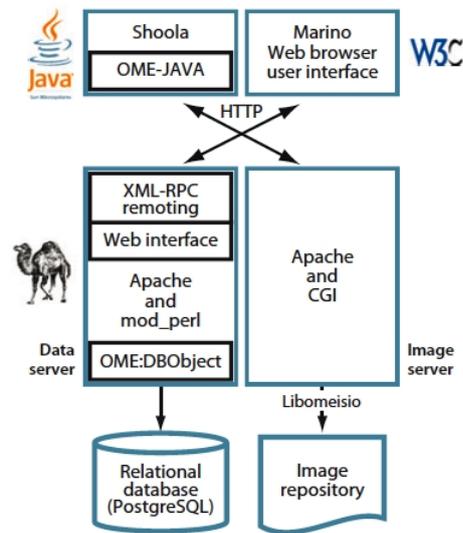
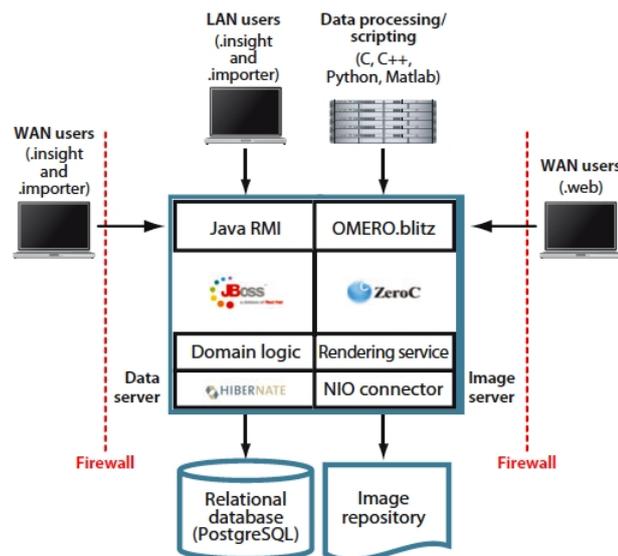


Figure 2.7: Scheme of the OMERO client-server infrastructure: a number of n clients (red) can connect to the server over a given network infrastructure. Each client can trigger server-side events (blue). Image data is stored (green) as image and metadata. Furthermore, the original file can be archived.

For this thesis, the OMERO beta-4.1.1 release has been used. The differences between the original OME server and the OMERO platform are summarised in Figure 2.8. The interaction of the different OMERO client applications for acquiring, managing and analysing image data with the OMERO server is shown in Figure 2.9.



(a) OME Server 2.6.x



(b) OMERO platform (beta-3 and beta-4)

Figure 2.8: Architecture of the OME server and the OMERO platform, including servers and client applications. (a) Architecture of the Perl-based OME server. (b) Architecture of the OMERO platform. While the beta-3 release was based on the JBoss Application Server [58], the beta-4 release has changed to an alternative remoting architecture called Internet Communications Engine (Ice) [134] (source: [116]).

ServerDesign

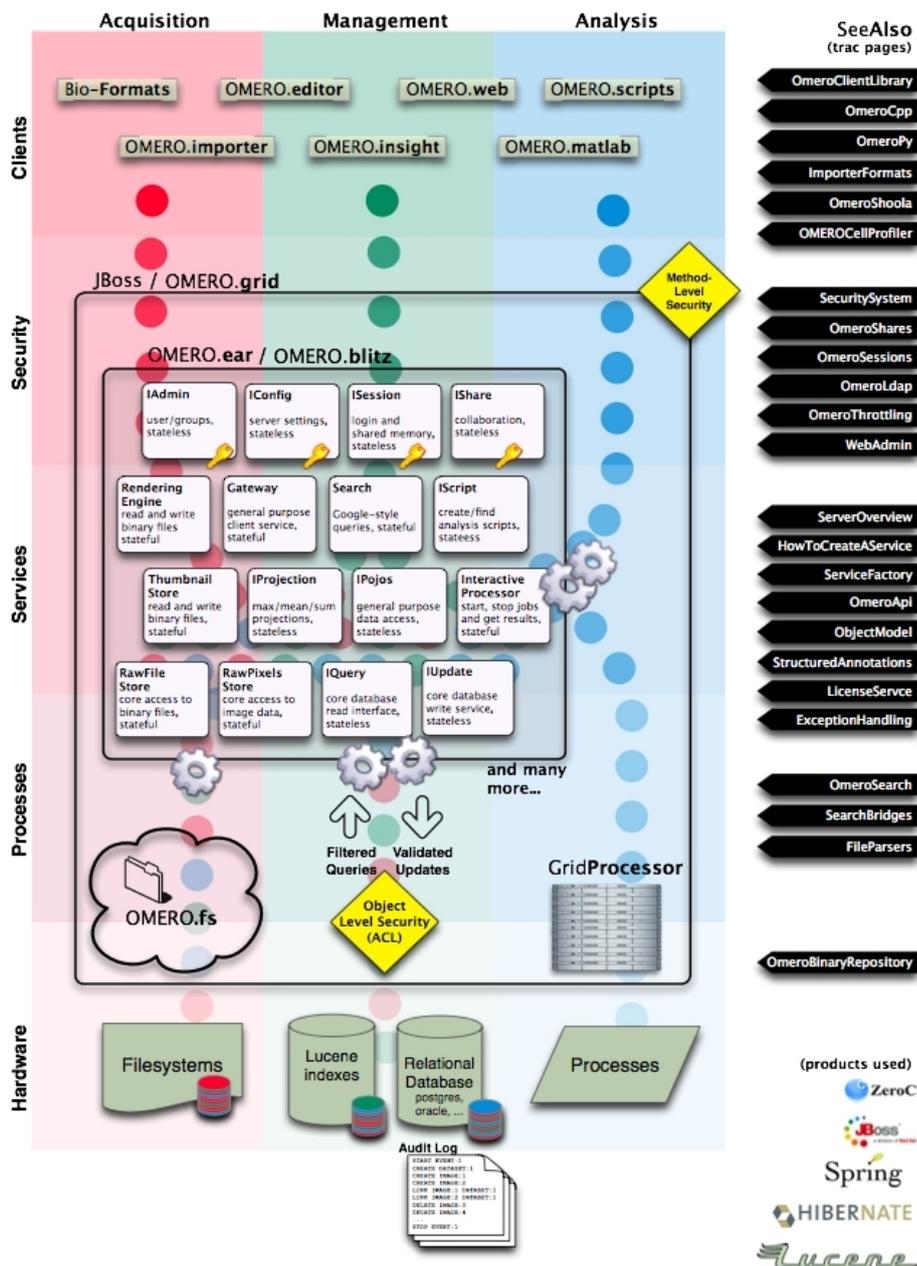


Figure 2.9: Schematic drawing of the server design and the interplay of different client applications for acquisition, management and analysis of image data with the OMERO server (source: [126]).

Bio-Formats

The Bio-Formats Java library (Laboratory for Optical and Computational Instrumentation [68], Madison, USA) is used to read and write an increasing number of microscopy file formats. The library is directly accessed by the OMERO.importer client in order to convert the raw file format into OME-XML and OME-TIFF (see below). In addition, it allows for archiving the raw file format. Users who need to handle a specific file format which is not yet supported by Bio-Formats can send a request to the Bio-Formats team. Even proprietary file formats can be passed through a reverse engineering routine and may thus become part of the Bio-Formats library.

OME-XML and OME-TIFF

Many file formats for microscopy images are proprietary and differ between manufacturers. In order to use them with software from other vendors, images have to be converted into another file format, a process during which a critical loss of metadata can occur. To avoid this problem, the OME-XML [87] project has developed a data model and file format specifications, which support the exchange of microscope imaging data. The data model is expressed as an XML scheme. It stores metadata such as microscope and camera information, dimensional parameters, hardware configuration, information about the experimenter and some details about the experiments itself. The purpose of the OME-XML data model is to put the actual image data into the right scope of context.

Using the OME-XML data model, OME-TIFF [86] combines the metadata with the actual image content. OME-TIFF is a high performance interchange format with OME-XML inside a regular TIFF file. Thus, an OME-TIFF file can be read with any TIFF-compatible software. The structure of the header of an OME-TIFF file is depicted in Figure 2.10.

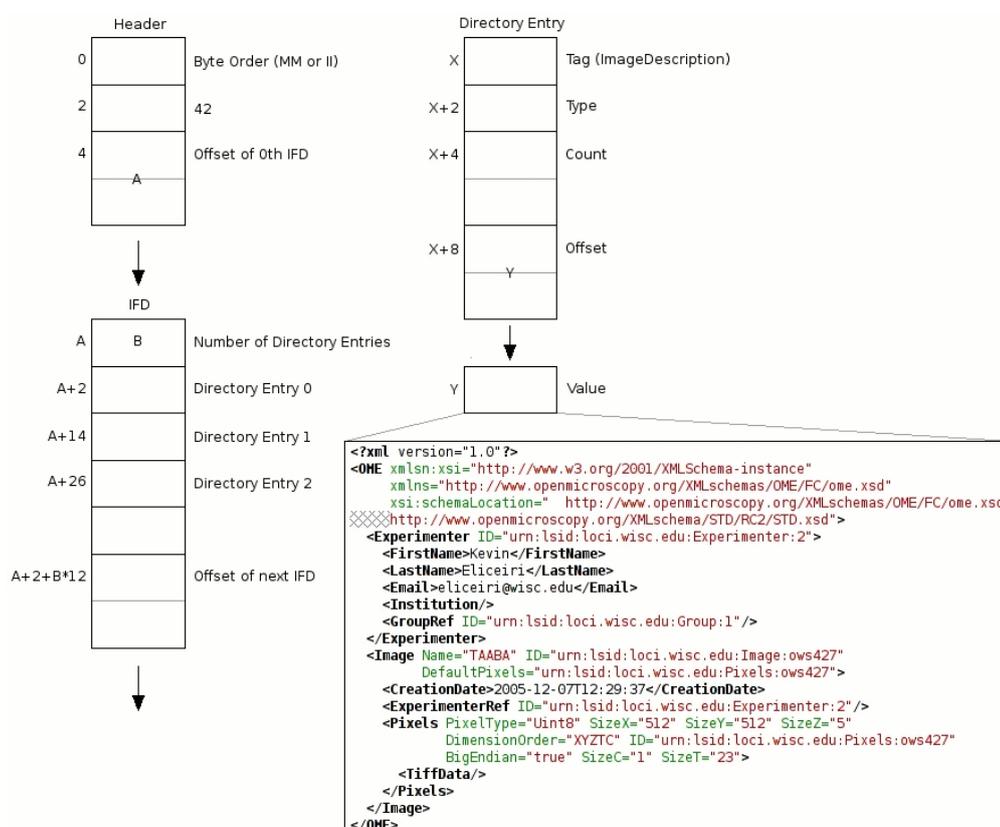


Figure 2.10: Structure of an XML-TIFF header. The text box contains metadata in OME-XML format (source: [87]).

2.7.2 Bacula®

The open source network backup solution Bacula® [10] has been developed by Kern Sibbald and his team. Since the start of the project in January 2000, Bacula® has become the most popular open source backup software for managing, recovering and verifying computer data within a heterogeneous network environment. Mainly implemented in C++ (see Section 2.5.3), the client-server architecture of Bacula® supports all major operating systems including Linux, Unix, Mac OS X and Microsoft Windows. Furthermore, Bacula® supports a wide range of professional tape drives including our IBM tape library (see Section 2.4.3). It stores its information either in a MySQL, PostgreSQL or SQLite database back-end.

2.8 Software for Distributed Computing

Our HCS environment consisted of individual modules, each performing specific tasks while running on dedicated hardware. Finally, each module contributed to the HCS result. Figure 2.11 depicts the distributed computing environment of our HCS system. Modules shown in green have been developed by ourselves and are described in Chapter 3. Modules shown in orange have been implemented by our co-operation partners and are described in this chapter.

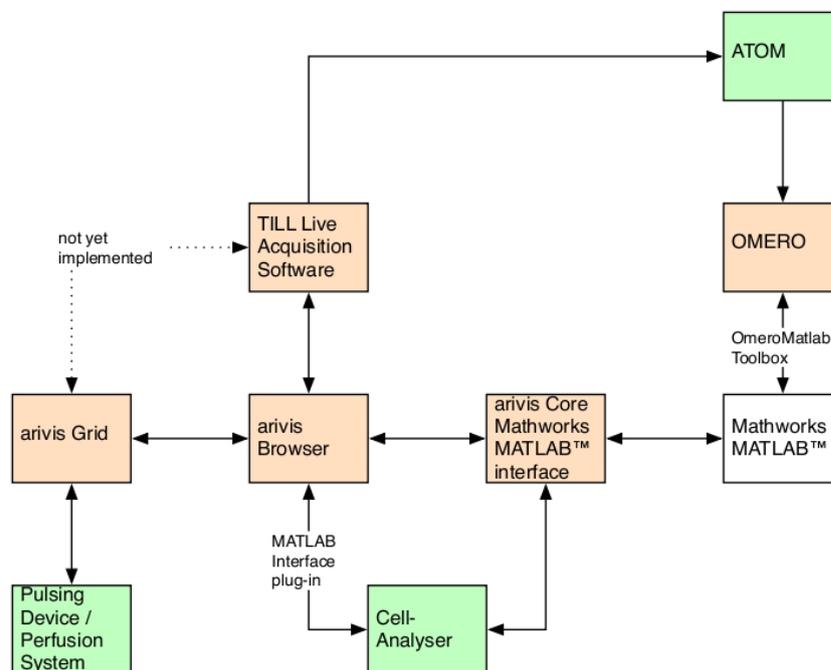


Figure 2.11: Distributed computing environment of our HCS system. Modules shown in green have been developed by ourselves while modules shown in orange have been implemented by our co-operation partners. Arrows show the direction of possible information flow between the different modules.

2.8.1 arivis Grid

The arivis Grid (arivis Multiple Image Tools GmbH [9], Rostock, Germany) is a communication framework that enables software developers to easily design and develop grid based solutions. Grid computing is characterised by a loose coupling of individual computers, which results in a virtual super computer. Following this principle of distributed computing, the arivis Grid is very flexible, especially with respect to the heterogeneity, the coordination and the geographical distribution of the different resources. This flexibility is supported by the use of open and standardised interfaces and protocols such as the transmission control protocol (TCP) together with the internet protocol (IP). The application programming interface (API) of the arivis Grid was available as a Java and a C++ version. It provided the following components:

Grid client

The first step to interact with the arivis Grid is to connect to the grid as a grid client. Since a grid client can manage multiple actors (see below), it is sufficient to register one grid client. Registering multiple grid clients allows an application to participate in different grids simultaneously.

Actors

Once a grid client has been registered, the second step is to create and to register an actor. An actor is an entity that can execute specific and well defined commands. These commands are called capabilities (see below).

Capabilities

Each actor is able to receive, to process and to send specific commands. These commands are defined using an XML scheme, which finally represents the actor's capabilities. Without capabilities, an actor cannot participate in a grid scenario.

Bridge

Communication between actors utilises a bridge, the low-level transport layer of the arivis Grid. The arivis Grid provided two different bridges: one bridge used the NaradaBrokering Framework [124] (Community Grids Lab, Bloomington, USA), the second bridge used TCP/IP.

The arivis Grid was implemented on the basis of the messaging server prototype, which is described in Chapter 3.

2.8.2 arivis Browser

The arivis Browser is a software, which can be used for acquisition, management, visualisation and interpretation of image data. It can efficiently cope with 2D images, 3D image stacks up to 5D image data ($x, y, z, t, channels$) with a maximal resolution of $2^{31} \times 2^{31}$ pixels. This efficiency is supported by the single image stack (SIS) file format, the native and proprietary file format of the arivis Browser. The underlying modular design makes the arivis Browser flexible and thus allows for an easy enhancement and adaption of the software.

Our version of the arivis Browser was equipped with the “LA connector” plug-in, which managed the communication with the TILL Live Acquisition software (see Section 2.8.4) in order to perform live analyses. Furthermore, the MATLAB Interface plug-in allowed the execution of MATLAB scripts directly within the arivis Browser.

2.8.3 arivis Core Mathworks MATLAB™ Interface

The arivis Core Mathworks MATLAB™ Interface (ABMI) connects the arivis Browser to MATLAB. It was available for Microsoft Windows as a 32-bit dynamic-link library (DLL). By using this DLL, MATLAB can be enhanced to handle enormous image datasets. In addition, the arivis Browser can make use of MATLAB’s analysis capabilities by communicating to the ABMI via socket connections, for instance. In contrast to the MATLAB Interface plug-in, the ABMI can be used

to access the arivis Browser and its underlying data core directly. This enables MATLAB to read and analyse images that are available in the SIS file format.

2.8.4 TILL Live Acquisition Software

The TILL Live Acquisition (LA) software (TILL Photonics) is part of a real-time imaging system for time-lapse recordings and automated screening. It provided an interface for control of the imaging hardware described in Section 2.3. While the LA software is mainly used for data acquisition, the arivis Browser (see Section 2.8.2) is used for data evaluation. For this, the LA software can directly communicate with the arivis Browser. The built-in graphical protocol editor of the LA software allows the generation and editing of workflows. Usually, a workflow involves the controlling of the imaging hardware and its periphery. The protocol editor therefore plays a central role with regard to planning and performing fully automated experiments.

2.9 Approaches for Image Processing and Image Analysis

In our case, images are acquired during an HCS experiment using the monochrome camera described in Section 2.3.1. To extract the information required for the computation of calcium transients such as shown in Figure 1.3, usually the original image is used to generate a binary image as one of the first steps. This binary image should contain the value 1 only at those points, where the cell is located. At all other points the value should be 0. Such a binary image also is referred to as a cell mask. Individual cell masks are necessary to obtain individual calcium traces, because they remove unwanted information originating from other cells if multiplied with the original image. Since more than one cell is usually captured in an image, the generation of a cell mask requires a clear separation (segmentation) of the single cells. Approaches which can be used for this purpose are introduced in the following.

2.9.1 Watershed Transformation

The watershed transformation is a segmentation algorithm used in image processing. It is based on the topology of the image, using the length of the gradients as elevation information. Utilising foreground (i.e. a region inside the cell) and background (i.e. a region outside the cell) markers determined with the aid of morphological operations such as erosion or dilation, successive flooding of the grey value relief is performed, starting from the marker positions. This approach is referred to as marker-based watershed transformation. The flooding process results in adjacent basins separated by edges. Ideally, each basin represents a single image object. The watershed transformation used for this thesis was implemented as part of the MATLAB Image Processing Toolbox. It used the Fernand Meyer algorithm [76].

2.9.2 Edge Detection

To compute the gradient magnitudes for the watershed transformation described above, the Sobel operator [113] (see below) has been used. It was implemented as part of the MATLAB Image Processing Toolbox, which provided a couple of sophisticated methods for edge detection. These methods can be used to detect cell boundaries in a microscope image. Each method is based on the computation of the numerical approximation of derivatives in the spatial domain of the image.

One of the earliest edge detection algorithms is called Roberts' Cross operator, which is also referred to as Roberts filter [103]. It has been developed in 1965 and is still in use, mostly due to its speed of computation and its capability of producing fine lines. As a first order derivative operator, an edge is introduced at those points where the gradient of a greyscale or binary image is maximum. The major drawback of the Roberts' Cross operator is its sensitivity to noise, which is always present in digital microscope images. Therefore, this edge detection method is not well suited for our HCS project.

In 1968, the Sobel operator has been developed [113]. It also detects edges at those points, where the gradient of the image is maximum. The gradient

approximation is computed by using coefficients to weight the image intensities. The Sobel operator is also based on first order derivatives and is therefore more robust with respect to noise than algorithms computing higher order derivatives. Despite its robustness, it does not always create closed contours.

The edge detection of the Prewitt operator [96], which was introduced in 1970, is very similar to the Sobel operator. The only difference is the lack of coefficients for weighting the current gradient direction.

Furthermore, the Image Processing Toolbox provided the Laplacian of Gaussian method, which is also referred to as Marr-Hildreth operator [73]. It has been developed in 1980 and is based on second order derivatives. After filtering an image with a Laplacian of Gaussian filter, edges are introduced by looking for zero-crossings, points in the image, where the values of the second order derivative cross through the point of zero. Although this algorithm produces closed contours, it is more sensitive to noise than algorithms based on first order derivatives. Since it does detect both, the maxima and the minima of the first derivative, false edges may be generated. Moreover, strong Gaussian smoothing may lead to incorrect edge locations.

Similar to the Sobel method, the Canny edge detection operator [22] detects edges by looking for local maxima of the image's gradient. It has been developed in 1986 and is again based on first order derivatives. By combining the computation of strong and weak edges, the Canny edge detection operator is more robust against noise and allows to detect true weak edges.

As can be deduced from the above, edge detection methods based on first order derivatives are better suited for the underlying HCS project than methods based on higher order derivatives of the spatial domain. It has turned out, that the Sobel edge detection method produces slightly better results than the Canny operator. This may be due to the fact that the boundaries of cardiomyocytes are relatively uniform with respect to their thickness, which may bypass the strength of the Canny operator of combining strong and weak edges. Coincidentally, the Sobel method is the default edge detection method used by MATLAB, if none of the other methods is specified. Hence, the Sobel operator has been used in the

CellAnalyser software, which is described in Chapter 3.

2.9.3 Circular Hough Transform

To compute the cell masks of cells with a regular roundish shape such as erythrocytes (see Section 2.10.2), the circular Hough transform was used. It is a variation of the classical Hough transform (see [47, 48]), which is used in image processing and analysis. While the latter was concerned with the identification of lines in an image, the circular Hough transform has been extended in order to identify positions of circles using the gradient field of an image.

2.10 Biological Specimen

In order to test the flexibility of the HCS system developed in this dissertation, the following cell types, which are differing in their regularity in shape as well as in their intrinsic behaviour, have been used:

2.10.1 Cardiomyocytes

Adult rat cardiomyocytes were taken from male Wistar rats (6–12 weeks old, 200–400 g body weight). Isolation and culture was performed as described in [61]. A detailed description of cardiomyocytes is given in Section 1.1.2.

2.10.2 Erythrocytes

Freshly drawn blood from wild type mice was used to isolate erythrocytes as described in [62]. These red blood cells have been chosen for this project due to their regular, roundish shape, which represents a contrast to the non-uniform shape of cardiomyocytes.

Chapter 3

Results

3.1 Preparatory Work

3.1.1 Software for Triggering Laboratory Equipment: TriggerIT

Controlling laboratory hardware can be achieved by many different means: (i) specialised bus systems such as HPI¹; (ii) general interfaces such as RS-232² or USB; (iii) by simple TTL level triggering. Each of these methods has their advantages and drawbacks. In an HCS environment very often real-time behaviour is an important prerequisite. Since TTL level triggering is often used for synchronising laboratory equipment in real-time, my first software project was to design and create a tool for general TTL level trigger synchronisation: TriggerIT. This tool has been implemented using LabVIEW (see Section 2.5.1) and the data acquisition board DT335 (see Section 2.1.2). Its purpose was to synchronise laboratory hardware and software in order to execute predefined measurement protocols. The triggering and wiring scheme is depicted in Figure 3.1: monochromator, measurement software and valve controller were triggered independently making use of three digital output channels of the DT335. Alternatively, TriggerIT itself could be triggered

¹HPI stands for Host Port Interface, a parallel data bus on a computer system.

²RS-232 is a standard for serial data and control signals and commonly used in computer serial ports.

by an external hardware, e.g. MyoPacer Field Stimulator (IonOptix [55], Milton, USA), making use of a digital input channel.

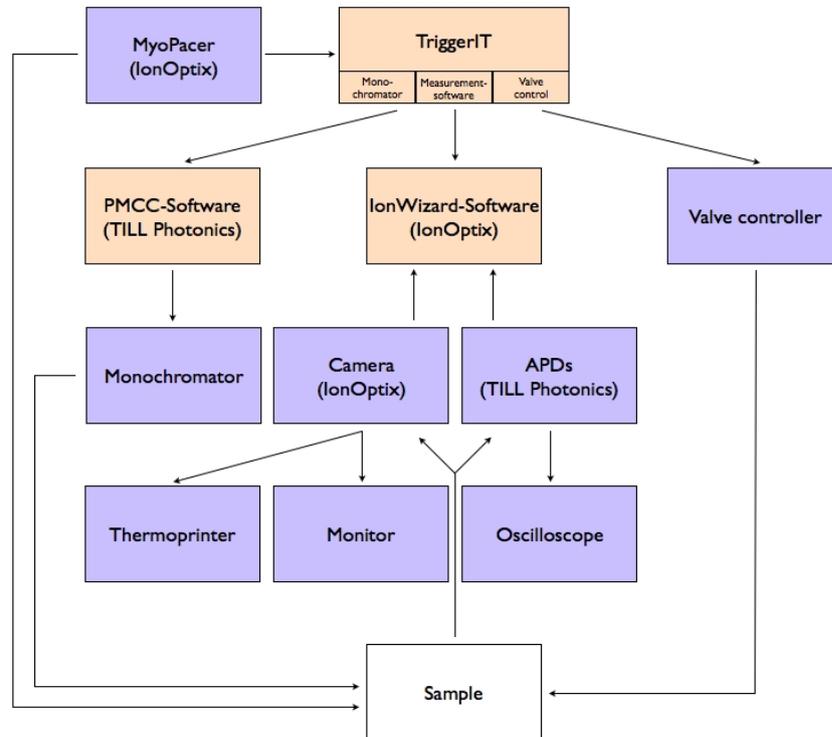


Figure 3.1: Information flow diagram of the setup running the TriggerIT software. Blue boxes represent hardware, orange boxes represent software. The directions of the trigger and information flow are represented by arrows.

Implementation

The GUI, depicted in Figure 3.2, was used to configure four independent digital output channels (monochromator, measurement software, valve controller plus an additional channel), arranged in four graphical panels. Each panel consisted of ten columns, where each column was used to set the TTL signal of the corresponding channel to high or low for a defined amount of time. Thus, each panel allowed to define a pulse train, which was executed a defined number of times. For ease of use,

names could be assigned to each panel. Defined pulse trains could be saved and loaded. By selecting the trigger option "sync with MyoPacer", TriggerIT waited for an incoming TTL signal on a digital input channel before executing the defined pulse trains. Otherwise, the pulse trains were executed directly after pressing the start button. Figure 3.3 depicts the flowchart of TriggerIT. The I/O channel setup of the DT335 is shown in Table 3.1.

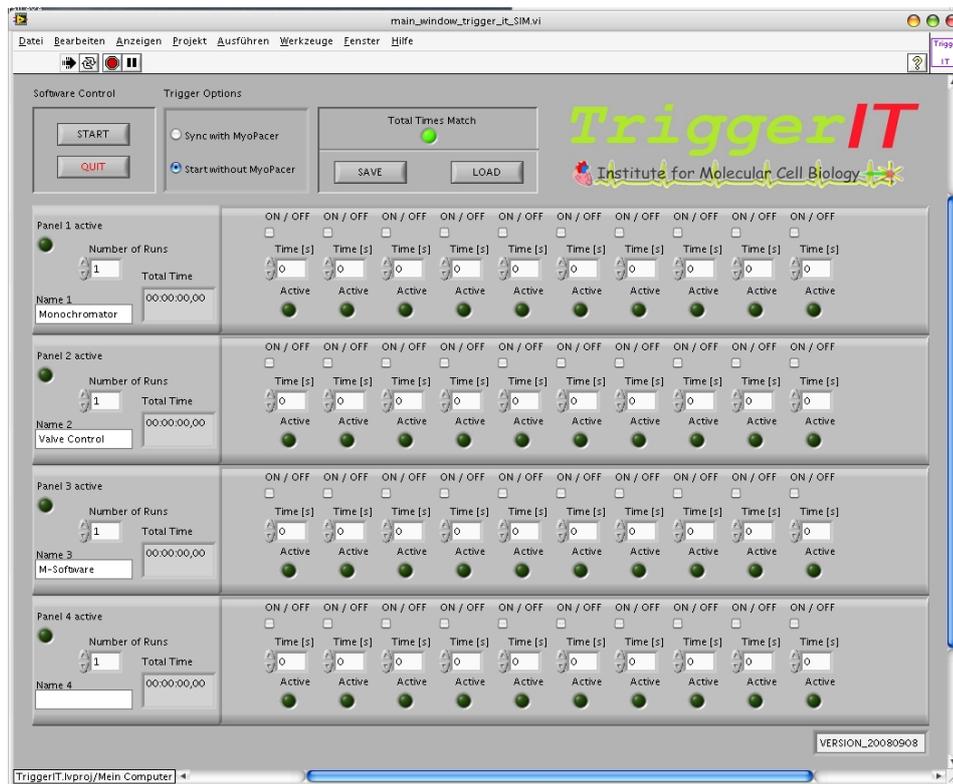


Figure 3.2: GUI of the TriggerIT software.

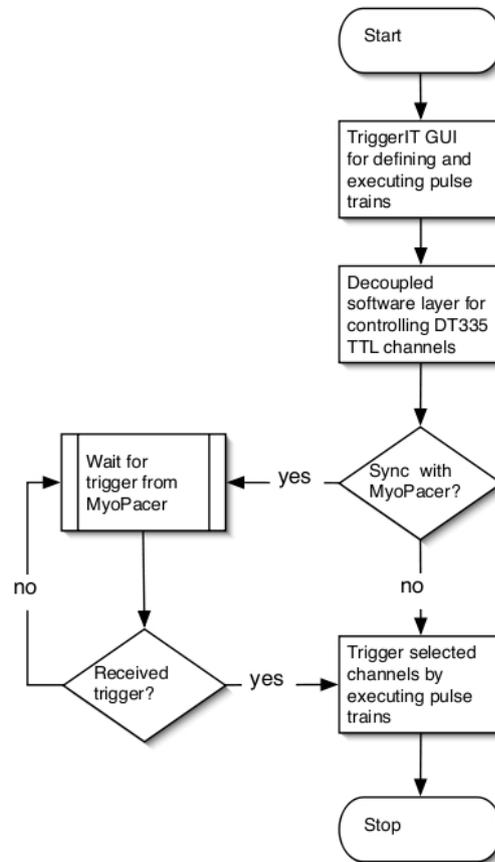


Figure 3.3: Flowchart of the TriggerIT software.

Table 3.1: I/O channel setup of the data acquisition board DT335. Four channels were used as digital output, one channel was used as digital input. Channel and port names correspond to the labelling predetermined by the hardware.

	Port 2	Port 3
Digital Output Channel	3	1 2 3
Digital Input Channel	5	

User interactions during an executed experiment were enabled by decoupling the GUI and the hardware layer using the producer-consumer design pattern (see Section 2.5.1). The GUI was kept alive by a while loop, which enabled the software to receive values entered by the user. These values were sent to a software layer running in a parallel while loop via a data queue. Otherwise, the GUI would have frozen until the pulse trains had been executed completely. The four digital output channels were controlled independently by using a while loop for each channel running in parallel. Each while loop can be considered as a software thread. Thus, TriggerIT handled the four digital output channels in a multi-threaded manner, allowing triggering multiple channels concurrently. Trigger signals were generated using real-time functions contained in the LabVIEW Real-Time Module. While the standard functions for timed applications only offered a resolution in the range of milliseconds, for real-time functions a resolution in the range of microseconds could be achieved.

3.1.2 Enhancement of the TriggerIT Software: ClickIT

Another important point regarding software development is usability. The functionality of software has to be made accessible to the user without negatively affecting its ease of use. This can be achieved by a clear and concise structure of the GUI as well as by providing easy ways for accessing and handling software features. Therefore, the software TriggerIT, as introduced in the previous section, was re-engineered to improve its usability, while maintaining its functionality. As a consequence, the revised software called ClickIT was also implemented using LabVIEW and the LabVIEW Real-Time Module. The GUI of ClickIT is shown in Figure 3.4. It was subdivided into the following areas:

Control Bar

This area contained buttons for an easy access to the following functions:

- *Initialise*: set all TTL levels to low, put the system in a defined state.

- *Trigger channel*: individually set the TTL level of four channels either high or low for a specified amount of time.
- *Wait*: set a pause with a specified duration.
- *Wait until user interacts*: also set a pause, but in contrast to the previously described wait function, this function interrupted the workflow until the user interacted with the system.

Info Pad

This area informed the user about the effect of the selected function. Including a function in a measurement protocol was achieved by pressing the “add to script” button.

Script Editor

This area was used to prepare a measurement protocol. A prerequisite was to improve the readability and handling of protocols compared to TriggerIT. Since most web browsers can interpret XML commands and thus display the content of even large XML files in a clearly arranged manner, I have decided to represent all functions of ClickIT as XML code. An overview of available XML commands and their relationship to the function buttons is given in Table B.1 (see appendix). Thus, measurement protocols could be read and modified with any text editor. Blocks of XML code could be copied and pasted arbitrarily. As a result and in contrast to TriggerIT, ClickIT was not limited to ten columns per channel.

System Control

This area was used to create, save and load protocols. It also enabled the user to start, stop and exit ClickIT.

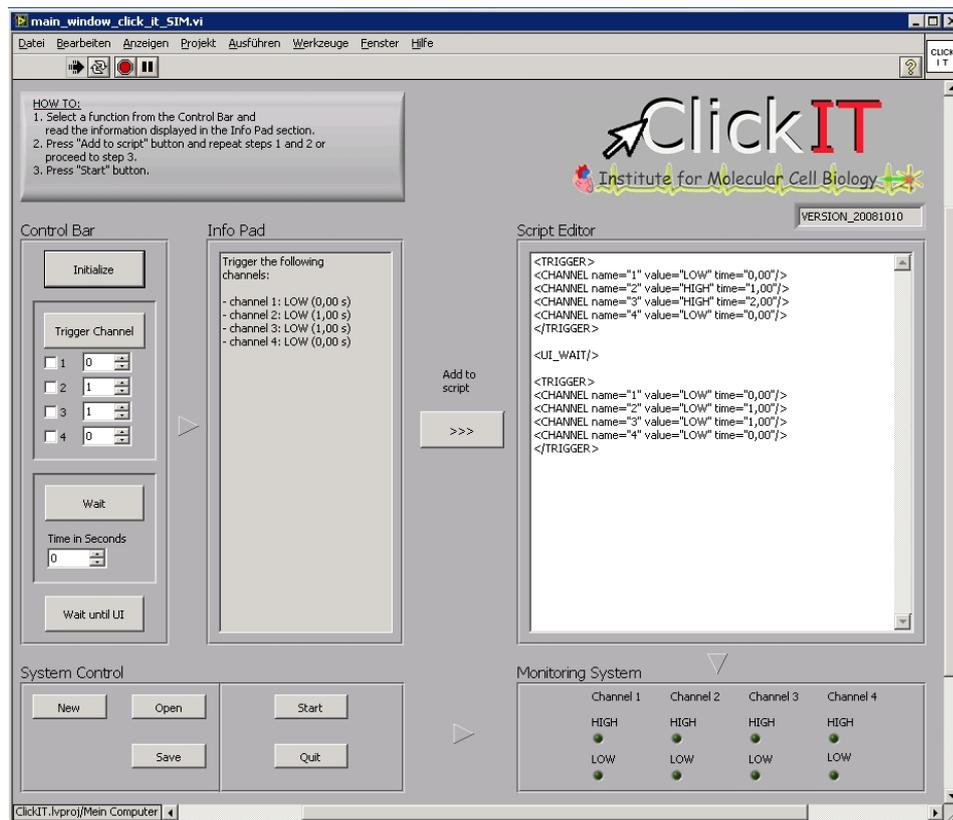


Figure 3.4: GUI of the ClickIT software. The clear and concise structure led to an improved usability in comparison to the TriggerIT software.

Monitoring System

This area showed the state of the TTL levels of the different channels. If an output channel was set to high, the corresponding LED indicator was enlightened.

3.1.3 A Software Prototype for Electrical Field Stimulation of a Single Well: PROTOzoon

Modularity was the underlying paradigm with regard to the development of the software for electrical stimulation of cardiomyocytes. Therefore, the most basic component of the software had to be as simple as possible. Instead of developing

the software top-down by immediately starting with code for a multi-well plate, I initially focused on writing efficient code for controlling the electrical field stimulation of a single compartment. From this idea the software PROTOzoon arose. The whole setup is depicted in Figure 3.5(a).

I used PROTOzoon to realise the following tasks (see Section 2.1 for hardware details):

- testing and setting up the hardware (NI SCC-68, NI PCI-6229, electrical circuit board) for electrically stimulating cardiomyocytes
- testing and finding of suitable SSR's (OMRON G3DZ, Panasonic AQY225R1S)
- precise generation of an alternating negative / positive pulse (see Figure 3.5(b) and 3.5(c))
- development of optimised code for the core of the software for electrical field stimulation

Implementation

PROTOzoon has been implemented using LabVIEW and the LabVIEW Real-Time Module. In the source code, two arbitrary digital output channels of the digital I/O-card NI PCI-6229 were defined. For every well two channels controlled the negative and the positive voltage, respectively. As depicted in Figure 3.6, the GUI enabled the variation of the pulse width as well as the gap between negative and positive pulses. The scheme is depicted in Figure 3.7. In order to verify correct switching of the relays, an oscilloscope was used to read out the electrical signal at the carbon electrodes.

Since the software was programmed under Windows XP, which is not a real-time operating system, I critically checked the LabVIEW code. During thorough testing of the software, I found that the use of Boolean “LED indicators” in time-critical scopes negatively affected the real-time behaviour. This is due to the fact, that assigning a value to a variable and sending the result to the GUI consumes

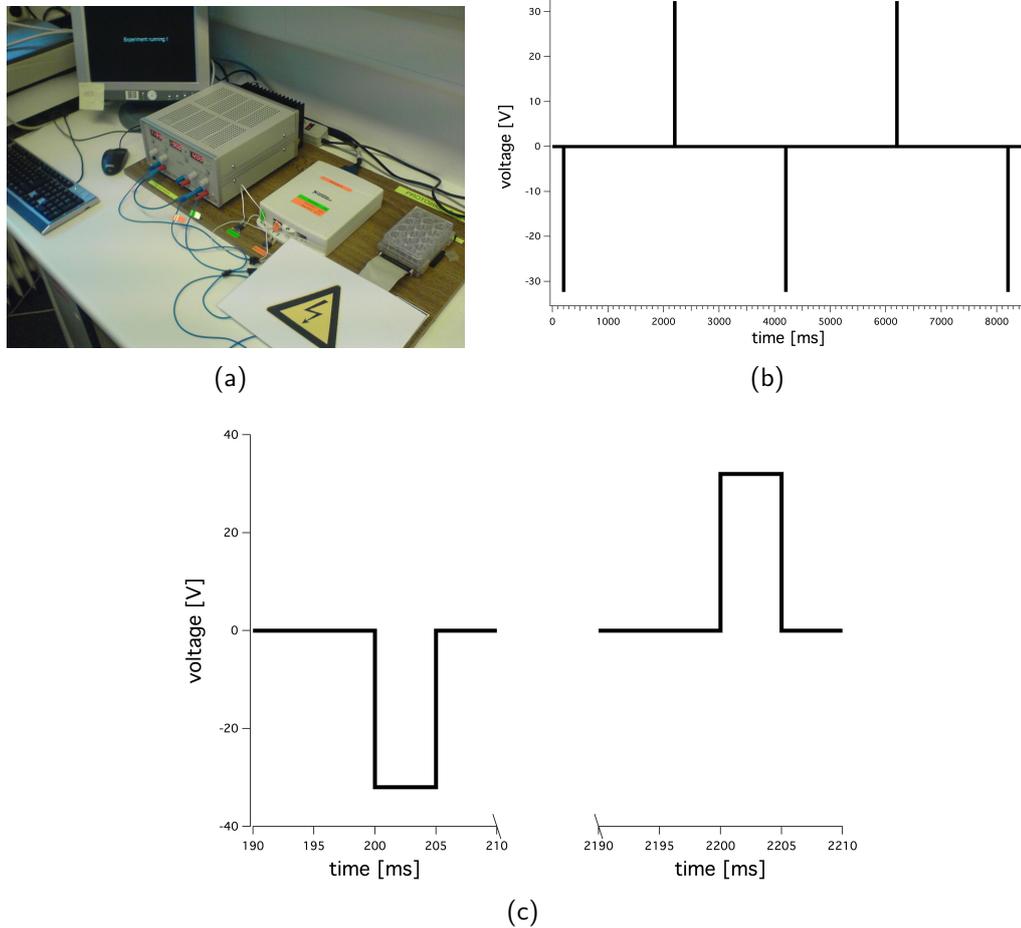


Figure 3.5: (a) Photo of the PROTOzoon setup: trigger signals were sent from the NI PCI-6229 (not shown) to the NI SCC-68 connector block (white box in the middle), which was connected to two relays. The relays were switching the current of the dual power supply (left) in an alternating manner and conducting it to the carbon electrodes of the lid (right). (b) Example of an alternating negative / positive pulse train generated by the PROTOzoon setup (voltage: ± 32 V; pulse width: 5 ms; frequency: 0.5 Hz). (c) Scale-up of an alternating negative / positive pulse.

computing power. Using the snippet of LabVIEW code shown in Figure 3.8, the shortest trigger that could be detected by the oscilloscope was 18 ms, although the pulse width was set to 5 ms on the software side. After removing the LED

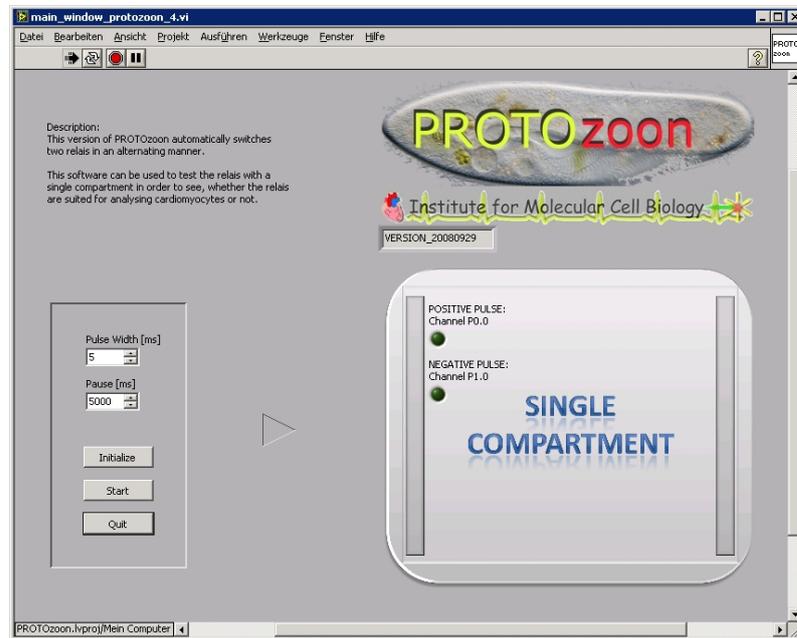


Figure 3.6: GUI of the PROTOzoon software.

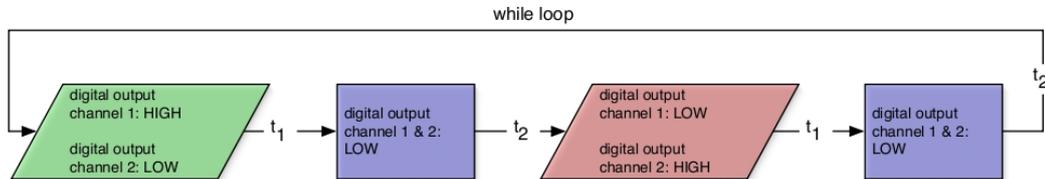


Figure 3.7: Scheme displaying the generation of the alternating negative / positive pulse train. A pulse is generated by setting the TTL level of a digital output channel to “high” for the time t_1 . Each pulse is followed by a pause (blue boxes), where all digital output channels are set to “low” for the time t_2 .

indicators from the GUI, pulse widths below 5 ms could easily be achieved. The conclusion from this finding was to use as less LED indicators in the GUI as possible, at least in time-critical scopes.

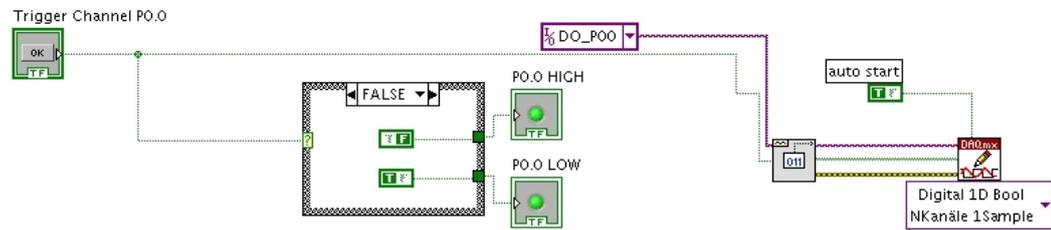


Figure 3.8: A snippet of LabVIEW code from the PROTOzoon software: using the two Boolean LED indicators "P0.0 HIGH" and "P0.0 LOW" permitted a minimal pulse width of 18 ms. After removing the two LED indicators, shorter pulse widths could easily be achieved.

3.1.4 Long-term Material Testing

For the design of the HCS stimulation plate, the material described in Section 2.1.1 had to be tested with respect to its robustness during heat sterilisation. Therefore, the carbon electrodes, the 50 pin ribbon cable interface, the sheet-metal screws and a polycarbonate sample with imprinted titanium-gold alloy conducting paths were autoclaved for a duration of 40 minutes at a temperature of 120 °C. The procedure was performed 30 times. To test the Cramolin Plastik protective paint, the polycarbonate sample had been covered by this transparent acrylic resin.

The electrical conductivity of the carbon electrodes and of the 50 pin ribbon cable interface was unaffected by the procedure described above. After the testing period, both components were still in good order.

The sheet-metal screws were covered by a thin oxidative layer, which inhibited the electrical conductivity of the screws. Since these screws have later been used for connecting the clamps holding the carbon electrodes to the circuit paths of the lid, they had to be covered by the Cramolin Plastik protective paint. This isolated the screws against moisture and thus preserved their electrical conductivity.

The Cramolin Plastik protective paint successfully isolated the conductive paths against external influences. Both, the protective layer and the underlying circuit paths were still intact. Since this paint should only be exposed to a temperature of 120 °C temporarily, a slight blistering occurred. Therefore, instead of using the

above mentioned sterilisation procedure, the lid had to be autoclaved at 80 °C for a longer period of time.

3.2 Experimental Control

This section describes both, the hardware and the software that has been developed for the experimental control of the HCS system.

3.2.1 General Setup

Figure 3.9 depicts a basic overview of the principle setup for electrical field stimulation in the 24-well plate described in Section 3.2.5. Pulse trains with TTL level generated by the digital I/O-card under the control of the Pulsing Device software controlled an array of SSR's in the SSR device (see Section 3.2.3), in which the two stimulation voltages of opposite sign were coordinated to generate alternating voltage pulses. The pulses were routed through the patch bay (see Section 3.2.4) onto the carbon electrode array of the 24-well plate.

3.2.2 Settings of the Voltage Source

According to our tests, a maximum voltage of 25 V was required to get a sufficient stimulation of the cells. The maximum current to be supplied can be derived from Equation 3.1:

$$I = \frac{U \cdot A}{\rho \cdot d} \quad (3.1)$$

where

I : maximum current

U : upper limit of the voltage (25 V)

A : effective area of the electrodes (15 mm × 10 mm)

ρ : specific electrical resistance of culture medium (125 Ω · cm)

d : distance between the electrodes (12 mm)

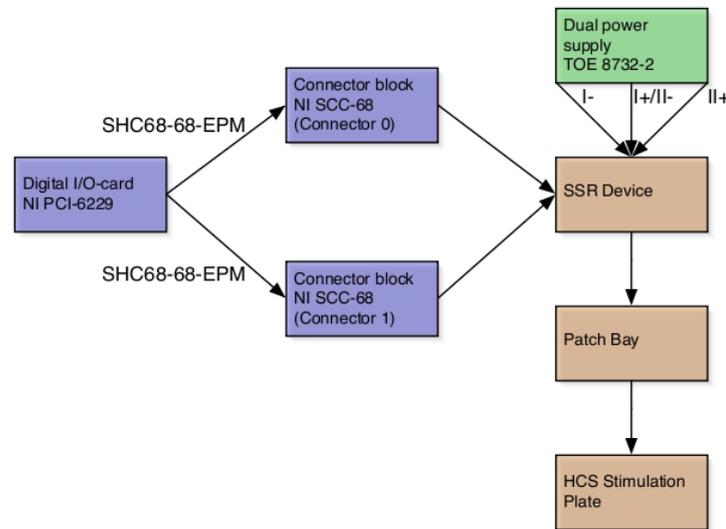


Figure 3.9: Arrangement of the different hardware components for electrical field stimulation (blue: obtained from National Instruments; green obtained from TOELLNER Electronic Instrumente GmbH; orange: own production).

In our case, a maximum current of 250 mA per well had to be applied, resulting in a maximum current of 6 A for the entire 24-well plate. Discussions with our scientists revealed, that experiments usually are repeated several times, in order to verify the results. Thus, I had the idea to combine 3 wells of the 24-well plate for concurrent chronic pulse application. Besides identical pulses in each of the 3 wells, this led to a maximal power supply output of 750 mA, which could be handled by the dual power supply described in Section 2.1.5. Additionally, this strategy allowed the application of an acute pulse protocol with pulse trains that may overlap with pulse trains in chronically stimulated wells, thus leading to a maximum power requirement of 1 A in the worst case. This was handled by the stimulation software (see Section 3.2.6), which enabled starting sequential pulsing of the eight well triplets, as shown in Figure 3.10. A slight gap between consecutive triplets prevented exceeding the upper power limit. By ensuring that only a maximum of 3 wells was being pulsed concurrently, application of an acute pulse protocol to a single well was possible, while the remaining wells maintained

their chronic stimulation mode. This is exemplified for one triplet in Figure 3.10. Figure 3.11 shows the wiring scheme of the circuit diagram for a single well. This circuit diagram was used for each well of the plate. I decided to connect the electrodes of each well with reference to the same mass, assuring that the voltage amplitudes inside each well had the same reference point.

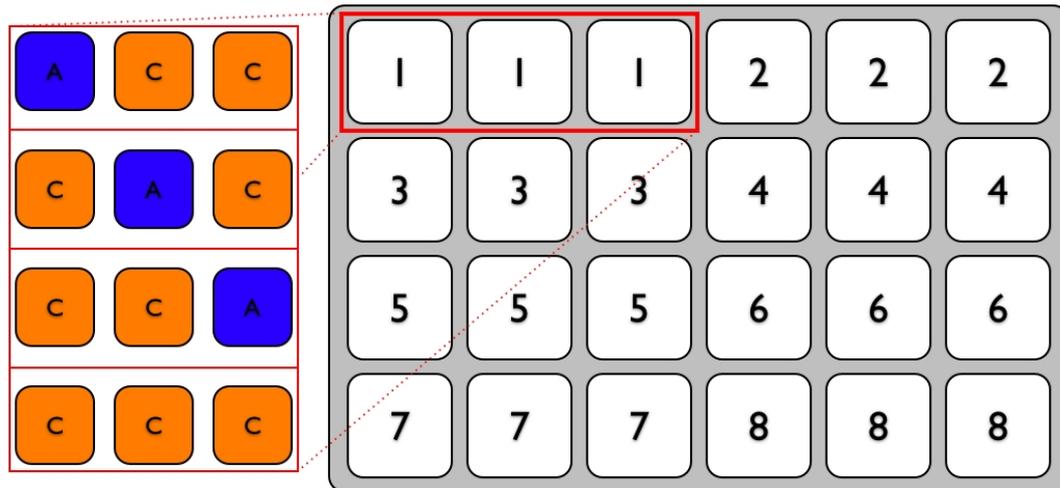


Figure 3.10: Pulsing pattern of the 24-well plate. The well plate was subdivided into triplets of concurrently pulsed wells. Triplets were pulsed in sequence of their numeration. One acute pulse protocol (blue box) could be assigned to any well of the plate, leaving the rest of the plate in chronic (orange boxes) stimulation mode. This is exemplified for one triplet (left part of the figure).

3.2.3 Solid State Relay Device

Conception

The solid state relay device (SSRD) contained the electrical circuit for applying electrical stimulation to each individual well of the 24-well plate. Due to the design of the lid (see Section 3.2.5), a maximum of 24 wells could be pulsed independently, requiring 48 SSR's in total.

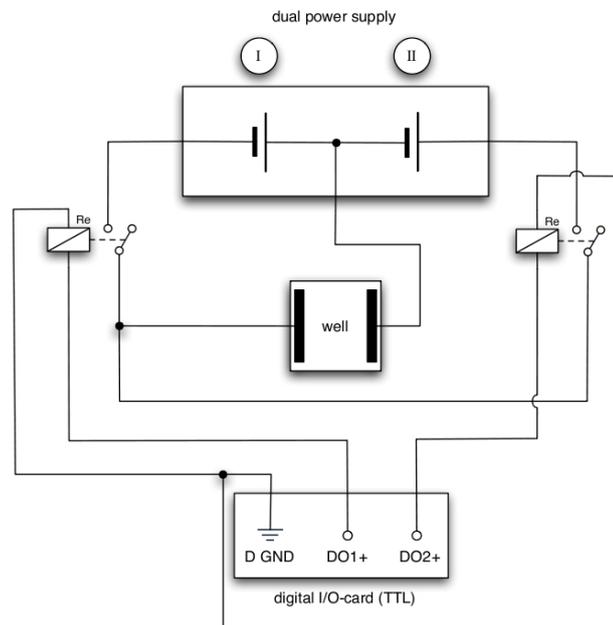


Figure 3.11: Scheme of the circuit diagram used for each well of the plate. The two relays were controlled by the digital I/O-card and used to generate electrically alternating pulses.

In order to alleviate the replacement of malfunctioning SSR's, the SSRD was constructed in a modular manner: with a single mainboard and 48 individual SSR modules. To avoid loose cables and to minimise the amount of time-consuming soldering work, the mainboard and the SSR modules were manufactured as printed circuit boards. The SSR modules were plugged onto the mainboard, as shown in Figure 3.12. The arrangement and labelling of the SSR modules corresponded to the GUI of the stimulation software (see Section 3.2.6), fostering an easy identification of malfunctioning SSR's.

Components

Table B.2 (see appendix) shows the components that have been used for the SSR modules and the mainboard of the SSRD and describes the corresponding task.

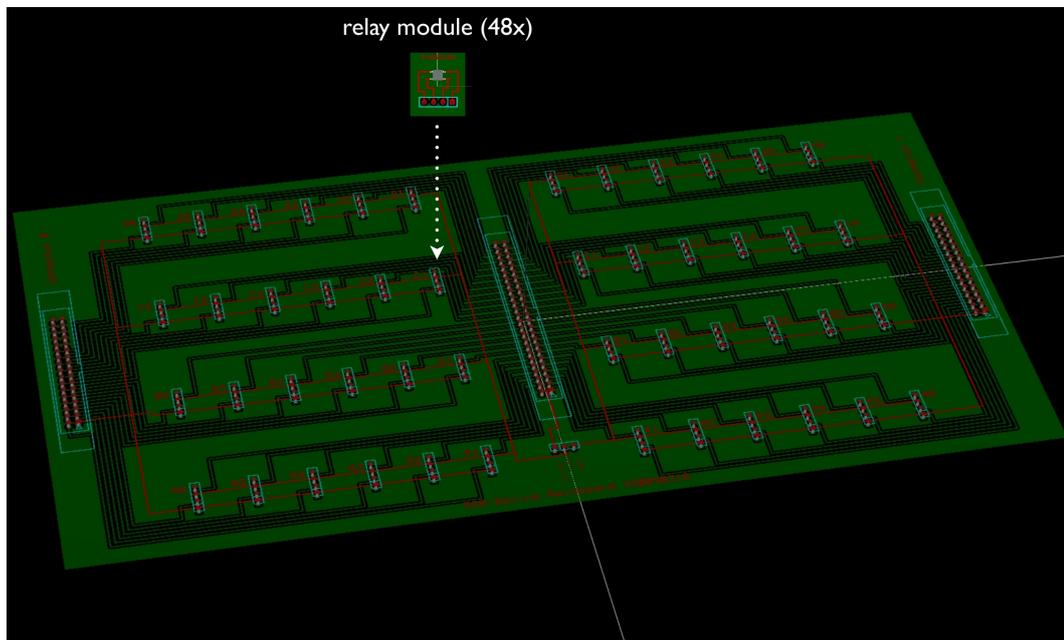


Figure 3.12: 3D-model of the circuit of the SSRD. As shown in the figure, the SSR modules were plugged onto the mainboard, which enabled an easy exchange.

Mainboard and SSR modules

The SSRD had been designed with the open source software KiCad (see Section 2.6.1). The construction plan depicted in Figure 3.13 shows the layout of the mainboard (top) and the SSR modules (bottom). The mainboard was designed as a double-layer board, in order to avoid overlapping conductive paths. Figure 3.13 shows a view onto the component side (darker lines), while the grey lines depict the soldering side. Since the SSR modules were less complex, they have been designed as single-layer boards.

The pin layout of the control voltage circuit is shown in Table B.3 (see appendix). The wiring scheme³ for one well is shown in Figure 3.11.

³The data of the conductive paths as well as the drilling data have been exported using the Gerber file format (see Section 2.6.1). After checking the files for correctness using the software GC-Prevue (see Section 2.6.2), the company Beta LAYOUT GmbH [16] (Aarbergen, Germany) then used this data to manufacture the boards.

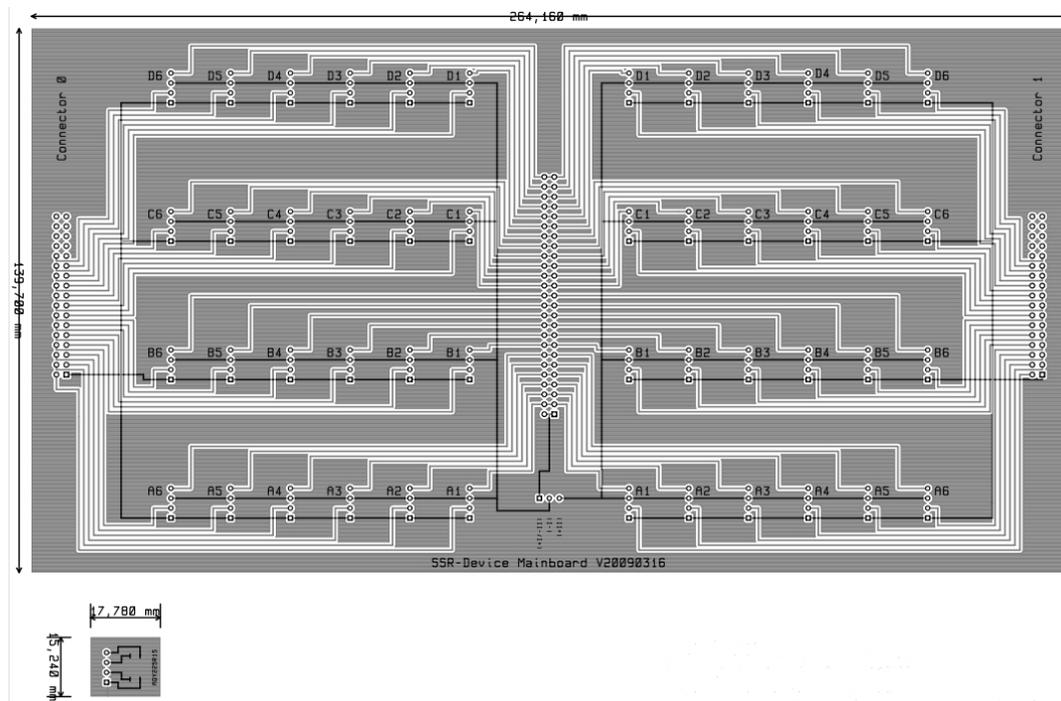


Figure 3.13: Construction plan of the SSRD mainboard (top) and SSRD modules (bottom). Black lines represent the component layer, grey lines represent the soldering layer.

3.2.4 Patch Bay

One of the design concepts for the hardware for electrical field stimulation was modularity. The SSRD (see Section 3.2.3) can be considered as such a module. As can be seen in Table B.3 (see appendix), it had a defined I/O hardware interface. In order to enable utilisation of lids with an arbitrary pin layout without modifying the SSRD, I developed a small device, the patch bay. It allowed a convenient adaption of the SSRD output voltage interface to an arbitrary conductive path layout on the lid. Patch wires could easily be reconnected via push-button clamps allowing arbitrary rerouting of the output voltage. Figure 3.9 shows the integration of the patch bay into the system. The components required for the patch bay are listed in Table B.4 (see appendix). The routing layout is shown in Table B.3.

3.2.5 HCS Stimulation Plate

Polycarbonate blank

The base of the lid was a piece of transparent and thermostable polycarbonate, which is depicted in Figure 3.14. Milled slots permitted the mounting of 48 carbon electrodes as described in Section 2.1.1. Holes for the perfusion, the pipetting needle and for the mounting of stainless steel clamps completed the polycarbonate blank. A schematic drawing of the arrangement of the drill-holes is shown in Figure 3.14.

Stainless steel clamps

The stainless steel clamps were used to hold the carbon electrodes by inducing a mechanical tension between the carbon electrodes and the polycarbonate of the lid. They were screwed to the polycarbonate lid by using sheet-metal screws and the corresponding drill-holes depicted in Figure 3.14.

Electrical circuit paths

The blueprint of the imprinted electrical circuit paths is shown in Figure 3.14: large rectangles representing the maximised contact surface for the stainless steel clamps allowed a reliable conduction of the current. Two pins of the 50 pin ribbon cable interface were used for electrical mass and connected to one carbon electrode per well. The remaining 48 pins were paired and connected to the second electrode, respectively. This ensured, that each carbon electrode was connected to two pins in order to provide redundancy. The pin assignment is listed in Table B.3 (see appendix). To protect the lid against short circuits, the electrical circuit paths and the exterior of the stainless steel clamps have been electrically isolated using the protective paint described in Subsection 2.1.1.

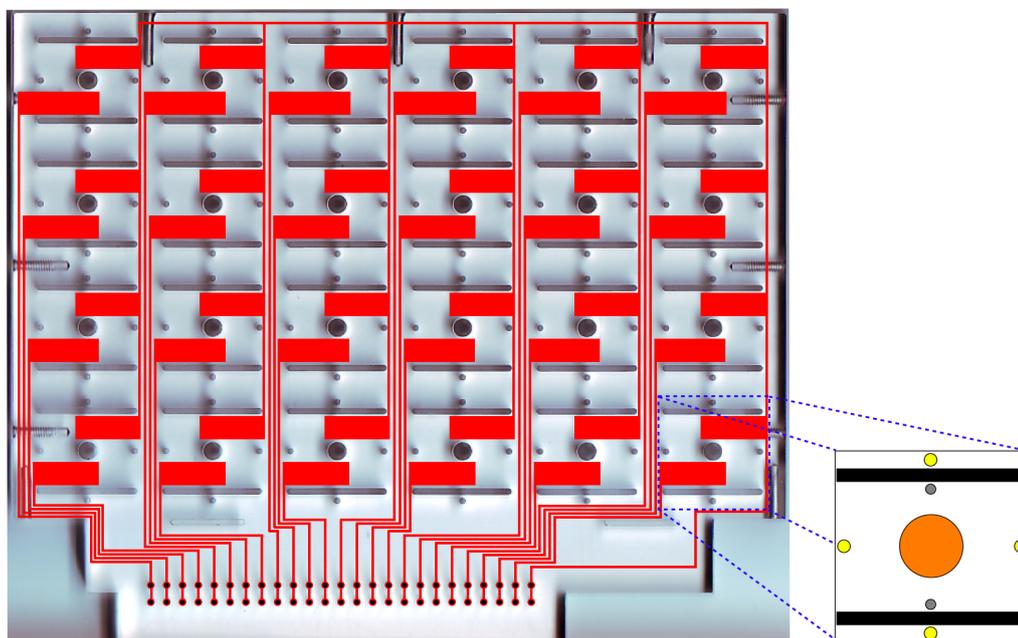


Figure 3.14: Blueprint of the circuit paths with maximised contact area (red rectangles) for the stainless steel clamps. The highlighted section shows the drill-holes for the perfusion (yellow), pipetting needle (orange) and the stainless steel clamps (grey) for the carbon electrodes (black).

3.2.6 Software for Electrical Field Stimulation: Pulsing Device

The Pulsing Device software was designed to enable two things: (i) programming of pulse protocols and (ii) assigning such pulse protocols to an arbitrary number of wells. The development of the software greatly benefited from previous work with respect to usability (ClickIT) and real-time behaviour (TriggerIT and PROTOzoon).

Defining pulse protocols

Stimulation regimes were defined in a pulse protocol using tab-separated values. The GUI of the pulse protocol editor provided fields for a user-friendly adjustment

of the duration (the period of time the pulse train is active), frequency (the number of pulses per second), voltage (the pulse amplitude) and pulse width (the duration of a single pulse). For repetitive pulse trains, a pause could be inserted between the different runs. The pulse protocol editor allowed to store pulse protocols, which could be reloaded and modified.

Assigning and executing pulse protocols

The assignment of a pulse protocol to an arbitrary well or to a group of wells was performed in the main GUI of the Pulsing Device software (see Figure 3.15).

The stimulation plate editor in the centre of the GUI allowed to select a pre-defined template of an HCS stimulation plate, that was graphically displayed accordingly. Plates with one, four or 24 wells were supported. Each well was independently associated with a pulse protocol by activating the checkbox labelled with ".pp". It was thus possible to change pulse protocols of individual wells, while all other wells remained in their current stimulation mode. Furthermore, wells that were supposed to be incorporated in the pulsing process could be selected by activating the checkbox labelled with "ON/OFF".

General information about the pulse protocol and the overall setup were displayed in the info field to the left of the stimulation plate editor. An info field to the right of the stimulation plate editor displayed the names of the pulse protocols assigned to the given well number.

The control field at the bottom of the GUI allowed to take influence on the pulsing process in real-time. Experiment settings of an entire well plate could be stored as XML file.

The Pulsing Device software provided two different pulse modes: one mode allowed the assignment of arbitrary pulse protocols to individual wells, permitting a maximum of 24 different pulse protocols. Due to the fact that a pulse protocol may contain pulse width and frequency values which might cause an overlapping with pulse trains in other wells, a power supply with an amperage of 6 A (24×250 mA) may be required in the worst case.

In contrast, the second mode allowed the assignment of only two different pulse

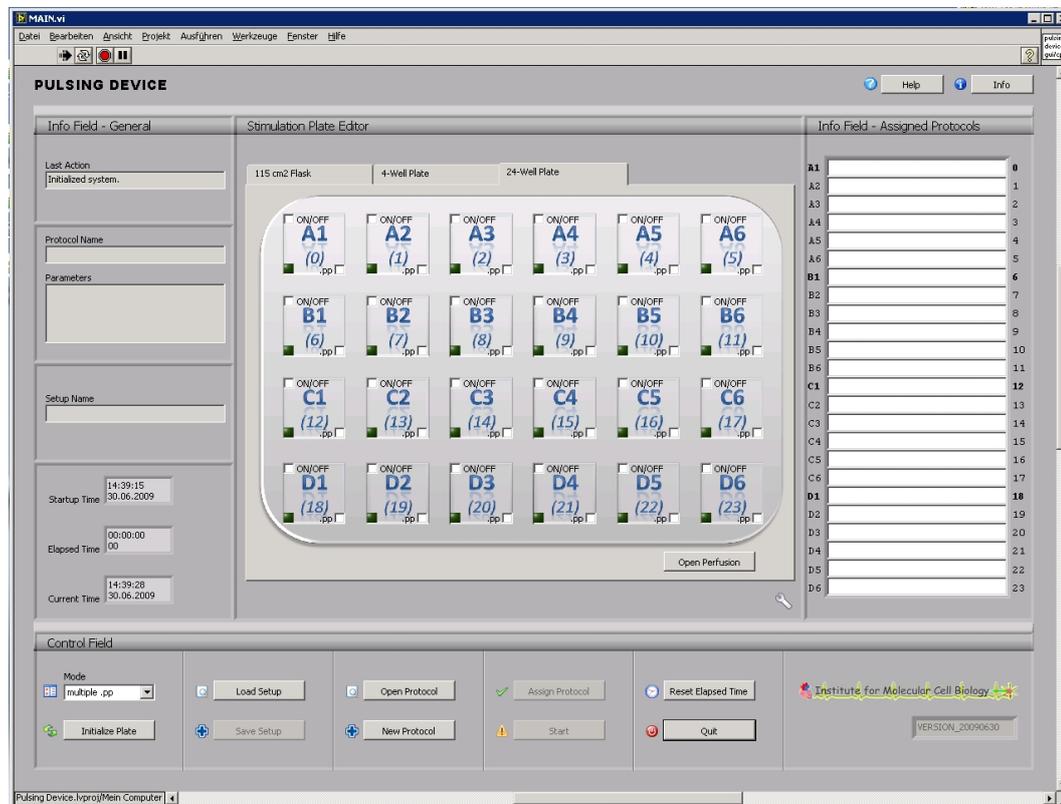


Figure 3.15: GUI of the Pulsing Device software. It was divided into an info field displaying general information about the setup and the pulse protocol (left), an area which allowed the selection of predefined stimulation plate templates as well as the selection and editing of specific wells (centre), an info field displaying the protocol names that had been assigned to a specific well (right) and a control field, which allowed to actuate a pulse experiment in real-time (bottom).

protocols: a chronic and an acute pulse protocol. The chronic pulse protocol ran as the default protocol in all 24 wells while the acute protocol was only active for the well of interest. This mode can be performed using the power supply described in Section 3.2.2. Nonetheless, due to the modular design of the hardware, the power supply can easily be exchanged, thus enabling the full spectrum of experiments.

Implementation details

The functionality of the Pulsing Device software has been implemented using LabVIEW and wrapped into a DLL that could be executed on any PC running Microsoft Windows. The GUI⁴ (see Figure 3.15) allowed the use of the Pulsing Device software as a stand-alone application. Due to the clear arrangement of the control elements emphasised by suitable icons, a high degree of usability was provided.

By using the producer-consumer design pattern (see Section 2.5.1), the software layer could be decoupled from the hardware layer while maintaining real-time accessibility of the entire system. The modular software architecture allowed the optimisation and modification of individual components. Especially the VI that executed the pulse trains by repeatedly calling the sub-VI that generated the alternating negative / positive pulse has been optimised. Each well participating in a pulsing process was running in its own thread. Active wells were marked in the GUI by a highlighted indicator. Once a well had finished its assigned pulse protocol, the corresponding well number was sent back to the GUI in order to make the well available again. This led to a deactivation of the corresponding indicator and the corresponding thread was terminated. Once all well-based threads had been terminated, the overall pulsing process finished and the Pulsing Device software automatically stopped. The different modules of the Pulsing Device software, their interplay as well as the direction of information flow between the different components are shown in Figure 3.16.

Program logic

The transition from the software layer to the hardware layer was implemented by tasks, that were defined using the MAX software described in Section 2.5.1. A task allows to write digital data to a single digital line or to multiple digital lines on a specified port of the data acquisition board. As shown in Figure 3.9, the pulse trains were sent from the digital I/O-card to two SCC-68 connector blocks, one

⁴The GUI has been optimised for a resolution of 1280×1024 pixels, which is the default resolution for most 19 inch flat panel displays.

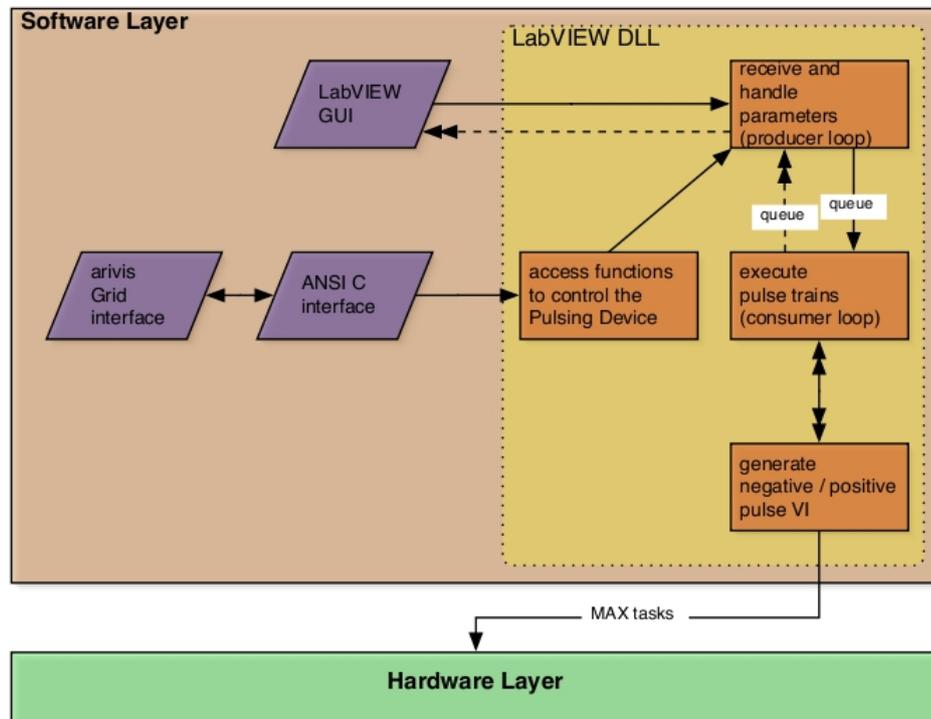


Figure 3.16: Architecture of the Pulsing Device software: wrapped into a DLL, the functionality of the software (orange boxes) could be accessed from the GUI, the ANSI C interface or from the arivis Grid interface (purple boxes). Arrows denote the direction of information flow. Double arrows denote multiple calls / responses of the same VI. Software- and hardware layer were decoupled using the producer-consumer design pattern in order to maintain real-time accessibility.

responsible for the negative and one for the positive voltages.

Software interfaces

Apart from the GUI, the Pulsing Device software could also be controlled via an ANSI C software interface, as shown in Figure 3.16. This software interface was implemented using the ANSI C programming environment LabWindows/CVI (see Section 2.5.2). Since CVI uses the same libraries as LabVIEW, it bridges the gap between LabVIEW and ANSI C. The ANSI C software interface thus provided

the same functionality as the stand-alone version of the Pulsing Device software. This enabled the possible integration of the software into large screening software packages. Each software that supports an ANSI C interface can control the Pulsing Device software by making use of the functions defined in the ANSI C interface. In this case, the ANSI C interface entirely replaces the GUI. Thus, the stand-alone module can also be considered as a self-contained module within an HCS system. A detailed description of the ANSI C interface is given in the Pulsing Device API, which can be found in Appendix A.1.

The third interface of the Pulsing Device software was the arivis Grid interface. It allowed the registration of the Pulsing Device as a grid client and thus enabled a communication with other grid clients. Hence, the Pulsing Device could be controlled from any grid client that is capable of sending the following grid commands:

- `Configure <int CompartmentID>`
`<string ProtocolID><bool isAcute?>`: configured the HCS stimulation plate by assigning arbitrary pulse protocols to individual wells
- `SwitchPulseProtocol <string OldProtocolID>`
`<string NewProtocolID>`: replaced an old pulse protocol by a new one
- `StartPulsing`: started the pulsing process
- `StopPulsing`: stopped the pulsing process
- `UpdatePulsingProcess`: updated variables used by the Pulsing Device DLL
- `SetVoltageChange <double deltaV>`: increased or decreased the stimulation voltage by $\pm\Delta V$
- `StartPerfusion <int CompartmentID>`: started the perfusion inside the specified well (see next section)
- `StopPerfusion`: stopped the perfusion system (see next section)

The Pulsing Device grid client was capable of receiving these grid commands and calling the corresponding functions of the ANSI C interface described above. Figure 3.16 depicts the interplay between the arivis Grid interface, the ANSI C interface and the Pulsing Device functionality of the DLL code.

3.2.7 Perfusion: Mounting Plate and Control Unit

Each of the 24 pinch valves of the mounting plate (see Section 2.2.1) was connected to an LED displaying the state (open / closed). This enabled the user to follow the state of the hardware independently of the software. The wiring scheme for one valve of the mounting plate and the control unit of the perfusion system is depicted in Figure 3.17. A PC was running the Perfusion System software (see Section 3.2.8), which controlled the 24 relays. It was connected to the USB relay modules (see Section 2.2.2) via an USB hub and allowed to open and close the entire electrical circuit. If the pinch valve was switched to open, the corresponding LED was enlightened and the gravity driven perfusion was activated. The connector pin assignment of the control unit is summarised in Table B.5 (see appendix).

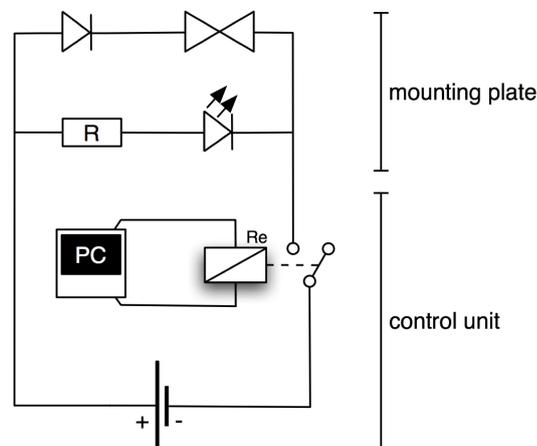


Figure 3.17: Wiring scheme for one pinch valve of the perfusion system.

3.2.8 Software for Application of Test Substances: Perfusion System

The Perfusion System software allowed to control the aforementioned perfusion hardware. It was implemented in LabVIEW and ran on any Microsoft Windows PC. Besides a GUI, it also provided an ANSI C interface and an interface for the arivis Grid. Additionally, the Perfusion System software could be started from within the Pulsing Device software. The API of the Perfusion System software is described in Appendix A.2.

Perfusing a well

Once the well had been connected with the corresponding pinch valve via a tube, the perfusion of the well could be started by either selecting the corresponding well in the GUI or by setting the corresponding well number in the ANSI C interface. Valid well numbers were within the range of [0; 23]. Thus, each well of the 24-well plate could be perfused one at a time. Selecting “Perfusion off” in the GUI or passing the value 24 to the ANSI C interface stopped the perfusion.

3.3 Data Processing and Analysis

3.3.1 An Open Image Analysis Toolbox for Fluorescence and White Light Imaging: CellAnalyser

General concept

In an HCS environment, image analysis software for both, online and post-screening analysis, should meet several demands: (i) the design of the software should be modular to enable easy enhancement of the software’s capabilities; (ii) the software should be accessible via different software interfaces such as a GUI, a command line interface (CLI) or an API to support different application environments; (iii) the choice of the programming language (at least for the API) should foster a fast and easy integration of the software into the HCS environment; (iv) with respect

to automated detection of image objects, the detected regions of interest (ROI's) should be passed to an analysis tool for further investigation.

The CellAnalyser software developed here has been implemented to cope with the above mentioned demands. Three different operation modes enabled the application of CellAnalyser adapted to different contexts:

(1) manual mode:

The GUI and the CLI allowed the use of CellAnalyser as a stand-alone tool for post-screening analysis. In this mode, automated cell detection was by-passed and ROI's had to be drawn completely by hand. This allowed the analysis of arbitrary image objects or image objects, that required the expertise of a scientist, e.g. because they were extremely difficult to distinguish from surrounding objects. Tools for drawing different ROI shapes such as the polygon, freehand or the ellipse tool were provided.

(2) semi-automatic mode:

In this mode, ROI's were detected by the algorithm and displayed on screen for interactive management. The user could decide to delete or add ROI's by hand using the interactive GUI or the MatVis interactive visualisation toolkit.

(3) automatic mode:

If integrated into an experimental environment, the API allowed the remote usage of CellAnalyser for analysing image data online (for details see next subsection). In this mode, analysis was not interrupted by interactive steps. Figure 2.11 shows the exemplified integration of the CellAnalyser software into the HCS environment.

Implementation

CellAnalyser has been implemented in MATLAB (see Section 2.5.7) and was used on computers running Mac OS and Microsoft Windows (32-bit and 64-bit, respectively). It was based on a modular design referred to as "open toolbox", which is

further described below.

Figure 3.18 shows an overview of the entire software concept of CellAnalyser: input data could either be fluorescence image stacks (*.tif* files or *.sis* files (with the aid of the ABMI described in Section 2.8.3)) or single white light images (all common file formats were supported). The selection of multiple image files was possible.

CellAnalyser could be controlled from different software interfaces: the GUI was used for post-screening analysis and allowed to select among different experiment scripts and operation modes (see above). Amongst others, parameters for background correction, analysis details and visualisation could be selected. The CLI allowed to call specific analysis routines directly from the MATLAB command window. It could be used for rapidly testing new analysis routines or to test and assemble a new experiment script. The arivis Browser plug-in enabled the execution of an experiment script directly from the arivis Browser by using its MATLAB Interface plug-in (see Section 2.8.2). In summary, each interface permitted the execution of an experiment script, which itself made use of the functions available in the toolbox.

Output data for the white light and for the fluorescence mode was written to disc using the comma separated values file format (*.csv*). ROI's were stored using the MATLAB matrix file format (*.mat*). Both file formats permitted usage of the stored values for further computations.

The open toolbox framework

The different analysis routines were grouped according to their field of application. This enabled the user to easily choose a specific function and plug it into the experiment script in order to tackle a biological problem. With this strategy, individual experiment scripts could be generated. In addition, users with knowledge in MATLAB could easily add their own functions to the toolbox and thus make them usable within the experiment script layer. This open design made CellAnalyser a dynamic, versatile and thus flexible tool for answering a diversity of biological questions related to white light or fluorescence image data. The toolboxes con-

tained functions for image processing and analysis, data analysis as well as for data representation and storage.

Image processing and analysis:

This toolbox basically provided two major functionalities: (i) removal of unwanted structures such as dust particles or scratches in the bottom of the well plate from the image and (ii) automated detection of ROI's. Therefore, the following functions were included:

- subtraction of background images
- elimination of noise by averaging a specified number of images
- intensity-based thresholding to separate darker from brighter regions and to thus exclude unwanted objects
- edge detection (see Section 2.9.2) and marker-based watershed transformation algorithms (see Section 2.9.1) for image segmentation
- computation of black-and-white masks for each object of interest
- size-dependent filtering
- storage and export of ROI's

Data analysis:

The data analysis toolbox for the fluorescence mode included the following functions:

- computation of the sum of intensity values for each ROI of a fluorescence image stack
- ratio-based peak detection
- computation of baselines and the corresponding peak duration at a given threshold

Data representation and storage:

Functions contained in this toolbox were used for data visualisation (e.g. for plotting calcium traces or labelling cell images). Additionally, import and export of ROI's and their corresponding intensity values was handled here.

The above mentioned functions were used to create custom-made experiment scripts. CellAnalyser provided script templates, which can be used as starting point for generating new scripts. Experiment scripts for analysing white light and fluorescence images of cardiomyocytes and erythrocytes are introduced in Section 3.3.2 and Section 3.3.3.

Summary

Flexibility in the field of application and a user-friendly handling of the software contribute to answering the variety of biological questions in a time-efficient manner. For this, CellAnalyser followed a modular design pattern, which allowed an easy enhancement as well as an easy adaption to a variety of biological problems. Proper segmentation of strongly overlapping or clustered cells is still a challenge for most image analysis software. Nevertheless, the interactive operation modes of CellAnalyser allowed to circumvent this problem and enabled the analysis of arbitrary types of biological objects present in image data. This point will be discussed further in Section 4.2.1.

3.3.2 Use Case I: Automated Detection and Analysis of Cardiomyocytes**Data input**

For white light and fluorescence imaging experiments cardiomyocytes had been used as described in Section 2.10.1. White light images had a resolution of 1388×1038 pixels, fluorescence images had a resolution of 172×128 pixels with 9000 images per recording. Each image stack contained recordings of two channels with different excitation wavelengths (channel 1: 340 nm, channel 2: 380 nm)

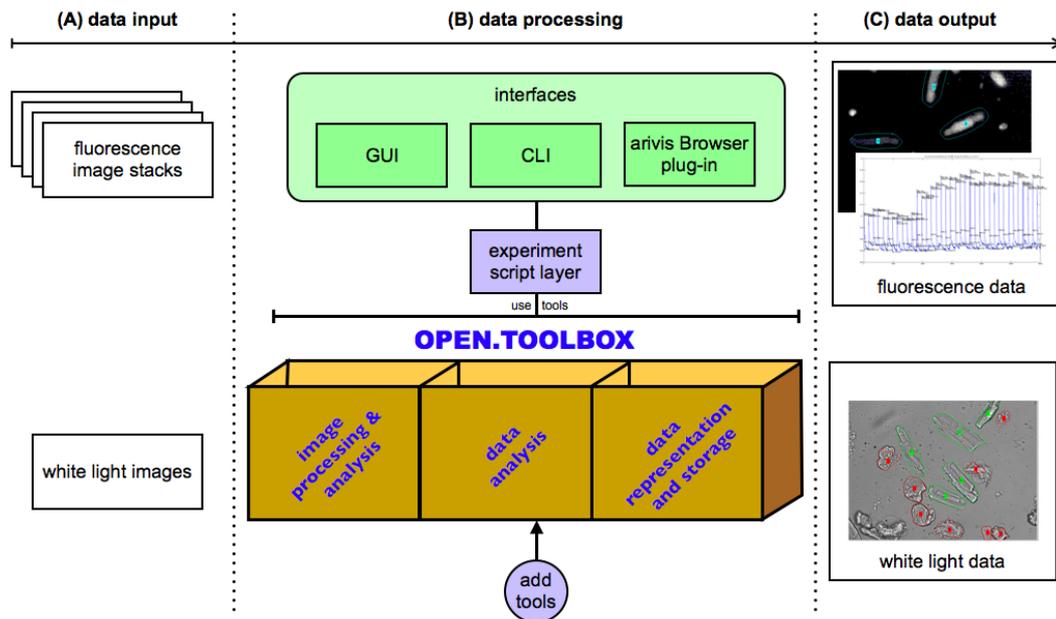


Figure 3.18: Overview of the software concept of CellAnalyser: (A) Input of white light or fluorescence image data. (B) Software interfaces, experiment script layer and the “open toolbox” concept. (C) The results of the analysis: traces of calcium transients (fluorescence approach) and classified cell shapes (white light approach).

in an alternating manner, resulting in 4500 images per channel. Additionally, a background image containing 300 images for each channel was used.

Data processing

Detection criteria

According to their shape, cells in white light images were classified into the groups “brick-like” (healthy cells), “round” (putative dead cells) and “unclassified” (e.g. clusters of heavily overlapping cells, that could not be separated by the

algorithm). The number of members of each group was counted and stored.

Applying the CellAnalyser software to fluorescence images, calcium transients in healthy cells captured in two-channel fluorescence image stacks had been computed. Integrated intensity values for each ROI were calculated according to Equation 3.2:

$$FR = \frac{\sum_{i=1}^n CH1PI_i}{\sum_{i=1}^n CH2PI_i} \quad (3.2)$$

where

FR: fluorescence ratio

CH1: channel 1

CH2: channel 2

PI: pixel intensity value

i: pixel number

Detection parameters

After the preprocessing step and the watershed transformation as described in Section 3.3.1 the image objects were segmented. To exclude small artefacts as well as clusters of cells, CellAnalyser first applied a size-dependent filtering method. Filtering was done by setting thresholds for the minimal and the maximal cell size.

Discrimination of brick-like and roundish cells was done using shape-dependent filtering. Here, each object was enclosed by an ellipse. The length of the ellipse's major axis was divided by the length of its corresponding minor axis. If this value was close to 1, the two axes had the same length and thus the corresponding cell had a roundish shape. Otherwise, the shape of the cell was classified as brick-like.

Cell classification and analysis

By combining size-dependent and shape-dependent filtering, suitable cells were detected. While the white light approach was used to distinguish between the different cell shapes, the fluorescence approach was used to perform further analysis on brick-like cells.

The following pseudo code describes the program sequence of an experiment script of the white light approach:

Algorithm 1 WHITE LIGHT APPROACH (CARDIOMYOCYTES)

```

1:  $I \leftarrow \text{readImage}(\text{filename})$ 
2:  $g \leftarrow \text{computeGradientMagnitudes}(I)$ 
3:  $BW \leftarrow \text{computeBWMask}(I)$ 
4:  $L \leftarrow \text{watershedSegmentation}(BW, g)$ 
5:  $LC \leftarrow \text{removeSmallBasins}(L)$  {size-dependent filtering (min)}
6:  $LCBW \leftarrow \text{lableMatrix2BW}(LC)$ 
7:  $CM \leftarrow \text{separateCellMasks}(LCBW)$ 
8:  $\text{sizeThreshold} \leftarrow \text{assign value}$ 
9:  $\text{shapeList} \leftarrow []$ 
10: if  $|CM| > 0$  then
11:   for each  $cm \in CM$  do
12:      $\text{size} \leftarrow \text{numberOfPixels}(cm)$ 
13:     if  $\text{size} > \text{sizeThreshold}$  then
14:       {size-dependent filtering (max)}
15:        $\text{shape} \leftarrow \text{"unclassified"}$ 
16:     else
17:       {shape-dependent filtering}
18:        $\text{majAxis} \leftarrow \text{ellipse}(cm)$ 
19:        $\text{minAxis} \leftarrow \text{ellipse}(cm)$ 
20:       if  $(\text{majAxis}/\text{minAxis}) \approx 1$  then
21:          $\text{shape} \leftarrow \text{"round"}$ 
22:       else
23:          $\text{shape} \leftarrow \text{"brick-like"}$ 
24:       end if
25:     end if
26:      $\text{shapeList}[cm] \leftarrow \text{shape}$ 
27:   end for
28: end if
29: return  $\text{shapeList}$ 

```

The program sequence of an experiment script of the fluorescence approach is described by the following pseudo code:

Algorithm 2 FLUORESCENCE APPROACH (CARDIOMYOCYTES, 2 CHANNELS)

```

1:  $[I, IB] \leftarrow \text{readImageStacks}(\text{image}, \text{backgroundImage})$ 
2:  $[I1, I2, IB1, IB2] \leftarrow \text{separateAlternatingChannels}(I, IB)$ 
3:  $[I1', I2'] \leftarrow \text{subtractBackground}(I1, I2, IB1, IB2)$ 
4:  $R \leftarrow \text{computeROIs}(I2')$  {detect brick-like cells similar to the white light approach}
5:  $\text{durationThreshold} \leftarrow \text{assign value}$ 
6:  $\text{resultList} \leftarrow []$ 
7: if  $|R| > 0$  then
8:   for each  $r \in R$  do
9:      $IV1 \leftarrow \text{sumIntensity}(r, I1')$ 
10:     $IV2 \leftarrow \text{sumIntensity}(r, I2')$ 
11:     $\text{ratio} \leftarrow IV1./IV2$ 
12:     $P \leftarrow \text{computePeaks}(\text{ratio})$ 
13:     $B \leftarrow \text{computeBaselines}(P)$ 
14:     $A \leftarrow \text{computeAmplitudes}(P, B)$ 
15:     $D \leftarrow \text{computeDurations}(A, P, \text{durationThreshold})$ 
16:     $\text{resultList}[r] \leftarrow [P, A, B, D]$ 
17:   end for
18: end if
19: return  $\text{resultList}$ 

```

Data output

As depicted in Figure 3.18, the output of the white light approach was visualised as an image, where cells were labelled according to their shape: brick-like cells were labelled green, round cells were labelled red, unclassified cells were labelled yellow (not shown). The number of members of each category was written to disc and could e.g. be used for automated quality control of the cell isolation.

The output of the fluorescence approach was a plot showing the time course of a calcium transient (also see Figure 3.18). The detected ROI's and the corresponding ratios of intensity values were written to disc.

3.3.3 Use Case II: Automated Detection and Analysis of Erythrocytes

Data input

For this use case, single channel fluorescence images of erythrocytes as described in Section 2.10.2 had been recorded. The image stack had a resolution of 320×240 pixels, comprising 132 images. Besides erythrocytes, the images also contained immature red blood cells, the reticulocytes. In contrast to erythrocytes, reticulocytes may contain ribonucleic acid (RNA), which appears as spots of brighter intensity values if excited with an excitation wavelength of e.g. 633 nm. Therefore, an additional single fluorescence image using this wavelength was recorded.

Data processing

Detection criteria

Cells were classified according to their cell type either as erythrocyte or as reticulocyte. Discrimination was based on the presence or absence of RNA (see above). Therefore, an appropriate intensity threshold had to be chosen. After excluding the reticulocytes from the superset containing all round objects identified in the image, analysis using only erythrocytes was performed.

Detection parameters

To detect only single blood cells and to thus exclude clusters of overlapping cells from the analysis, appropriate values for the minimum and maximum radius of a cell had to be chosen. These values were used for the circular Hough transform (see Section 2.9.3), which was applied to detect only cells lying within this range.

Cell classification and analysis

By combining the above mentioned intensity-based and size-dependent thresholds, an analysis solely based on erythrocytes was performed. The goal was to observe the fluorescence signal of single cells over time. Analysis of the responsiveness of erythrocytes with respect to a chemical stimulus (e.g. application of lysophospha-

tidic acid) was done using Equation 3.3:

$$F_{sig} = \frac{\Delta F}{F_0} \quad (3.3)$$

where

F_{sig} : calcium-related fluorescence signal

ΔF : change in fluorescence

F_0 : initial fluorescence level

F_{sig} describes the change of the calcium-related fluorescence signal over time with respect to its initial value. The analysis is described by the following pseudo code:

Algorithm 3 FLUORESCENCE APPROACH (ERYTHROCYTES, 1 CHANNEL)

```

1:  $I \leftarrow readImageStack(filename1)$ 
2:  $IS \leftarrow readSingleImage(filename2)$ 
3:  $R \leftarrow detectReticulocytes(IS)$  {intensity-based threshold}
4:  $C \leftarrow detectCircularShapes(I)$  {circular Hough transform}
5:  $E \leftarrow excludeReticulocytes(C, R)$   $\{C \setminus R\}$ 
6:  $resultList \leftarrow []$ 
7: if  $|E| > 0$  then
8:   for each  $e \in E$  do
9:      $IV \leftarrow sumIntensity(e, I)$ 
10:     $\frac{\Delta F}{F_0} \leftarrow compute\frac{\Delta F}{F_0}(e, IV)$ 
11:     $resultList[e] \leftarrow \frac{\Delta F}{F_0}$ 
12:   end for
13: end if
14: return  $resultList$ 

```

Data output

In addition to the number of detected erythrocytes, the corresponding intensity values and the $\frac{\Delta F}{F_0}$ values were written to disc. Figure 3.19 shows a collection of

plots for 25 erythrocytes. Each panel could be enlarged individually for further investigations.

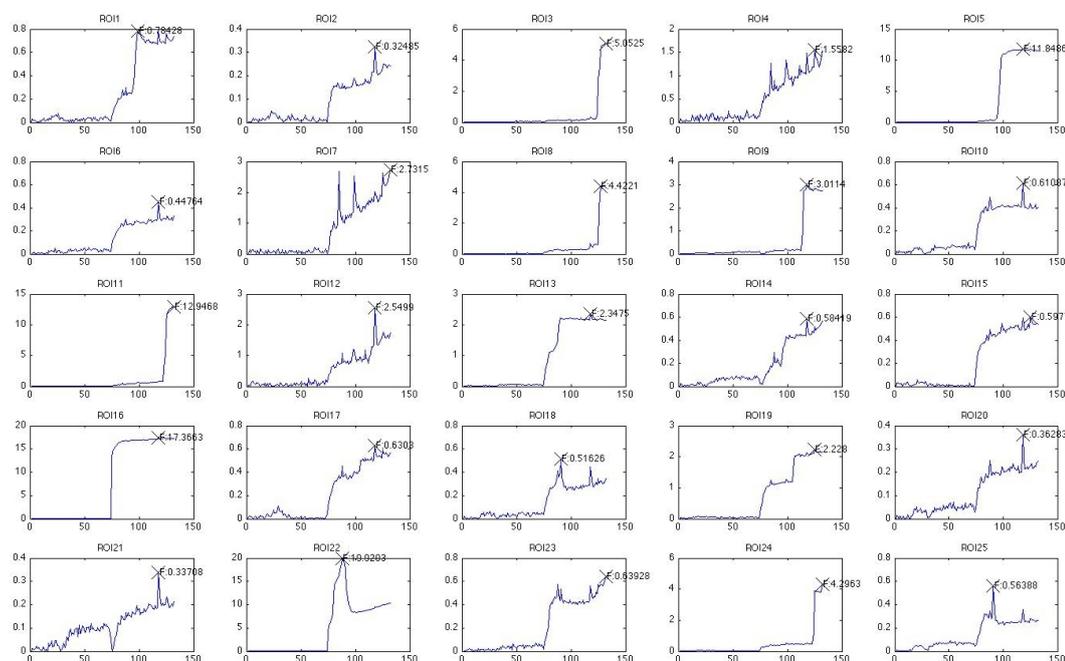


Figure 3.19: CellAnalyser: resulting $\frac{\Delta F}{F_0}$ plot for 25 erythrocytes.

3.4 Data Storage, Archiving and Management

3.4.1 Software for Automated Import of Images into OMERO: AuTO.iMporter (ATOM)

General concept

For storing and managing our image data we used the OMERO platform (see Section 2.7.1), which allowed manual image import with the OMERO client OMERO.importer. But manual import of image data has several drawbacks: (i) the user has to keep track of the image file transfer himself; (ii) the import process

cannot start until the file is closed. Manual import may thus lead to an increased booking time of the experimental setup; (iii) if multiple images are recorded during an experiment, manual import can either be done after each recording period or at the end of the experiment. While the first solution may lead to an interrupted workflow, the latter import process can be quite time-consuming, depending on the data volume as well as on the quality of the connection between client and server. In our lab, for instance, data volumes of 15 to 20 GB per week are being generated, of which the more™ is contributing about 8 GB.

From the perspective of the user, an automated import process would circumvent many of those problems. Ideally, the import process should run in the background during an imaging experiment, without negatively affecting the data acquisition process. Furthermore, the necessity for user interactions should be kept to a minimum. In order to add this functionality to OMERO, I have developed and implemented the automated OMERO add-on AuTO.iMporter (ATOM). It used the OMERO.importer API and was thus able to handle all file formats supported by the Bio-Formats library.

A prerequisite for importing data into OMERO is the assignment of both, a project- and a dataset name. Without this, data cannot be imported, neither by the OMERO.importer nor by using the API. To pre-organise acquired image data, ATOM allowed the specification of project- and dataset names. If no import destination was specified, images were assigned to a default project, while the dataset name was derived from the folder structure of the monitored image directory. Both could be rearranged later by using the OMERO.insight client.

ATOM provided two operation modes: the *manual mode* allowed to comfortably import post-screening image data that were stored on an USB hard disc or a DVD, for instance. This mode can be used for importing data acquired earlier.

The *automatic mode* was used during data acquisition. In this mode a predefined image directory could be monitored for new data content. The monitoring process as well as the import process did not interfere with data acquisition. During each monitoring cycle, ATOM generated a snapshot of the content of the image directory for keeping track of files that had been added or that did change since the

last import process. The possibility of defining additional file criteria (see below) in combination with the use of standard Java routines for file handling enabled ATOM to avoid the import of incomplete data.

The program sequence including the above mentioned operation modes is depicted in Figure 3.20.

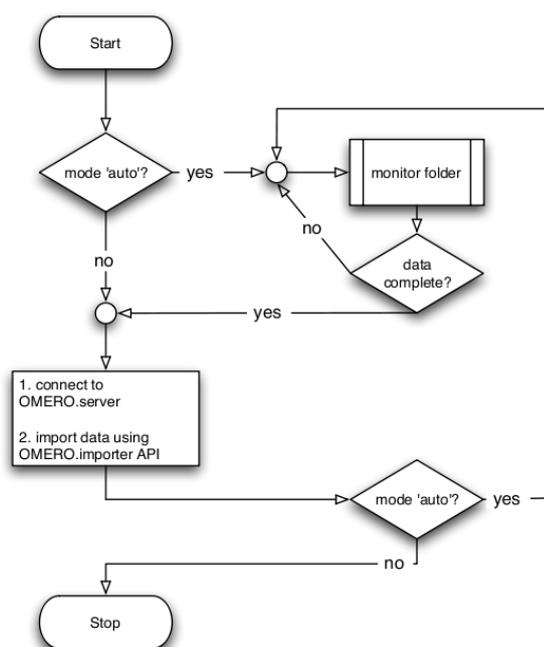


Figure 3.20: Flowchart of the ATOM software.

Implementation

ATOM has been implemented in Java (see Section 2.5.5). Due to its platform independency, ATOM could run in a heterogeneous network environment. It used packages and libraries from the OMERO beta-4.1.1 release.

Wrapped in a Java archive (.jar) file, the software could be controlled from the command line. To alleviate the handling, a shell script executing all necessary commands was provided. Within Microsoft Windows this shell script could be

directly executed from the desktop by double-clicking on the icon. The command line parameters of ATOM are summarised and described in Table 3.2.

Table 3.2: Overview of command line parameters of the ATOM software.

Parameter	Description
-a (mandatory)	set the import mode to “automatic”
-m (mandatory)	set the import mode to “manual”
-d (optional)	specified the image directory to be monitored
-raw (optional)	additionally archived data in raw format
--project (optional)	specified the project name
--dataset (optional)	specified the dataset name
--help (optional)	displayed a help screen
--version (optional)	displayed the current program version of ATOM

These optional parameters could be stored in a local property file and were parsed from this file, if not specified in the command line. Otherwise, parameters specified in the command line overwrote parameters defined in the property file. The property file also contained information about connection details of the OMERO server as well as information about the cycle interval for monitoring the image directory.

Classes and members

ATOM

This class was the main entry point of the ATOM software. It contained the main function, which started an instance of ATOM using the selected import mode.

CheckArgumentList

The argument list of the ATOM program call was parsed for parameters according to Table 3.2. Missing mandatory parameters caused an error message informing the user about possible solutions. Missing optional parameters were replaced by parameters specified in the local property file. Defined optional parameters overwrote parameters specified in the local property file.

ManualMode

The manual mode utilised the functionality of the automatic mode by calling members of the `AutoMode` class for one time only.

AutoMode

A while loop kept the monitoring process alive. The cycle could be interrupted by pressing `Control-C` on computers running Microsoft Windows or Linux. On computers running Mac OS the key combination `Command-C` had to be employed.

MonitorFolder

The `MonitorFolder` class contained the basic functionality to monitor a specified directory. Once new or modified files had been detected, the import process was triggered, provided that the files were closed.

SpecialFileCriteria

Many file formats – such as TIFF – consist of a single image file that can be imported once it has been closed. Other formats such as multi-file formats might need a specific treatment in order to correctly import image data and the corresponding metadata. This is the case, if e.g. metadata and image data are stored separately. An example for this is the VisiTech International (Sunderland, United Kingdom) `XYS` file format. Here, metadata are stored in a first file (ending with `.exp`), while the image data are stored in a second file (ending with `.xys`). An `XYS` file is considered as closed, if a third file with the suffix `.html` has been written to disc. This last file contains additional metadata and links to the first two files.

Thus, in order to preserve the relationship between metadata and image data during import, I had to define a rule enabling `ATOM` to wait until the file with the extension `.html` has been created and closed. This rule also caused `ATOM` to ignore the two other files. The definition of a class for such rules was necessary, since the `OMERO.importer` has not been designed for online import. Therefore, `ATOM` provided the functionality for handling the automated import of multi-file

image formats during an imaging experiment. The `SpecialFileCriteria` class might be updated and enhanced by additional rules once treatment of further special file formats is required.

HandleProperties

This class contained routines for handling properties defined in the local property file.

TriggerImport

The connection between ATOM and the OMERO server was initiated by the `TriggerImport` class. Once all information about valid import candidates had been collected and stored in an import list, ATOM connected to the OMERO server using packages and libraries of the OMERO.importer client. After processing the entire import list, the connection to the server was closed again and the monitoring cycle was restarted.

An overview of the different classes of ATOM and their dependencies is given in Figure 3.21.

The current version of ATOM is compatible to the beta-4.1.1 release of OMERO, which uses the Internet Communications Engine (Ice) [134] as the underlying remoting architecture. A previous version, which is compatible to the JBoss-based [58] beta-3 release, also exists. With the beta-4 release, OMERO.fs (OME consortium, Dundee, Scotland), which provides the functionality of a file system monitor, has been introduced. Its first application, OMERO.dropbox (OME consortium), also has the ability to automatically import image files into OMERO. The differences between ATOM and OMERO.dropbox will be discussed in Section 4.2.3.

3.4.2 Network Security and Backup Solution

In our institute, acquisition, analysis and storage of data involved computer systems that were connected over a network. This computer-based workflow required

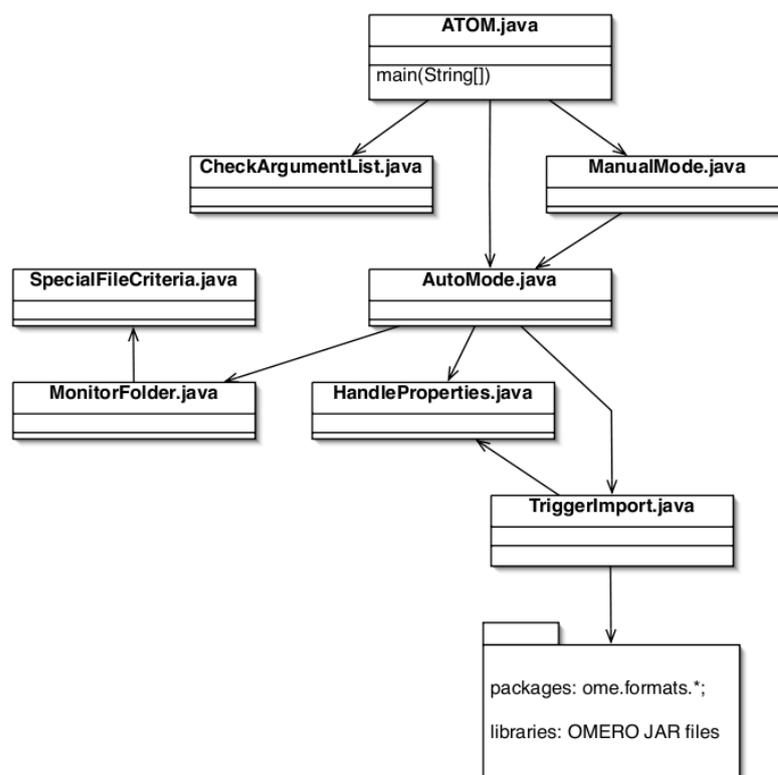


Figure 3.21: Java class dependencies of the ATOM software. Arrows denote “uses” relations.

two major aspects to be fulfilled amongst others: protection against (i) unwanted data access from outside; (ii) unwanted damage or data loss. Both aspects implied careful balancing of data accessibility against data security (aspect (i)) and performance against redundancy (aspect (ii)). An unbalance might inevitably result either in untrustworthy data (i.e. if the protective measures are too weak) or in an inefficient workflow (i.e. if the protective measures are too strong). Therefore, I carefully analysed the network infrastructure of our institute as well as the amount of image data that was being generated. Using this information, I developed a network security and backup solution tailored to our needs. Figure 3.22 shows an overview on the concept, which is explained below.

Protection against unwanted data access

Transferring data from an OMERO client to the OMERO server over a network requires a TCP/IP connection between the two computer systems. A controlled connection can be achieved by a firewall, which is a filter containing access authorisation rules for specific ports on the target system. Without such a firewall, data on the server are unprotected and vulnerable for unwanted access.

Due to the security policy of our campus computing centre (Zentrum für Informations- und Kommunikationstechnik, Homburg / Saar, Germany), the network of our institute was separated from the world wide web (WWW) by two firewalls with a demilitarised zone (DMZ) in between. A DMZ is a subnetwork, which allows providing network services that can be accessed from the WWW. Thereby, access to the network is restricted to computers in the DMZ. Thus, a DMZ can be considered as an additional layer of security.

To provide access to our OMERO server only for staff members and cooperation partners, the eServer, hosting the OMERO server, had been placed inside the DMZ (see Figure 3.22). For client-server communication the TCP port 4063 was unlocked for incoming connections. Once a connection had been established, data could be transferred in both directions. The second firewall separated the OMERO server from its database back-end. The OMERO image repository and the corresponding metadata were stored on a RAID that was located in our internal network. The PostgreSQL database containing the metadata was hosted on our pServer. To enable communication between the OMERO server and the database back-end, the second firewall has been configured to unlock the TCP port 5432.

This enabled OMERO clients to connect the OMERO server from outside as well as from inside of the campus network. Concomitantly, our data were protected against external attackers. To preserve a maximum level of security, only a minimal amount of services necessary for OMERO and the backup software (see below) was running on our hardware.

Protection against unwanted data loss

To enable a user-friendly and rapid workflow, data protection systems have to offer very responsive data read and write operations. This performance was mainly influenced by two hardware components: the network component and the hard discs of the server. In addition, data protection systems also have to offer data security in both, data transfer and data storage. The key word for this is redundancy. Literally, redundancy means “plentiful”, which again means “more than once”. In both cases, more than one component of each type of hardware was present in our system.

The eServer as well as the pServer were equipped with two network adapters. Both servers were configured to automatically balance their data traffic optimally utilising the two adapters concurrently. In this case, redundancy allowed to optimise performance. At the same time, both servers were configured to handle hardware damage of one of the two adapters. Here, the operational adapter would replace the malfunctional adapter. In this case, redundancy preserved reliability.

The principle of redundancy was also designed into the second hardware component, the hard disc. Both servers were equipped with RAID systems. The eServer contained a RAID 1 consisting of two hard discs with an additional hot spare hard drive. In RAID 1 systems, one hard disc is mirrored onto a second hard disc, offering a maximum level of redundancy. The hot spare drive was used in case one of the two hard discs of the RAID 1 failed. Then, the data of the functional disc is automatically mirrored onto the hot spare disc, allowing the broken disc to be replaced during full operation of the RAID system.

The hard disc array of the pServer and the disc shelves described in Section 2.4.2 were configured as a RAID 5 system which combined both, redundancy of data and performance. Since this system was used to store our image data, a RAID level 5 was a good compromise between performance and redundancy.

For long-term archiving, image data that have not been used for a defined number of days were migrated to tape. Additionally, user data from the desktop PC's were stored on tape by using Bacula® (see Section 2.7.2).

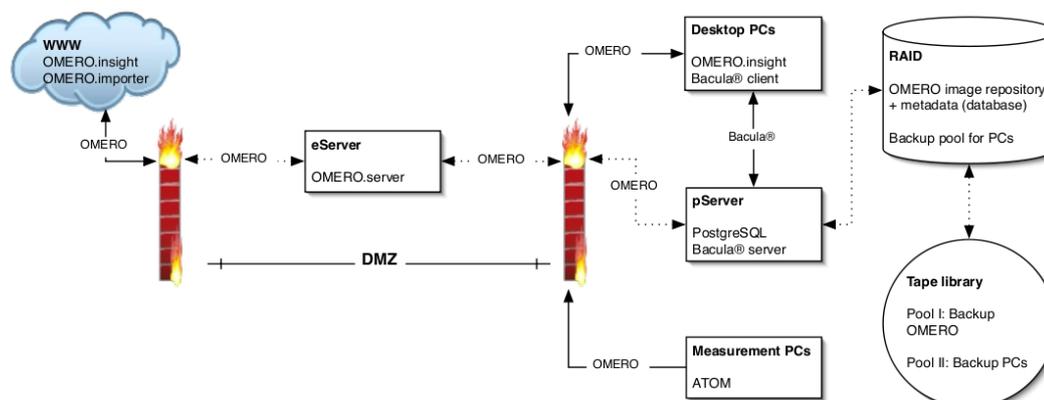


Figure 3.22: Schematic drawing of our network architecture. Arrows denote the direction of information flow. Dotted lines indicate redundant connections.

3.5 Data Communication and Coordination

3.5.1 A Multi-Client Communication Framework: Messaging Server

Application scenario

Our HCS system was composed of multiple modules such as a device for electrical and chemical / hormonal stimulation of cardiomyocytes, a digital microscope, algorithms for automated image analysis and a system for automated data storage and management. Since further devices such as a pipetting robot are likely to be included, the system had to be designed with expandability and scalability in mind.

For me, a device is a piece of hardware that is controlled by software. Devices originate from different vendors and their corresponding software is implemented using different programming languages. Nonetheless, in order to ensure an unobstructed interaction of multiple devices within an automated HCS system, a well coordinated communication between the different kinds of software was mandatory.

Varying biological questions require different measurement methods and thus different devices. Usually, devices that are not needed to perform a certain mea-

surement, are switched off. Thus, the overall system configuration changes with the biological question.

While some devices can be controlled using a single computer, it is recommended to run devices which are more resource intense on dedicated systems. Both scenarios imply different kinds of communication protocols, in order to enable an efficient workflow.

To meet such requirements, I have developed and implemented a communication concept, which could be used to coordinate communication between any number of devices.

General concept

The concept comprised a dedicated messaging server (acceptor, [109]) with the task to route messages between clients (connectors, [110]). As depicted in Figure 3.23, this server should provide different communication types (e.g. via sockets, pipes or shared memory segments) to enable efficient communication between both, clients running on different computers as well as clients running on a single computer. In such a system, each client should be able to become the messaging server, if necessary. This could be achieved by attaching a platform independent DLL containing the server functionality to each client. Once a server is available, clients could send and receive messages. This is shown in the flowchart in Figure 3.24. Thereby, a directed routing of messages should be possible. This involved the problem of how to address specific clients. To test such scenarios, the Messaging Server prototype had been developed.

Implementation

The Messaging Server software has been implemented in C++ (see Section 2.5.3). Once a client was started, it tried to connect to the server by making use of the server's IP address and port number. For this, default values were stored in a local property file that was identical for each client. If no server was available, the first client automatically started the server routines and thus became the messaging server. If the IP address of the server was different from the value stored in the

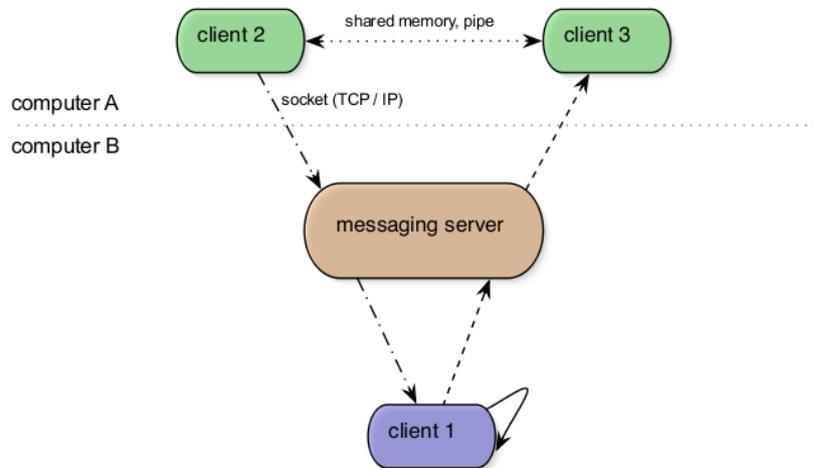


Figure 3.23: Scheme of the messaging framework. Clients might be located on the same computer or on different computers. If located on the same computer, clients could communicate using shared memory or pipe communication. Clients on different computers could communicate using TCP socket communication.

local property file, it had to be passed to each client manually. In a later version, this step might be automated by making use of Apple's service discovery protocol Bonjour [8].

To enable directed routing of messages, each client had a unique identification number, which was used to address the client. This number was composed of an arbitrary client name and the client's IP address. The latter was used to determine the communication type: clients running on the same computer communicated with each other using shared memory or pipes as communication layer. Clients running on different computers communicated via TCP sockets. Further communication types can be included by adding new sub-classes to the communication class (see next section).

In case the server was shut down, clients were disconnected automatically.

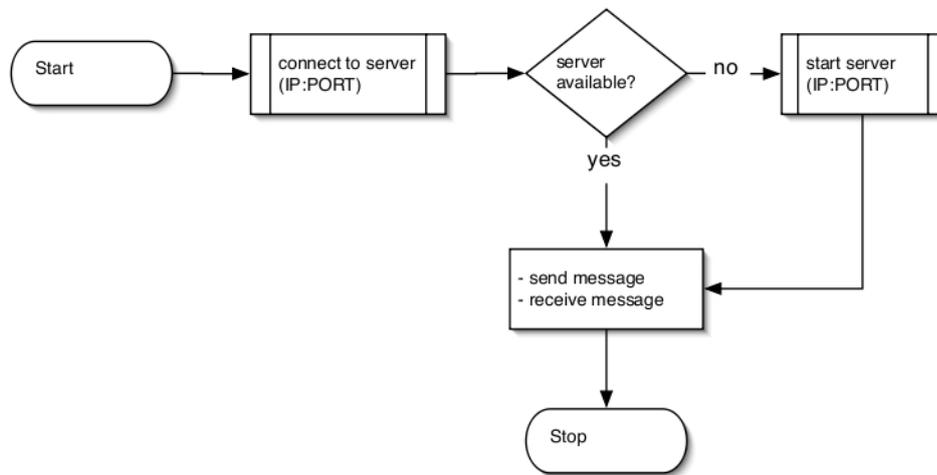


Figure 3.24: Flowchart of the Messaging Server software. If a client tried to connect to a non-existing messaging server, the client himself could overtake the role of the messaging server. Once a client had successfully connected to the messaging server, it was ready for sending and receiving messages.

Classes and members

Communication

This class contained wrapper functions for sending and receiving messages via the defined connection types. Each connection type was a sub-class of the Communication class (see Figure 3.25). Thus, the communication base class could be easily enhanced by adding new sub-classes, which specify further communication types.

Socket communication

If the communication type socket was selected, clients located on different computers communicated with each other by using a TCP socket. Therefore, the IP address and port number of the target client had to be applied by the sending client.

Shared memory

Clients located on the same computer exchanged information by using the shared memory communication type. The sender wrote information into memory seg-

ments and shared these segments with the receiver-client, which had then access to the information. Shared memory communication was the fastest communication type amongst clients.

Pipes

A second way for clients to communicate with each other, if located on the same computer, were pipes. The messaging server provided two types of communication via pipes: standard pipes and named pipes. Standard pipes are used to feed an output stream of a process directly into the input of the next process. Once the process has finished, the pipe disappears. Named pipes are an extension of the standard pipe concept. In contrast to standard (unnamed) pipes, named pipes do not disappear beyond the life of a process. Hence, named pipes have to be deleted once they are not longer used. A named pipe can be addressed by its name and is thus similar to a socket.

Helpers

The Helpers class contained useful auxiliary functions besides the messaging server's networking and server functionality.

Server

The Server class contained the server functionality. A while loop kept the server alive, allowing it to passively listen for incoming connections. The server was using a client map which contained identification numbers and IP addresses of registered clients. Once a client disconnected from the server, the corresponding entry in the client map was deleted. Thus, the server always knew how many and which clients currently were connected.

External libraries

To run the Messaging Server software in a multi-threaded manner, the C++ library Boost (see Section 2.5.4) has been used.

An overview of the class hierarchy of the Messaging Server software is given in Figure 3.25.

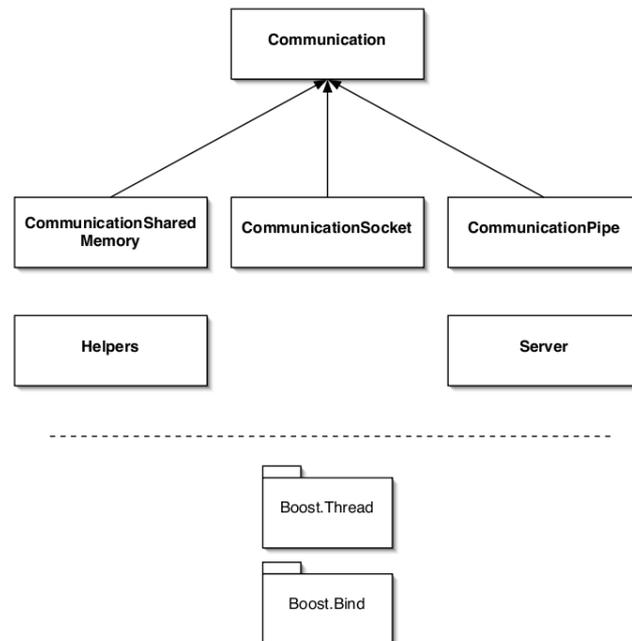


Figure 3.25: C++ class hierarchy of the Messaging Server software. The different communication types were derived from the **Communication** base class. Further communication types could be added by adding new sub-classes.

Conclusion

The Messaging Server software was a prototype, which demonstrated the necessity of a flexible and extendible communication framework within an HCS environment. Its purpose was to establish and to coordinate communication between different components of the HCS system. As a prototype, the messaging server has successfully convinced our software co-operation partner. The final version of the communication framework is called arivis Grid (see Section 2.8.1). It has been implemented by the arivis - Multiple Image Tools GmbH and is based on the messaging server prototype.

3.5.2 Use Case: Example Screening Run

This section describes an exemplified screening run using the following modules:

- *TILL Live Acquisition software* (see Section 2.8.4): controlled the digital microscope more™ (see Section 2.3.1) and served as starting point hosting the measurement protocol
- *arivis Browser* (see Section 2.8.2): used for image analysis and served as interface to the arivis Core Mathworks MATLAB™ interface (see Section 2.8.3)
- *arivis Core Mathworks MATLAB™ interface*: used as interface between the arivis Browser and the automated image analysis package CellAnalyser (see Section 3.3.1)
- *arivis Grid* (see Section 2.8.1): used to coordinate the communication between different grid clients
- *OMERO* (see Section 2.7.1): used for data storage, archiving and management
- *ATOM* (see Section 3.4.1): used for automated import of image data into OMERO
- *Hard- and software for electrical field stimulation* (see Section 3.2): used for electrical stimulation of cardiomyocytes
- *Hard- and software for chemical / hormonal stimulation* (see Section 3.2): used for chemical / hormonal stimulation of cardiomyocytes
- *Automated image analysis package CellAnalyser*: used for online and post-screening analysis of cardiomyocytes

Experimental setup

The goal of this use case was to demonstrate sequential scanning of each well of the 24-well plate for contracting cardiomyocytes. If suitable cells were found, an

image series was recorded, while an acute pulse protocol was assigned and applied to the well of interest. At the same time, the corresponding well was perfused. Once the image series had been completed, the data were automatically imported into OMERO and the screening run continued with the next well.

Workflow

Once the cardiomyocytes had been plated in each well of the HCS stimulation plate, each compartment had to be connected to the corresponding pinch valve of the perfusion hardware with a tube. The entire plate was then put into the microtitre stage of the more™. After that, the experimental setup was defined using the graphical protocol editor of the TILL Live Acquisition software. It is summarised by the following pseudo code, where $W = \{w \in \mathbb{N} \mid 0 \leq w \leq 23\}$:

Algorithm 4 EXEMPLIFIED EXPERIMENTAL SETUP

```

1: startPulseProtocol( $\forall w \in W, chronic$ ) {Pulsing Device}
2: for each  $w \in W$  do
3:   moveXYZ( $w$ ) {TILL Live Acquisition}
4:    $C \leftarrow FindContractingCells(w)$  {CellAnalyser}
5:   if  $|C| > 0$  then
6:     startPerfusion( $w$ ) {Perfusion System}
7:     changePulseProtocol( $w, acute$ ) {Pulsing Device}
8:     startImageAcquisition( $w$ ) {TILL Live Acquisition}
9:     stopImageAcquisition( $w$ ) {TILL Live Acquisition}
10:    closeImageStack( $w$ ) {arivis Browser}
11:    changePulseProtocol( $w, chronic$ ) {Pulsing Device}
12:    stopPerfusion( $w$ ) {Perfusion System}
13:  else
14:    continue
15:  end if
16: end for

```

The ATOM software was not directly invoked by the measurement protocol. Since it was autonomously monitoring the folder where the image data were stored, it automatically triggered the import process as soon as an image stack file had been

closed. Images could then be accessed within OMERO for post-screening analysis. The information flow between the different modules developed in this thesis is shown in Figure 2.11.

3.5.3 Practical Application: Screening Inside a Single Well

The use case described in the previous section illustrates the putative interaction of the HCS components during a fully automated screen. Since the TILL Live Acquisition software could not yet be connected to the arivis Grid (see Figure 2.11), such a fully automated workflow was not possible. Therefore, electrical and chemical stimulation during the following experiment were controlled manually. The TILL Live Acquisition software was used to control the more™.

Experimental protocol

Cardiomyocytes from mice with a GCaMP2⁵ knock-in were used for single-channel fluorescence recordings. GCaMP2 has an absorption maximum at 487 nm and an emission maximum at 508 nm [119]. Calcium signals from individual cells were recorded while screening one well. Therefore, different positions inside the well were accessed randomly. The experimental protocol⁶ is shown in Figure 3.26 (details are explained in the following subsections): cells were electrically stimulated by repetitively applying a pulse protocol. After a control period, cells were transiently perfused with isoproterenol (also referred to as isoprenaline), a β -adrenaline agonist, that causes an increase in calcium transient amplitude and contractile force. Perfusion as well as electrical stimulation was performed globally on the entire well. I therefore set up a screening run that employed switching to 7 different recording positions in a repetitive sequence throughout the entire experiment time. Images were stored on our OMERO server using ATOM (see Section 3.4.1) and analysed using CellAnalyser (see Section 3.3.1).

⁵GCaMP2 is a genetically engineered calcium indicator, that changes its fluorescence upon calcium binding.

⁶The experimental protocol was designed and executed by Qinghai Tian.

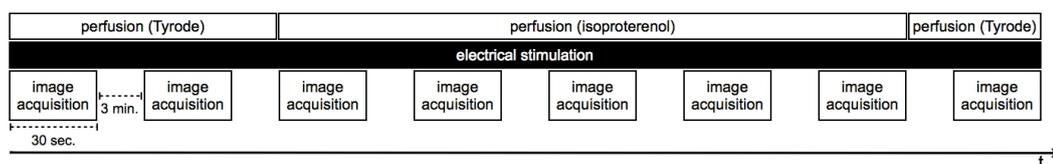


Figure 3.26: Experimental protocol of the GCaMP2 experiment. Each image acquisition period comprised 750 images, which were combined into one TIFF file with 6000 frames in total. The protocol was applied to 7 different positions inside the well.

Electrical and chemical stimulation

Cells were electrically stimulated applying alternating pulse trains (voltage: ± 20 V; pulse width: 5 ms; frequency: 0.5 Hz) to the entire well. For chemical stimulation, isoproterenol (100 nM) was used. The flow rate of the perfusion was set to 1.25 ml/min. Figure 3.27 shows the 24-well plate and the lid placed on the more™. The lid was connected to the perfusion system, which can be seen in the background.



Figure 3.27: Hardware for electrical and chemical stimulation. Two carbon electrodes per well enabled electrical stimulation of the cells. Openings inside the lid were used to connect the perfusion via two thin silicone tubes. The enlightened LED in the mounting plate of the perfusion system indicated perfusion was active for the well of interest.

Data storage and archiving

The 7 TIFF stacks (dimensions (each): $172 \times 128 \times 6000$) were automatically imported into OMERO using ATOM. Each TIFF file contained 8 recordings of the same well position (see Figure 3.26). Figure 3.28 shows the GUI of the OMERO.insight client, which was used to manage (e.g. organise, annotate) the acquired images.

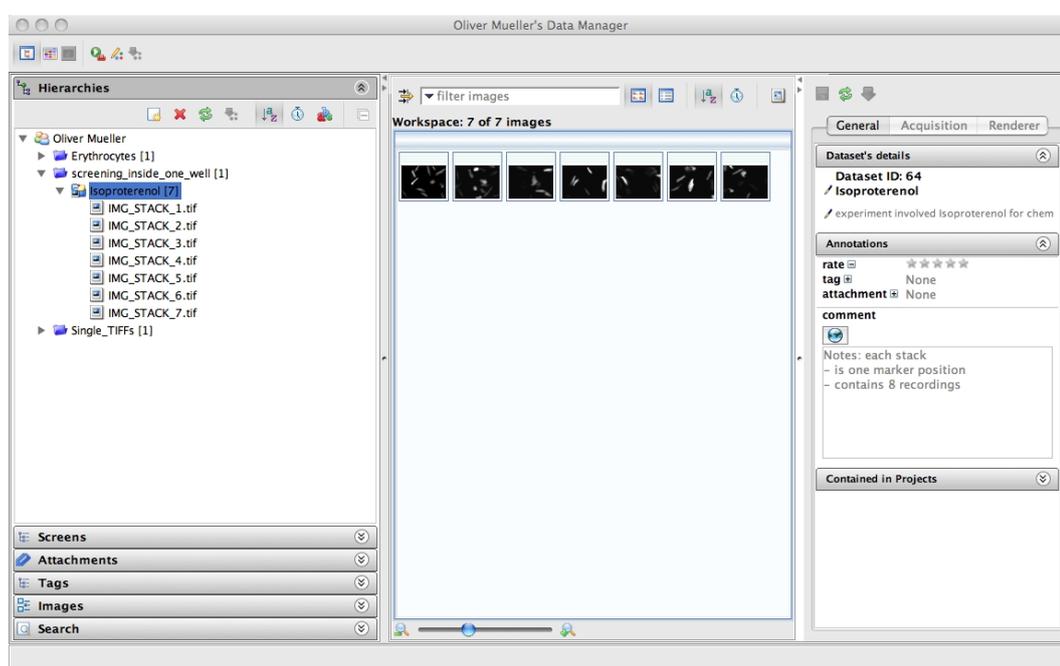


Figure 3.28: Data manager of the OMERO.insight client: the 7 image stacks were stored on the OMERO server.

Image analysis and result

Using CellAnalyser in semi-automatic mode, cells were automatically detected and displayed during post-screen analysis. As an example, Figure 3.29 shows the GUI containing labelled ROI's of one TIFF stack.



Figure 3.29: CellAnalyser - GUI with labelled ROI's.

Figure 3.30 shows the corresponding table of intensity values for each ROI, which are displayed graphically in Figure 3.31. During the application of isoproterenol the expected increase in transient amplitude was observed.

The screenshot shows an Excel spreadsheet with a table of intensity values. The formula bar at the top displays the value 2.1681e+05. The table contains 25 rows of data, each with 5 columns labeled A through E. The values are presented in scientific notation.

	A	B	C	D	E
1	2.1681e+05	4.3064e+05	1.1849e+06	5.7529e+05	5.0721e+05
2	2.1771e+05	4.3151e+05	1.1758e+06	5.6453e+05	5.1776e+05
3	2.18E+008	4.3139e+05	1.1759e+06	5.6718e+05	5.1648e+05
4	2.1666e+05	4.3002e+05	1.1725e+06	5.6644e+05	5.1503e+05
5	2.1647e+05	4.29E+008	1.1705e+06	5.6297e+05	5.1405e+05
6	2.1722e+05	4.2882e+05	1.1715e+06	5.6410e+05	5.1375e+05
7	2.16E+008	4.28E+008	1.1681e+06	5.61E+008	5.12e+05
8	2.1488e+05	4.2689e+05	1.1646e+06	5.6029e+05	5.1147e+05
9	2.1595e+05	4.2584e+05	1.1635e+06	5.5883e+05	5.1084e+05
10	2.1497e+05	4.2653e+05	1.1629e+06	5.60E+008	5.0921e+05
11	2.1409e+05	4.24E+008	1.1593e+06	5.5676e+05	5.0884e+05
12	2.1356e+05	4.2299e+05	1.1574e+06	5.5581e+05	5.0782e+05
13	2.1351e+05	4.2451e+05	1.1577e+06	5.5665e+05	5.0753e+05
14	2.1295e+05	4.2264e+05	1.1545e+06	5.54E+008	5.0567e+05
15	2.1225e+05	4.2087e+05	1.1514e+06	5.5242e+05	5.0467e+05
16	2.1194e+05	4.2292e+05	1.1524e+06	5.5402e+05	5.0465e+05
17	2.1213e+05	4.2341e+05	1.15E+009	5.5373e+05	5.0727e+05
18	2.1141e+05	4.2291e+05	1.1515e+06	5.5257e+05	5.0962e+05
19	2.12E+008	4.22E+008	1.1486e+06	5.5062e+05	5.0801e+05
20	2.1092e+05	4.22E+008	1.1448e+06	5.4903e+05	5.0607e+05
21	2.1123e+05	4.2158e+05	1.1459e+06	5.4877e+05	5.0454e+05
22	2.1017e+05	4.1964e+05	1.1443e+06	5.4746e+05	5.02e+05
23	2.0946e+05	4.1833e+05	1.1395e+06	5.4625e+05	4.9927e+05
24	2.0915e+05	4.1814e+05	1.1385e+06	5.45E+008	5.00E+00E
25	2.0876e+05	4.1694e+05	1.1393e+06	5.4458e+05	4.9774e+05

Figure 3.30: Table of intensity values for each ROI.

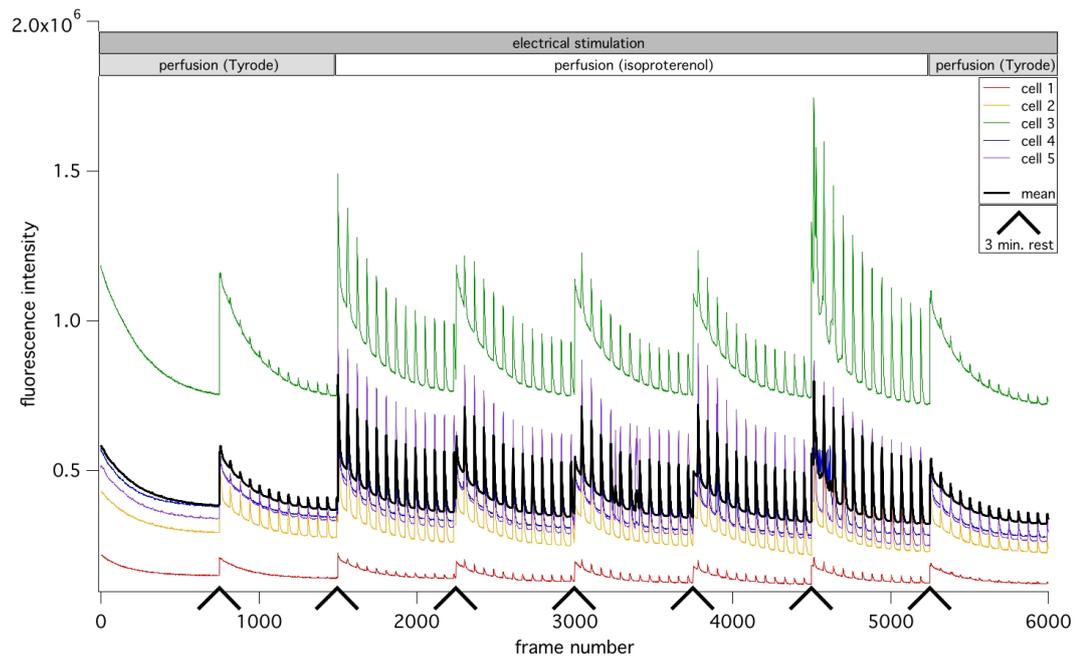


Figure 3.31: Fluorescence transients for each ROI. Perfusion with isoproterenol led to an increased transient amplitude. The black trace indicates the mean of 5 cells, the coloured traces represent fluorescence transients of each individual cell. The repetitive loss of fluorescence is due to a photoconversion phenomenon [30, 131] of GCaMP2.

3.5.4 General Outline of the HCS Project

This chapter concludes with a general outline comprising the components developed in this dissertation, which is shown in Figure 3.32.

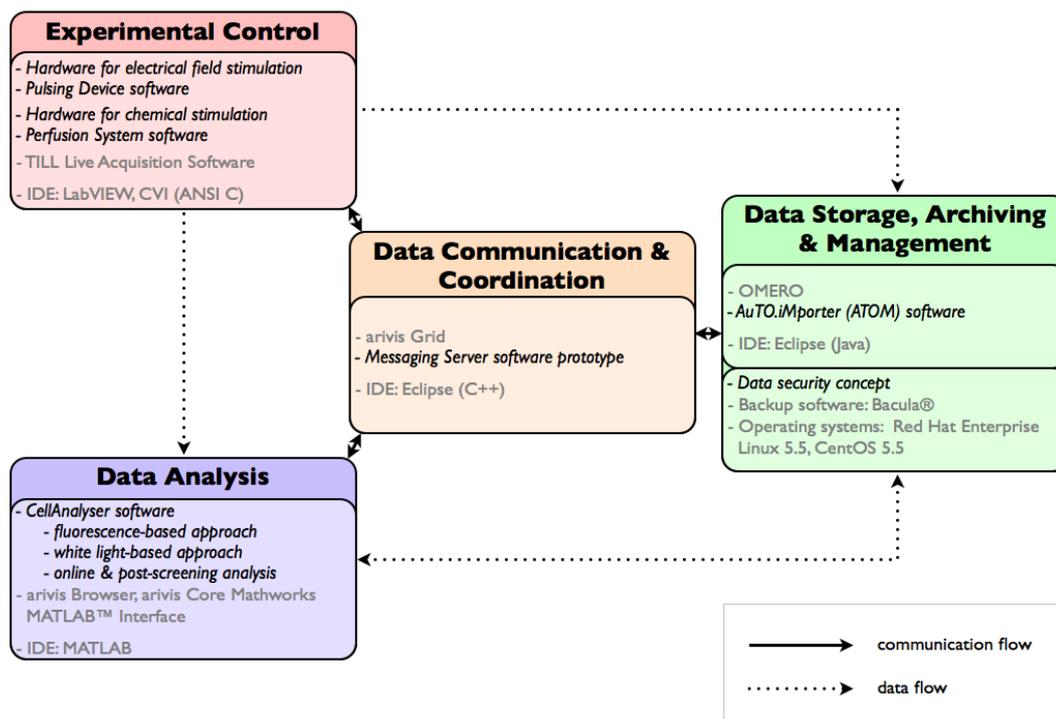


Figure 3.32: Project overview: modules were classified according to their field of functions. Components denoted in black/italic have been developed and implemented by myself. Components denoted in grey stem from other manufacturers and have been used within the project. Arrows denote the direction of information flow.

Chapter 4

Discussion

The results of this dissertation cover a wide variety of subject areas such as the design of hardware enabling cell stimulation (24-well plate / lid combination, perfusion, SSRD, patch bay) and the development of software for experimental control (Pulsing Device, Perfusion System), data analysis (CellAnalyser), storage (ATOM) and communication (Messaging Server). In the following, the most important results are discussed.

4.1 Hardware

4.1.1 Electrical Stimulation Involving Individual Wells

To my knowledge, the application of arbitrary pulse protocols to individual wells as described in Section 3.2.6 provides a flexibility which is unique and commercially not available up to now. Although the functionality of the Pulsing Device is comparable to the C-Pace EP Culture Pacer [54] from IonOptix, there are important functional differences. The latter was developed for chronic stimulation of cardiomyocytes only during their time in culture. Its eight output channels allow the application of electrical stimuli up to ± 40 V. Since the maximum current is limited to 240 mA, the C-Pace EP cell culture stimulator allows to pace one well per channel [54]. According to Equation 3.1, this maximum current is slightly below

the requirements for pacing one well of our well plate. Using this device together with our hardware for electrical field stimulation (24-well plate / lid) would allow concurrent pacing of 8 wells in total and may lead to unwanted voltage drops in the worst case. In contrast, the device developed in this thesis allows to stimulate each of the 24 wells with its own stimulation regime and thus offers incomparable more flexibility.

The power supply of the Pulsing Device providing the negative and positive voltages may be a potential starting-point for future work: although control of pulse train parameters such as the duration, frequency, pulse width and the number of execution steps is possible for each individual well, the voltage can only be set globally, i.e. for the entire well plate. To provide an independent control of the voltage for all 24 wells individually, either 24 dual power supplies or 24 independent voltage controllers utilising the same dual power supply would have to be installed. While the first solution is too cost-intensive, the second solution would require the design of an analogue controlled electrical circuit for each well.

A scale-up from 24 to 96 or even more wells is possible. In addition to changes in the well plate / lid design (e.g. electrodes, circuit paths), this would also involve changes on the access of the wells. Considering such a high number of compartments, an independent control of each individual well would rather complicate the handling of the well plate as well as involve major changes on both, the hardware and the software side. As an alternative, combining single wells to groups of wells could be reasonable. For instance, a 96-well plate could be controlled utilising the same I/O-hardware by combining 4 wells and connecting them to one output channel of the patch bay described in Section 3.2.4. This would involve only slight changes on the GUI of the control software. Since such a scale-up involves changes in the well size and thus changes in the design of the electrodes, the maximum current has to be recomputed according to Equation 3.1.

4.1.2 Elastic Coating and Oleophobic Foil

As described in Section 2.1, the bottom of the well plate combines two surface properties: (i) the coating of the top of the foil is elastic and mimics the local environment of cardiomyocytes (such as the extracellular matrix, the local elastic network which physiologically occurs between cardiomyocytes); (ii) the bottom surface of the foil is coated with an oleophobic layer, which prevents immersion oil from sticking. While the first property results in significantly increased contractions of cardiomyocytes (with identical calcium transients) when compared to non-elastic substrates, the second property enables the use of oil immersion objectives in optical screens [80].

Since the use of immersion oil is mandatory for using high NA objectives, a stable oil drop between the foil and the objective tip was an essential requirement. Although one could imagine mechanisms for continuous oil supply, this would have increased the complexity of the system significantly. In [80] I have tested two immersion oils using standard well plates. I could observe, that both immersion oils reacted with the material of the bottom of the well plate. As a result, the surface was opaque. Due to these problems and its consequences, commercially available HCS systems such as the BD Pathway™ high-content bioimager (BD Biosciences, San Jose, USA) cannot use oil immersion objectives when using plastic plates [11]. Most likely for those reasons, e.g. the Opera High Content Screening System (Perkin Elmer, Waltham, USA) only uses high NA water immersion objectives [94].

Therefore, the oleophobic foil was beneficial, if not necessary, for high-resolution optical screens. Besides avoiding a break of the oil drop, the surface of the foil does not cloud due to the oil. Potentially, the oil repellent property of the foil could be increased in order to enable even higher speeds concerning the X/Y-stage movement.

4.2 Software

4.2.1 Image Analysis Software for Automated Cell Detection and Classification

Several software tools for automated cell detection and classification are currently available. Although implemented for the same purpose, each software has its individual advantages and disadvantages. A selection of scientific image processing and analysis software will be discussed in the following subsections.

ImageJ

The first version of the image processing software ImageJ [5, 101] has been released in 1997. Since then, it has become one of the most popular tools used in microscope [65, 97] and medical [64] image processing. Its Java-based open-source architecture allows to extend and adapt the software according to the individual needs. Functionality can be added by using predefined or self-developed plug-ins. Automation of tasks can be implemented by defining macros. Since ImageJ can make use of the Bio-Formats library (see Section 2.7.1), it can read various file formats. Especially its fast Java routines for importing large image stacks make ImageJ a valuable tool for analysing large time series.

CellProfiler™ and CellProfiler Analyst

After ImageJ, CellProfiler™ was the first free open-source software for high-throughput cell image analysis [24]. Developed as an easy-to-use tool that can be used by scientists without any programming skills, it can be applied to a wide range of biological images that are based on agar plates, cell microarrays, microscope slides or multi-well plates, amongst others [71, 75]. CellProfiler™ has also been used in image-based HTS and HCS experiments [75]. Based on a modular design, it provides over 50 modules for performing image analysis on the cellular and subcellular level. Besides automated image correction and cell segmentation,

CellProfiler™ can be used to perform various per-cell measurements such as determining the cell size or the cellular content of a specific protein.

Implemented in MATLAB, CellProfiler™ follows a pipeline concept, which allows to connect different modules. Information then flows from one module to the next module making use of the pipeline, leading to a sequential arrangement of the different analysis steps [24]. Apart from the MATLAB implementation, which uses the MATLAB Image Processing Toolbox, a compiled C++ implementation is available. The latter version is faster and can thus be used for time-critical application scenarios. To further increase computation speed, CellProfiler™ can distribute the workload using clusters of computers, if available. Furthermore, CellProfiler™ is integrated with the OME project and can thus be used together with OMERO [88].

Although CellProfiler™ provides a variety of strengths, the software has several drawbacks, when considering our project: (i) since its main focus is on 2D images, CellProfiler™ has very limited support for 3D image stacks, as they occur during a time-lapse experiment [24]; (ii) depending on the number of parameters, analysis of a single image can take from 20 seconds up to five minutes [24]. The average computation time per image is two minutes [24]; (iii) because of the sequential arrangement of the different modules, different image analysis steps such as illumination normalisation and cell segmentation cannot be parallelised. Due to this serialisation, the analysis of individual images during an HCS experiment represents a bottleneck [75]; (iv) although the algorithm for image segmentation is quite robust and widely used, it cannot be applied directly to segment cardiomyocytes or erythrocytes. This is due to the fact that the segmentation starts with segmenting the nuclei [24, 75], which requires a proper staining of the nuclei. Erythrocytes do not have a nucleus while ventricular myocytes usually contain two nuclei (see Section 1.1.2), which would result in two or more segmented objects per cell. Furthermore, cytotoxic effects might occur due to the nuclear staining. Since all further steps for automated cell segmentation depend on this first step, this approach is not compatible with our needs.

Because the MATLAB code of CellProfiler™ is open-source, suitable modules can be either included into the CellAnalyser framework or the other way around: tools of the CellAnalyser software can be adapted and converted into modules that can be used within the CellProfiler™ pipeline concept.

To interactively explore large, multi-dimensional image-derived data and thus to further analyse cells that have been detected by CellProfiler™, the open-source Java-based software CellProfiler Analyst [60] can be used. The software provides machine learning-based methods for phenotype scoring and can be trained to robustly recognise complex morphological phenotypes for automated cell classification [59]. Both, CellProfiler™ and CellProfiler Analyst are designed for post-screening analysis and depend on the user's feedback during analysis. Nonetheless, their source code can be used to extract functionality valuable for our HCS approach and to make it available for online routines of the CellAnalyser software.

DetecTIFF©

The image analysis software DetecTIFF© has been published in 2009 [42]. It allows to perform object recognition and quantification from digital images in a fully automated manner by applying dynamic intensity thresholding and size-dependent particle filtering. DetecTIFF© is implemented in LabVIEW and uses the Vision Development Module (National Instruments) for image processing and analysis routines. It can be used for automated HCS.

Although the software can communicate with other software implemented in LabVIEW [42], the lack of interfaces to other programming languages than LabVIEW makes it unfeasible for our HCS environment, which utilises MATLAB as common interface language for data analysis.

Advanced Cell Classifier

Similar to the CellProfiler Analyst, the Advanced Cell Classifier (ACC) [36] applies machine learning-based methods in order to analyse data from cell-based high-content screens with minimal user interaction. It is also implemented in MATLAB and apart from MATLAB's Image Processing Toolbox, ACC utilises the Neural

Networks Toolbox (The MathWorks). Unfortunately, only a pre-compiled version of the software can be downloaded.

Summary

Although MATLAB allows the usage of Java code, ImageJ's fast routines for importing image stacks can only be used in a very limited way. The maximum memory (heap) sizes that can be used for Java routines in MATLAB are 128 MB (32-bit) and 196 MB (64-bit), respectively [123]. Hence, ImageJ's routines can only be used to import small image stacks into MATLAB. In this case, the performance of these routines is even outperformed by MATLAB's native functions for importing images. Therefore, a migration of ImageJ's import routines into MATLAB does not make sense in our case.

Since the MATLAB code of CellProfiler™ is open-source, suitable modules can be migrated and integrated into the CellAnalyser framework (see Section 3.3.1).

Altogether, none of the above mentioned packages for image processing and image analysis offers all tools and algorithms that were needed for our HCS approach. Therefore, CellAnalyser is an attempt to combine the best strategies of available image processing software and to incorporate them into our HCS environment. Its modular design enables a seamless incorporation of arbitrary image analysis algorithms such as specific algorithms for evaluating calcium signals from simple $\frac{\Delta F}{F_0}$ computations or the more complicated detection of calcium sparks [25].

4.2.2 Potential Improvements of the CellAnalyser Framework

Image import into MATLAB

The most time consuming part of the fluorescence-based image analysis described in Section 3.3.2 is the import of an image stack. Reading a TIFF-stack comprising 9000 planes from hard disc takes up to 650 seconds even on a 64-bit PC with 8 GB of RAM using MATLAB 2008a. This is due to the fact that MATLAB first extracts various information about the image stack such as the total number of

planes. Therefore, each individual image plane is first counted before the actual loading of the planes starts.

To shorten this inefficient process, I developed a strategy which omits counting of the image planes and directly starts with reading the planes from hard disc instead. Unfortunately, the loop, which is used to read the images has to know the total number of image planes prior to the actual import process. Therefore, I have defined a value n for this number, which is used to initialise an image stack with n planes containing the value 0. During the import process, the 0 values are overwritten by the current image's content. If the real number of total image planes is lower than n , the iteration process stops and the remaining planes are removed. This strategy allows to shorten the import process to 325 seconds under the same conditions as mentioned above. The only drawback is, that image stacks with more than n image planes cannot be completely imported without manually adapting n .

Fortunately, MATLAB 2009a has been improved with respect to the import of large image stacks. Using MATLAB 2009a on a typical 32-bit PC with 2 GB of RAM, the same TIFF-stack can be imported in 131 seconds.

Similar to the CellProfiler™, the CellAnalyser software executes the different sections of an experiment script sequentially, starting with the image import procedure. Detection and analysis steps as well as the final result are thus delayed until the import process has been completed. In order to overcome this problem, the import process could be parallelised with respect to the following steps. Since the computation of the ratio according to Equation 3.2 is only based on one image per channel, the use of two images per computation step instead of the whole matrix of intensity values would tremendously decrease memory usage. Furthermore, the current result could already be displayed and updated during computation, giving the user the opportunity to immediately interact with the analysis process.

Automated cell detection

When cells are clearly separate or in loser proximity from one another, the CellAnalyser software can automatically detect ROI's. But especially on crowded cell

samples, the algorithm cannot properly separate clusters of connected or overlapping cells. Unfortunately, clusters of cells can occur during an experiment. Cell-based image analysis, especially in combination with screening applications, is still a great challenge [33, 35]. Since image segmentation in general is still the most challenging step in image analysis [24], further efforts have to be made to address this critical point. Apart from the implementation of machine learning-based methods as mentioned above, utilisation of heuristic optimisation approaches such as simulated annealing [104] may be valuable.

4.2.3 ATOM and OMERO.dropbox

The beta-4 release of OMERO provides the file system monitor OMERO.dropbox, which pursues the same goal as ATOM: the automated import of image data into OMERO. It is based on OMERO.fs, a system of file services, which has the capability of notifying OMERO.dropbox in case files have been added, modified or deleted. In contrast to ATOM, OMERO.fs and OMERO.dropbox run on the same computer as the OMERO.server application. From here, OMERO.dropbox monitors the dropbox directory, which has to be a subdirectory of the OMERO binary repository directory. On the server, subdirectories consisting of the OMERO user name can be created for each user. Once the user copies data into his user directory, the import process is triggered automatically.

Although OMERO.dropbox is a fully integrated OMERO background client, the current development stage has the following drawbacks: (i) since OMERO.fs *“is only available for specific versions of certain operating systems”* [91], the application of OMERO.dropbox is restricted to operating systems supported by OMERO.fs; (ii) monitoring a network-attached share is *“strictly not supported”* [91]. Therefore, users have to make their dropbox folder accessible to the acquisition system over the network. Most imaging systems are based on Microsoft Windows while most server systems are based on Unix / Linux. Sharing files in such a heterogeneous network environment requires the usage of a network file system mount like Samba [105], which is not always feasible. Problems such as connection time-outs or low information flow-rates are not unlikely to occur when

using Samba for transferring large files from a Windows PC to a Unix / Linux server over the network. Otherwise, image data would have to be copied into the dropbox folder manually, using e.g. the secure copy (SCP) protocol. In this case, the automation step would be by-passed; (iii) by simply copying data into the dropbox directory, the flexibility of spontaneously assigning project and dataset names gets lost; (iv) since OMERO.fs and OMERO.dropbox are still in their infancies, copying large number of files “*may result in files failing to import*” [91].

ATOM avoids the above mentioned drawbacks by automatically importing image data directly into OMERO, without the necessity of mounting an external file system. It is running on the PC attached to the acquisition system and does not depend on the capabilities of an external system of file services.

As a future perspective, ATOM could be enhanced by a copy mechanism which allows to automatically copy files from an acquisition system into a dropbox directory without the need of a Samba mount. This would allow the triggering of OMERO.dropbox by ATOM and thus enable acquisition software to automatically import image data e.g. via SCP. As a second outlook, ATOM could be enhanced by a cleaning mechanism, which automatically deletes images that have already been imported into OMERO from the acquisition system.

4.3 Overall System

Today, commercial HCS systems are used to screen cell lines instead of primary cultured cells (see Section 1.3). For screening multi-well plates, either air [11] or water immersion objectives [94] are used (instead of high NA oil immersion objectives). Although hardware for chronic stimulation of electrically excitable cells exists [53, 54], handling of the stimulation regime is rather limited and inflexible. Considering chemical stimulation, the BD Pathway™ high-content bioimager [11] does not allow liquid exchange during a screening run. Instead, it allows only single addition of a substance. Our approach overcomes the above mentioned limitations of commercially available HCS systems by providing the following features:

- screening of both, cell lines and primary cultured cells
- application of air as well as high NA oil immersion objectives for screening multi-well plates
- flexible electrical stimulation of cells
- continuous perfusion of the well

Furthermore, our system is scalable (e.g. incorporation of well plates with a larger number of wells is possible) and expandable (e.g. integration and control of a pipetting robot via the arivis Grid).

4.4 Future Perspectives

4.4.1 Workflow Essentials

Feedback loop

Using the current version of CellAnalyser in its full automated mode, all objects that fulfill the selection criteria described in Section 3.3.2 (size, shape) are analysed. Nevertheless, to a varying degree dead cells as well as cells that do not respond to the electrical stimulus contributed to the population of objects recognised by my algorithm. In order to foster faster analysis times and reduce data that needs post-processing by the experimenter, it would be highly desirable that the software itself would be able to readily identify “good” cells. For the sake of simplicity let us only distinguish living from dead cells – but these criteria can of course be defined differently. The CellAnalyser software contains functions which allow to distinguish between these two states of cells by measuring changes in the length of the cell’s longitudinal axis. If there is no such change during an experiment, the cell can be ignored. This result has only to be fed into a kind of feedback loop during an online measurement. If sent to the TILL Live Acquisition software, for instance, a Boolean flag could be used to either enable or disable image acquisition at a particular well position, thus leading to a possible decrease of the final data volume as well as a reduction of the user-driven processing time.

TILL Live Acquisition software integration

In order to establish fully automated HCS protocols, both, the Pulsing Device software and the Perfusion System software have to be integrated into the graphical protocol editor of the TILL Live Acquisition software. This allows the access and control of the corresponding hardware from this central point of execution. Due to the fact that the TILL LA protocol editor cannot communicate with the ICU using conditional statements and since protocols can only be uploaded to the ICU in one go (not step-by-step), the integration of the above mentioned devices utilising the arivis Grid currently represents a problem for the software developers of TILL Photonics. In a measurement protocol, conditional statements are required to turn a specific device either on or off at a defined point of an iteration loop. Since the definition of such a point can vary from protocol to protocol, a flexible implementation is needed. Therefore, a discussion about possible solutions is currently in progress. Since TILL Photonics is planning the integration of a Python¹ interface, which can upload a measurement protocol to the ICU successively, handling of conditional statements would become possible. By using the Simplified Wrapper and Interface Generator SWIG [12], ANSI C functions of the API of the grid clients can be directly accessed from a Python script. Due to the modular design of the overall system, no modifications to the arivis Grid or the above mentioned devices are necessary. Hence, the Python interface could be a potential solution to foster the integration of our devices into the TILL LA protocol editor.

4.4.2 Add-ons

Artificial neural network

Due to their variety in shape, cardiomyocytes cannot be detected by applying a general binary mask as a filter. Similar to the CellProfiler Analyst or the Advanced Cell Classifier software described in Section 4.2.1, an adaptive algorithm which can be trained by analysing and classifying different sets of cells is desired. Machine

¹Python is a cross-platform programming language, that has been developed by the Dutch computer programmer Guido van Rossum in 1991.

learning algorithms applying artificial neural networks (ANN) [104] for pattern recognition appear as an almost ideal candidate to address such challenges. Similar to neurones in the brain, an ANN can be used for collection, processing and discrimination of signals [104]. Thereby, an ANN is tolerant towards noisy input (in this case, the term “noisy” mainly refers to the variety of different cell shapes and not solely to camera or illumination noise usually contained in digital images). As a future perspective, the CellAnalyser software could be enhanced by a module which is dedicated to this task. MATLAB’s Neural Networks Toolbox or parts of the CellProfiler Analyst source code might be a good starting point.

Arivis Grid connection

Currently, the Pulsing Device and the Perfusion System software can connect to the arivis Grid by using an ANSI C class for registering a grid actor as described in Section 2.8.1. The arivis Multiple Image Tools GmbH is working on a solution which allows to use LabVIEW source code instead of ANSI C source code to fulfill this task. Then, it will be possible to register a grid actor by simply placing the corresponding LabVIEW icon on the block diagram, similar to the example described in Section 2.5.1.

As can be seen in Figure 2.11, the TILL Live Acquisition software cannot yet be connected to the arivis Grid. In the future, each device of the HCS system is to be equipped with the capability of connecting to the arivis Grid. This will tremendously increase the flexibility of the entire system.

4.5 Conclusion and Outlook

In my doctoral thesis, I presented a proof of concept for automated image-based HCS of primary cultured adult cardiomyocytes. Together with earlier work on the cell culture of primary isolated adult cardiomyocytes (see [130]), my work will foster the construction of a fully automated HCS system using these cells. The key aspects of development concentrate on the flexibility, scalability and expandability of the HCS environment. In contrast to commercial HCS systems as described

in [120] and similar to the open-source paradigm known from computer software, our approach is open for custom-made add-ons that have been developed in-house. Furthermore, commercial products with an open API can be incorporated into the system. In the medium run, a pipetting robot could e.g. be integrated by connecting it to the arivis Grid. Further image analysis software supporting bi-directional MATLAB interfaces such as Imaris (Bitplane [19], Zurich, Switzerland) may also be taken into consideration.

Bibliography

- [1] Ansi/sbs 1-2004: Microplates - footprint dimensions
- [2] Ansi/sbs 2-2004: Microplates - height dimensions
- [3] Ansi/sbs 3-2004: Microplates - bottom outside flange dimensions
- [4] Ansi/sbs 4-2004: Microplates - well positions
- [5] Abramoff M, Magelhaes P, Ram S (2004) Image processing with imagej. Biophotonics International 11(7):36–42
- [6] Allied Vision Technologies GmbH. <http://www.alliedvisiontec.com/>
- [7] Apple Inc. <http://www.apple.com/>
- [8] Apple Inc (2010), Bonjour. <http://developer.apple.com/opensource/>
- [9] arivis Multiple Image Tools GmbH. <http://www.arivis.com/>
- [10] Bacula. <http://www.bacula.org/>
- [11] BD Biosciences (2010), Faqs. <http://www.bdbiosciences.com/instruments/pathway/faqs/index.jsp>
- [12] Beazley D, Simplified wrapper and interface generator (swig). <http://www.swig.org/>
- [13] Bell Laboratories. <http://www.alcatel-lucent.com/wps/portal/BellLabs/>

- [14] Bers DM (2002) Excitation-Contraction Coupling and Cardiac Contractile Force. Kluwer Academic Publishers
- [15] Bers DM (2008) Calcium cycling and signaling in cardiac myocytes. *Annu Rev Physiol* 70:23–49
- [16] Beta LAYOUT GmbH.
<http://www.pcb-pool.com/>
- [17] Bio-Chem Fluidics. <http://www.biochemfluidics.com/>
- [18] Bio-Formats. <http://www.loci.wisc.edu/software/bio-formats/>
- [19] Bitplane Scientific Software. <http://www.bitplane.com/>
- [20] Boost C++ Libraries. <http://www.boost.org/>
- [21] Cairns JA, Connolly SJ (1991) Nonrheumatic atrial fibrillation. risk of stroke and role of antithrombotic therapy. *Circulation* 84(2):469–481
- [22] Canny J (1986) A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8(6):679–698
- [23] Carl Zeiss AG, 100 Jahre fluoreszenzmikroskopie - 1908 entdeckung der mikro- lumineszenz. <http://www.zeiss.de/>
- [24] Carpenter AE, Jones TR, Lamprecht MR, Clarke C, Kang IH, Friman O, Guertin DA, Chang JH, Lindquist RA, Moffat J, Golland P, Sabatini DM (2006) Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol* 7(10):R100
- [25] Cheng H, Lederer WJ (2008) Calcium sparks. *Physiol Rev* 88(4):1491–1545
- [26] Clapham DE (2007) Calcium signaling. *Cell* 131(6):1047–1058
- [27] COSMO Electronics Corporation. <http://www.cosmo-ic.com/>

- [28] Cox G (2007) *Optical Imaging Techniques in Cell Biology*. CRC Press, Taylor & Francis Group
- [29] Datatranslation. <http://www.datatranslation.com/>
- [30] Dickson RM, Cubitt AB, Tsien RY, Moerner WE (1997) On/off blinking and switching behaviour of single molecules of green fluorescent protein. *Nature* 388(6640):355–358
- [31] Digital Imaging and Communications in Medicine. <http://medical.nema.org/>
- [32] Dove A (2003) Screening for content—the evolution of high throughput. *Nat Biotechnol* 21(8):859–864
- [33] Echeverri CJ, Perrimon N (2006) High-throughput rna screening in cultured cells: a user's guide. *Nat Rev Genet* 7(5):373–384
- [34] Eclipse. <http://www.eclipse.org/>
- [35] Eggert US, Mitchison TJ (2006) Small molecule screening by imaging. *Curr Opin Chem Biol* 10(3):232–237
- [36] Eidgenössische Technische Hochschule Zürich, Advanced cell classifier. <http://acc.ethz.ch/>
- [37] Extensible Markup Language. <http://www.w3.org/XML/>
- [38] Flegel KM (1995) From delirium cordis to atrial fibrillation: historical development of a disease concept. *Ann Intern Med* 122(11):867–873
- [39] Fuster V, Rydn LE, Cannom DS, et al (2006) *Acc/aha/esc 2006 guidelines for the management of patients with atrial fibrillation: A report of the american college of cardiology/american heart association task force on practice guidelines and the european society of cardiology committee for practice guidelines (writing committee to revise the 2001 guidelines for the management of patients with atrial fibrillation): Developed in collaboration with the*

- European Heart Rhythm Association and the Heart Rhythm Society. *Circulation* 114:e257–e354
- [40] Gaudreault M, Carrier P, Larouche K, Leclerc S, Giasson M, Germain L, Gurin SL (2003) Influence of sp1/sp3 expression on corneal epithelial cells proliferation and differentiation properties in reconstructed tissues. *Invest Ophthalmol Vis Sci* 44(4):1447–1457
- [41] GC-Prevue. <http://www.graphicode.com/>
- [42] Gilbert DF, Meinhof T, Pepperkok R, Runz H (2009) Detectiff: a novel image analysis routine for high-content screening microscopy. *J Biomol Screen* 14(8):944–955
- [43] Go AS, Hylek EM, Phillips KA, Chang Y, Henault LE, Selby JV, Singer DE (2001) Prevalence of diagnosed atrial fibrillation in adults: national implications for rhythm management and stroke prevention: the anticoagulation and risk factors in atrial fibrillation (atria) study. *JAMA* 285(18):2370–2375
- [44] Haney SA (ed.) (2008) *High Content Screening*. John Wiley & Sons, Inc., Hoboken, New Jersey
- [45] Hann MM, Oprea TI (2004) Pursuing the leadlikeness concept in pharmaceutical research. *Curr Opin Chem Biol* 8(3):255–263
- [46] Hoffmann E, Janko S, Reithmann C, Steinbeck G (2002) Auslösemechanismen von Vorhofflimmern. *Z Kardiol* 91:24–32
- [47] Hough P (1959) Machine analysis of bubble chamber pictures. In: *Proc. Int. Conf. High Energy Accelerators and Instrumentation*
- [48] Hough P (1962), *Methods and means for recognizing complex patterns*
- [49] Hüser J, Mannhold R, Kubinyi H, Folkers G (eds.) (2006) *High-Throughput Screening in Drug Discovery*. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim

- [50] ibidi GmbH. <http://www.ibidi.de/>
- [51] IBM Corporation. <http://www.ibm.com/>
- [52] International Society for Advancement of Cytometry.
<http://www.isac-net.org/>
- [53] IonOptix Corporation (2007) C-Dish User's Guide
- [54] IonOptix Corporation (2008) C-Pace EP User's Guide
- [55] IonOptix LLC. <http://www.ionoptix.com/>
- [56] ITW Chemische Produkte GmbH & Co KG. <http://www.itwcp.de/>
- [57] Jahangir A, Lee V, Friedman PA, Trusty JM, Hodge DO, Kopecky SL, Packer DL, Hammill SC, Shen WK, Gersh BJ (2007) Long-term progression and outcomes with aging in patients with lone atrial fibrillation: a 30-year follow-up study. *Circulation* 115(24):3050–3056
- [58] Jboss Inc, Jboss application server. <http://www.jboss.org>
- [59] Jones TR, Carpenter AE, Lamprecht MR, Moffat J, Silver SJ, Grenier JK, Castoreno AB, Eggert US, Root DE, Golland P, Sabatini DM (2009) Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. *Proc Natl Acad Sci U S A* 106(6):1826–1831
- [60] Jones TR, Kang IH, Wheeler DB, Lindquist RA, Papallo A, Sabatini DM, Golland P, Carpenter AE (2008) Cellprofiler analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics* 9:482
- [61] Kaestner L, Scholz A, Hammer K, Vecerdea A, Ruppenthal S, Lipp P (2009) Isolation and genetic manipulation of adult cardiac myocytes for confocal imaging. *J.Vis. Exp.* 31

- [62] Kaestner L, Tabellion W, Weiss E, Bernhardt I, Lipp P (2006) Calcium imaging of individual erythrocytes: problems and approaches. *Cell Calcium* 39(1):13–19
- [63] Kamensky LA, Melamed MR (1969) Instrumentation for automated examinations of cellular specimens. *Proceedings of the IEEE* 57:2007–2016
- [64] Karmonik C, York M, Grossman R, Kakkar E, Patel K, Haykal H, King D (2010) An image analysis pipeline for the semi-automated analysis of clinical fmri images based on freely available software. *Comput Biol Med* 40(3):279–287
- [65] Khanadeev VA, Khlebtsov BN, Staroverov SA, Vidyasheva IV, Skaptsov AA, Ileneva ES, Bogatyrev VA, Dykman LA, Khlebtsov NG (2010) Quantitative cell bioimaging using gold-nanoshell conjugates and phage antibodies. *J Biophotonics*
- [66] KiCad. http://www.lis.inpg.fr/realise_au_lis/kicad/
- [67] Knollmann BC, Roden DM (2008) A genetic framework for improving arrhythmia therapy. *Nature* 451(7181):929–936
- [68] Laboratory for Optical and Computational Instrumentation. <http://www.loci.wisc.edu/>
- [69] LabVIEW. <http://www.ni.com/labview/>
- [70] LabWindows/CVI. <http://www.ni.com/lwcvl/>
- [71] Lamprecht MR, Sabatini DM, Carpenter AE (2007) Cellprofiler: free, versatile software for automated biological image analysis. *Biotechniques* 42(1):71–75
- [72] Lenaerts I, Bito V, Heinzl FR, Driesen RB, Holemans P, D'hooge J, Heidbüchel H, Sipido KR, Willems R (2009) Ultrastructural and functional

- remodeling of the coupling between ca^{2+} influx and sarcoplasmic reticulum ca^{2+} release in right atrial myocytes from experimental persistent atrial fibrillation. *Circ Res* 105(9):876–885
- [73] Marr D, Hildreth E (1980) Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences* 207(1167):187–217
- [74] medGadget (2009), Internet journal of emerging medical technologies. <http://www.medgadget.com/>
- [75] Metaxas D, Whitaker R, Rittcher J, Sebastian T (eds.) (2006) Methods for high-content, high-throughput image-based cell screening., *Proceedings of the Workshop on Microscopic Image Analysis with Applications in Biology (MIAAB)*, Jones TR, Carpenter AE, Golland P, Sabatini DM, Copenhagen, Denmark, updated as a book chapter: Jones TR, Carpenter AE, Golland P, Sabatini DM (2008) Methods for high-content, high throughput image-based cell screening. In: Rittscher J, Machiraju R, Wong STC, eds. *Microscopic Image Analysis for Life Science Applications*. Norwood, MA: Artech House Publishers; 209-221. (Book chapter)
- [76] Meyer F (1994) Topographic distance and watershed lines. *Signal Processing* 38:113–125
- [77] Microsoft Corporation. <http://www.microsoft.com/>
- [78] Minimum Information About a Cellular Assay. <http://miaca.sourceforge.net/>
- [79] MISFISHIE Standard Working Group. <http://mged.sourceforge.net/misfishie/index.php/>
- [80] Müller O, Tian Q, Zantl R, Kahl V, Lipp P, Kaestner L (2010) A system for optical high resolution screening of electrical excitable cells. *Cell Calcium* 47(3):224–233
- [81] National Instruments. <http://www.ni.com/>

- [82] National Instruments Corporation, Application design patterns: Producer/consumer. <http://zone.ni.com/devzone/cda/tut/p/id/3023/>
- [83] National Instruments Corporation (2006) User Guide SCC-68
- [84] Nattel S, Li D, Yue L (2000) Basic mechanisms of atrial fibrillation—very new insights into very old ideas. *Annu Rev Physiol* 62:51–77
- [85] Naturwissenschaftliches und Medizinisches Institut. <http://www.nmi.de/>
- [86] OME-TIFF. <http://ome-xml.org/wiki/OmeTiffSpec/>
- [87] OME-XML. <http://www.ome-xml.org/>
- [88] OME Trac (2009), Running cellprofiler with omero.blitz. <http://trac.openmicroscopy.org.uk/omero/wiki/OmeroCellProfiler/>
- [89] OMRON Corporation. <http://www.omron.com/>
- [90] Ontology Working Group. <http://mged.sourceforge.net/ontologies/index.php/>
- [91] Open Microscopy Environment, Omero.fs and omero.dropbox support website. <http://www.openmicroscopy.org/site/support/omero4/server/fs/>
- [92] Panasonic Corporation. <http://www.panasonic.net/>
- [93] Pechiney Plastic Packaging Company. <http://www.packworld.com/>
- [94] Perkin Elmer (2009) OPERA - The Ultra High Speed, High Resolution Solution for High Content Screening
- [95] PostgreSQL. <http://postgresql.de/>
- [96] Prewitt JMS (1970) "Object Enhancement and Extraction", in *Picture Processing and Psychopictorics*. Academic Press, New York

- [97] Puchkov EO (2010) Brownian motion of polyphosphate complexes in yeast vacuoles: characterization by fluorescence microscopy with image analysis. Yeast
- [98] Qeli E, Wiechert W, Freisleben B (2005) Visual exploration of time-varying matrices. Proceedings of the 2005 International Conference on Information Visualization, IEEE Press, London, UK :889–895
- [99] Quacom Informationssysteme GmbH. <http://www.quacom.com/>
- [100] Ramakrishnan R, Gehrke J (2003) Database Management Systems. McGraw-Hill Professional
- [101] Rasband W (1997–2010), Imagej. <http://rsb.info.nih.gov/ij/>, u. S. National Institutes of Health, Bethesda, Maryland, USA
- [102] Robenek H (ed.) (1995) Mikroskopie in Forschung und Praxis. GIT Verlag GmbH, Darmstadt
- [103] Roberts LG (1965) "Machine Perception of Three-Dimensional Solids", in Optical and Electro-Optical Information Processing. MIT Press, Cambridge, Mass.
- [104] Russell S, Norvig P (2003) Artificial Intelligence - A Modern Approach (Second Edition). Prentice Hall
- [105] Samba. <http://www.samba.org/>
- [106] Schiebler TH (2005) Anatomie. Springer-Verlag, Berlin, Heidelberg, New York
- [107] Schildt H (2003) C/C++ GE-PACKT. mitp-Verlag, Bonn
- [108] Schildt H, O'Neil J (2005) Java 5 GE-PACKT. mitp-Verlag
- [109] Schmidt DC (1995) Acceptor - a design pattern for passively initializing network services. EuroPLoP '95

- [110] Schmidt DC (1996) Connector. EuroPLoP '96
- [111] Schunk-Gruppe. <http://www.schunk-group.com/>
- [112] Shorte SL, Frischknecht F (eds.) (2007) Imaging Cellular and Molecular Biological Functions. Springer-Verlag, Berlin Heidelberg
- [113] Sobel I, Feldman G A 3x3 isotropic gradient operator for image processing, presented at a talk at the Stanford Artificial Project in 1968, unpublished but often cited, orig. in Pattern Classification and Scene Analysis, Duda,R. and Hart,P., John Wiley and Sons,'73, pp271-2
- [114] Spornitz UM (1996) Anatomie und Physiologie. Springer-Verlag, Berlin
- [115] Sun Microsystems. <http://java.sun.com/>
- [116] Swedlow JR, Goldberg IG, Eliceiri KW, Consortium OME (2009) Bioimage informatics for experimental biology. Annu Rev Biophys 38:327–346
- [117] Swedlow JR, Lewis SE, Goldberg IG (2006) Modelling data across labs, genomes, space and time. Nat Cell Biol 8(11):1190–1194
- [118] Systems Biology Markup Language. <http://sbml.org/>
- [119] Tallini YN, Ohkura M, Choi BR, Ji G, Imoto K, Doran R, Lee J, Plan P, Wilson J, Xin HB, Sanbe A, Gulick J, Mathai J, Robbins J, Salama G, Nakai J, Kotlikoff MI (2006) Imaging cellular signals in the heart in vivo: Cardiac expression of the high-signal ca²⁺ indicator gcamp2. Proc Natl Acad Sci U S A 103(12):4753–4758
- [120] Taylor DL, Haskins JR, Giuliano KA (eds.) (2007) High Content Screening. Human Press, Totowa New Jersey
- [121] Tektronix Inc. <http://www.tek.com/>
- [122] The MathWorks. <http://www.mathworks.de/>

- [123] The MathWorks, Technical solutions - how do i increase the heap space for the java vm in matlab 6.0 (r12) and later versions?
<http://www.mathworks.de/support/solutions/en/data/1-1812C/index.html>
- [124] The NaradaBrokering Project. <http://www.naradabrokering.org/>
- [125] The Open Biological and Biomedical Ontologies. <http://obofoundry.org/>
- [126] The Open Microscopy Environment. <http://www.openmicroscopy.org/>
- [127] The usable image project. <http://www.usableimage.org/>
- [128] TILL Photonics GmbH. <http://www.till-photonics.com/>
- [129] TOELLNER Electronic Instrumente GmbH. <http://www.toellner.de/>
- [130] Viero C, Kraushaar U, Ruppenthal S, Kaestner L, Lipp P (2008) A primary culture system for sustained expression of a calcium sensor in preserved adult rat ventricular myocytes. *Cell Calcium* 43(1):59–71
- [131] Wang Y, Shyy JYJ, Chien S (2008) Fluorescence proteins, live-cell imaging, and mechanobiology: seeing is believing. *Annu Rev Biomed Eng* 10:1–38
- [132] Wijffels MC, Kirchhof CJ, Dorland R, Allessie MA (1995) Atrial fibrillation begets atrial fibrillation. a study in awake chronically instrumented goats. *Circulation* 92(7):1954–1968
- [133] Wolf PA, Dawber TR, Thomas HE, Kannel WB (1978) Epidemiologic assessment of chronic atrial fibrillation and risk of stroke: the framingham study. *Neurology* 28(10):973–977
- [134] ZeroC™, Internet communications engine (ice). <http://www.zeroc.com/>

Appendix A

Documentations

A.1 Pulsing Device API

The following pages show the API of the Pulsing Device software. It describes the functions available in the ANSI C interface. These functions can be accessed by any software supporting an ANSI C interface and thus allow an external control of the Pulsing Device. The arivis Grid interface of the Pulsing Device software makes use of the functions described in the Pulsing Device API.

Pulsing Device ANSI C Interface (Version 20090625)

Note: Functions written in **BOLD** are accessible via the arivisGrid software ...

functions.h**Pulsing Mode**

```
DEFAULT_MODE = "0";

// HIFN Sets the the stimulation mode: "multiple .pp" (0) or
// "single .pp" (1) for acute pulsing procedure.
// HIPAR mode/Input: mode to select.
void setMode(char *mode);

// HIFN Returns the mode of the pulsing procedure ("multiple .pp" (0) or
// "single .pp" (1) for acute pulsing procedure).
char *getMode();
```

Stimulation Plate

```
DEFAULT_PLATE_SIZE = 24;

// HIFN Sets the size of the HCS Stimulation Plate (i.e. number of
// compartments).
// HIPAR size/Input: size of the HCS Stimulation Plate.
void setStimulationPlateSize(int size);

// HIFN Returns the size of the HCS Stimulation Plate (i.e. number of
// compartments).
int getStimulationPlateSize();
```

Stimulation Voltage

```
DEFAULT_DELTA_V = 0.0;

// HIFN Sets the voltage change (delta V) if the image processing software
// does not measure any cell movements.
// HIPAR delta_V/Input: the voltage to be added(+) / subtracted (-) to/
// from the values defined in the pulsing protocols.
void setVoltageChange(double deltaV);

// HIFN Returns the voltage change DELTA_V [V].
double getVoltageChange();
```

```

Pulse Protocols

// HIFN Populates the protocol array with pulse protocols.
// HIPAR compartmentID/Input: the index of the protocol array, where the
// protocol will be stored.
// HIPAR protocol/Input: the path to the pulse protocol file (.pp).
void setPulseProtocol(int compartmentID, char *protocol);

// HIFN Returns the protocol of a given compartment.
char *getPulseProtocol(int compartmentID);

_acutePulseProtocol = "";
_chronicalPulseProtocol = "";

// HIFN Populates the protocol array with a chronical pulse protocol
// (0..23).
// HIPAR protocol/Input: the path to the pulse protocol file (.pp).
void setChronicalPulseProtocol(char *protocol);

// HIFN Returns the chronical pulse protocol.
char *getChronicalPulseProtocol();

// HIFN Populates the protocol array with an acute pulse protocol.
// HIPAR compartmentID/Input: the index of the protocol array, where the
// protocol will be stored.
// HIPAR protocol/Input: the path to the pulse protocol file (.pp).
void setAcutePulseProtocol(int compartmentID, char *protocol);

// HIFN Returns the acute pulse protocol.
char *getAcutePulseProtocol();

// HIFN Displays the whole protocol array.
// HIRET Returns a pointer to the protocol array.
char *displayStimulationPlateLayout();

DEFAULT_DIRECTORY = "C:\\";

// HIFN Sets the default directory of the pulse protocols.
// HIPAR directory/Input: the path to the default directory of the pulse
// protocols.
void setPulseProtocolDirectory(char *directory);

// HIFN Returns the default directory of the pulse protocols.
char *getPulseProtocolDirectory();

// HIFN Returns a list of .pp files in the default directory.
// HIRET Returns a pointer to an array of characters which contains a list
// of .pp files in the default directory.
char **getAvailablePulseProtocols();

```

Pulse Protocols (cont.)

```

// HIFN Modifies the HCS Stimulation Plate Layout by replacing a specified
// protocol by a new one.
// HIPAR oldProtocol/Input: the pulse protocol to be replaced.
// HIPAR newProtocol/Input: the pulse protocol to be used instead of
// oldProtocol.
void switchPulseProtocol(char *oldProtocol, char *newProtocol);

// HIFN Updates the protocol array, stops and re-starts pulsing process
// with new parameters.
// HIPAR compartmentID/Input: the index of the protocol array, where the
// protocol will be stored.
// HIPAR protocol/Input: the path to the pulse protocol file (.pp).
void changePulseProtocol(int compartmentID, char *protocol);

```

Functions directly interacting with the LabVIEW-DLL

```

// HIFN Starts a pulsing process by making use of the DLL function
// "controlPulsingDevice" (see PulsingDevice.h). Protocol delimiter
// is "\t " (handled inside the LabVIEW DLL).
void startPulsingProcessDLL();

// HIFN Starts startPulsingProcessDLL() as a secondary thread.
int CVICALLBACK startPulsingThread(void *functionData);

// HIFN Terminates startPulsingThread() thread with ID gTID.
int CVICALLBACK stopPulsingThread(void *functionData);

// HIFN Initially starts and stops pulsing thread to avoid harming cells
// due to initial CPU processing/loading time.
void init();

// HIFN Starts startPulsingThread() (secondary thread) and returns to the
// main thread, allowing other processes being executed in parallel.
// Recommendation: use this function after calling init() (then, the
// DLL is loaded, no delay, no harm to the cells ...)
void startPulsingProcess();

// HIFN Starts stopPulsingThread() (secondary thread) and returns to the
// main thread.
void stopPulsingProcess();

// HIFN Calls stopPulsingProcess() and startPulsingProcess() in order to
// update the DLL state after changing the settings "on the fly".
void updatePulsingProcess();

```

PulsingDevice.h

Note: The following functions are provided by LabVIEW VIs of the DLL:

```
void __cdecl controlPulsingDevice(char Mode[], LVBoolean *StartStop,
                                char ProtocolString[],
                                double VoltageChangeDeltaV);

void __cdecl initHardware(void);

long __cdecl LVDLLStatus(char *errStr, int errStrLen, void *module);
```

PulsingDeviceActorC.h

Note: The following functions are used to access the arivisGrid:

```
/// HIFN Helper function for parsing user input /* K&R2 p29 */
int getline(char s[], int lim);

/// HIFN Connects to the arivisGrid using "narada-tcp" and "localhost".
/// HIRET Returns the state of the connection (0: ok, -1: failed).
int connect();

/// HIFN Disconnects from the arivisGrid.
int disconnect();

/// HIFN Registers an actor based on "pulsingDeviceCapabilities.xml"
to the grid.
int registerActor();

/// HIFN Receives commands from the grid.
void receivedCommand(char *commandID, HPAYLOAD payload, char *sender);

/// HIFN Interprets commands from the grid and calls the corresponding
Pulsing Device functions.
void interpretUserInput(char *commandID, HPAYLOAD payload);

/// HIFN Keeps the grid listener alive.
int startGridListener();
```

Pulsing Device ANSI C Interface

Documentation

July 2009

Grid commands:

- Configure `<int CompartmentID> <string ProtocolID> <bool isAcute?>`
- SwitchPulseProtocol `<string OldProtocolID> <string NewProtocolID>`
- StartPulsing
- StopPulsing
- UpdatePulsingProcess
- SetVoltageChange `<double deltaV>`
- StartPerfusion `<int CompartmentID>`
- StopPerfusion

A.2 Perfusion System API

The following pages show the API of the Perfusion System software. The design of this API is similar to the Pulsing Device API described in the previous section.

Perfusion System ANSI C Interface (Version 20091202)

Note: Functions written in **BOLD** are accessible via the arivisGrid software ...

functions.h**Perfusion:**

```
void setPerfusion(int compartmentID);  
int getPerfusion();
```

Functions directly interacting with the LabVIEW-DLL

```
int CVICALLBACK startPerfusionSystemThread(void *functionData);  
int CVICALLBACK stopPerfusionSystemThread(void *functionData);  
void startPerfusionSystem();  
void stopPerfusionSystem();
```

Perfusion.h

Note: The following functions are provided by LabVIEW VIs of the DLL:

```
void __cdecl Window1_perfusion_system(uint32_t Optionsfelder);  
void __cdecl ControlPerfusionSystem(int8_t CompartmentID);  
void __cdecl Cvi_perfusion_system(uint32_t Optionsfelder);  
long __cdecl LVDLLStatus(char *errStr, int errStrLen, void *module);
```

PerfusionActorC.h

Note: The following functions are used to access the arivisGrid:

```
/// HIFN Helper function for parsing user input /* K&R2 p29 */
int getline(char s[], int lim);

/// HIFN Connects to the arivisGrid using "narada-tcp" and "localhost".
/// HIRET Returns the state of the connection (0: ok, -1: failed).
int connect();

/// HIFN Disconnects from the arivisGrid.
int disconnect();

/// HIFN Registers an actor based on "perfusionCapabilities.xml"
/// to the grid.
int registerActor();

/// HIFN Receives commands from the grid.
void receivedCommand(char *commandID, HPAYLOAD payload, char *sender);

/// HIFN Interprets commands from the grid and calls the corresponding
/// Perfusion System functions.
void interpretUserInput(char *commandID, HPAYLOAD payload);

/// HIFN Keeps the grid listener alive.
int startGridListener();
```

Grid commands:

- StartPerfusion <int CompartmentID>
- StopPerfusion

Appendix B

Tables

Table B.1: Overview of implemented XML commands and their interrelation to the buttons of the GUI.

Button	XML command	Description
Initialise	<code><INIT/></code>	sets all TTL levels to low
Wait until user interacts	<code><UI_WAIT/></code>	sets a pause until the user interacts with the system
Wait	<code><WAIT time="..."/></code>	sets a pause with a specified duration
Trigger Channel	<code><TRIGGER></code> <code><CHANNEL name="1" value="HIGH" time="..."/></code> <code><CHANNEL name="2" value="HIGH" time="..."/></code> <code><CHANNEL name="3" value="HIGH" time="..."/></code> <code><CHANNEL name="4" value="HIGH" time="..."/></code> <code></TRIGGER></code>	sets the TTL level of four channels either high or low
	<code><SCRIPT>...</SCRIPT></code>	defines beginning and end of a script

Table B.2: Components of the SSRD.

Component	Quantity	Description
SSR Panasonic AQY225R1S	48	controlled via NI PCI-6229
4 pin single in line (SIL) connector (male)	48	pin connector of the SSR modules; 2 pins are control voltage 2 pins are load voltage
4 pin SIL connector (female)	48	socket for the SSR modules; 2 pins are control voltage 2 pins are load voltage
3 pin SIL connector (female)	1	connector for the voltage source; one pin is +, one pin is –
3 pin SIL connector (male)		one pin is shared ground
34 pin ribbon cable socket connector	2	control voltage of the SSR
50 pin ribbon cable socket connector	1	load voltage to the HCS stimulation plate $((24 \times load + 1 \times GND) \times 2$ rows of pins)

Table B.3: Pin layout of the control voltage circuit (NI SCC-68), patch bay routing and of the HCS stimulation plate.

NI SCC-68		patch bay		HCS stimulation plate	
connector 0	connector 1	input	output	input	well number
–	–	49, 50	1, 2	1, 2	GND
1 (D GND)	1 (D GND)	–	–	–	–
2 (P0.0)	2 (P0.8)	47, 48	49, 50	49, 50	A1 / 0
4 (P0.1)	3 (P0.9)	45, 46	41, 42	41, 42	A2 / 1
3 (P0.2)	4 (P0.10)	43, 44	33, 34	33, 34	A3 / 2
6 (P0.3)	5 (P0.11)	41, 42	25, 26	25, 26	A4 / 3
5 (P0.4)	6 (P0.12)	39, 40	17, 18	17, 18	A5 / 4
8 (P0.5)	7 (P0.13)	37, 38	9, 10	9, 10	A6 / 5
7 (P0.6)	8 (P0.14)	35, 36	47, 48	47, 48	B1 / 6
10 (P0.7)	9 (P0.15)	33, 34	39, 40	39, 40	B2 / 7
9 (P1.0)	10 (P0.16)	31, 32	31, 32	31, 32	B3 / 8
12 (P1.1)	11 (P0.17)	29, 30	23, 24	23, 24	B4 / 9
11 (P1.2)	12 (P0.18)	27, 28	15, 16	15, 16	B5 / 10
14 (P1.3)	13 (P0.19)	25, 26	7, 8	7, 8	B6 / 11
13 (P1.4)	14 (P0.20)	23, 24	45, 46	45, 46	C1 / 12
16 (P1.5)	15 (P0.21)	21, 22	37, 38	37, 38	C2 / 13
15 (P1.6)	16 (P0.22)	19, 20	29, 30	29, 30	C3 / 14
18 (P1.7)	17 (P0.23)	17, 18	21, 22	21, 22	C4 / 15
17 (P2.0)	18 (P0.24)	15, 16	13, 14	13, 14	C5 / 16
20 (P2.1)	19 (P0.25)	13, 14	5, 6	5, 6	C6 / 17
19 (P2.2)	20 (P0.26)	11, 12	43, 44	43, 44	D1 / 18
22 (P2.3)	21 (P0.27)	9, 10	35, 36	35, 36	D2 / 19
21 (P2.4)	22 (P0.28)	7, 8	27, 28	27, 28	D3 / 20
24 (P2.5)	23 (P0.29)	5, 6	19, 20	19, 20	D4 / 21
23 (P2.6)	24 (P0.30)	3, 4	11, 12	11, 12	D5 / 22
26 (P2.7)	25 (P0.31)	1, 2	3, 4	3, 4	D6 / 23

Table B.4: Patch bay components.

Component	Quantity	Description
50 pin ribbon cable connector	2	input load voltage from SSRD; output load voltage to HCS stimulation plate
12 way terminal block with push-button clamps	3	allows for an easy reconnection of conduction paths
patch wires	50	routing of 24× output voltage +1× GND

Table B.5: Connector pin assignment of the control unit. Pins 1-48 are connected to the negative terminal, pins 49 and 50 are connected to the positive terminal of the power supply.

Output pin	Valve ID	Output pin	Valve ID
1, 2	A1 / 0	27, 28	C2 / 13
3, 4	A2 / 1	29, 30	C3 / 14
5, 6	A3 / 2	31, 32	C4 / 15
7, 8	A4 / 3	33, 34	C5 / 16
9, 10	A5 / 4	35, 36	C6 / 17
11, 12	A6 / 5	37, 38	D1 / 18
13, 14	B1 / 6	39, 40	D2 / 19
15, 16	B2 / 7	41, 42	D3 / 20
17, 18	B3 / 8	43, 44	D4 / 21
19, 20	B4 / 9	45, 46	D5 / 22
21, 22	B5 / 10	47, 48	D6 / 23
23, 24	B6 / 11	49, 50	plus terminal
25, 26	C1 / 12		

Appendix C

Programs

The enclosed CD contains the programs (state: July 2010) described in this dissertation.

Publications ¹

Papers

2010

→ Oliver Müller, Qinghai Tian, Roman Zantl, Valentin Kahl, Peter Lipp, Lars Kaestner: **A System for Optical High Resolution Screening of Electrical Excitable Cells.** *Cell Calcium*.

2007

Ingolf Sommer, Oliver Müller, Francisco S. Domingues, Oliver Sander, Joachim Weickert & Thomas Lengauer: **Moment Invariants as Shape Recognition Technique for Comparing Protein Binding Sites.** *Bioinformatics*.

Posters and Abstracts

2010

→ Oliver Müller, Lars Kaestner, Peter Lipp: **ATOM - Automated Import of Image Data into OMERO.** (*poster at the OME User's Meeting*).

→ Lars Kaestner, Qinghai Tian, Oliver Müller, Sandra Ruppenthal, Peter Lipp: **High Resolution Microscopy Meets High Content Screening.** (*abstract at the Focus on Microscopy conference (FOM2010)*).

¹Publications relevant for this dissertation are marked with an arrow.

2009

→ Lars Kaestner, Oliver Müller, Aline Flockerzi, Karin Hammer, Wiebke Tabellion, Qinghai Tian, Sandra Ruppenthal, Anke Scholz, Peter Lipp: **Towards Cardiac Safety Screens by Single Cell Imaging Procedures.** *Biophys. J.*

→ Lars Kaestner, Qinghai Tian, Oliver Müller, Aline Flockerzi, Karin Hammer, Sandra Ruppenthal, Anke Scholz, Peter Lipp: **Concepts for Optical High Content Screens of Excitable Primary Isolated Cells for Molecular Imaging.** (*abstract at the European Conferences on Biomedical Optics (ECBO2009)*).

2008

→ Lars Kaestner, Oliver Müller, Aline Flockerzi, Sandra Ruppenthal, Anke Scholz, Peter Lipp: **Towards High Content Screening of Primary Cultured Adult Cardiac Myocytes.** (*abstract at the Focus on Microscopy conference (FOM2008)*).

2006

Oliver Müller, Ingolf Sommer, Francisco Domingues, Oliver Sander, Hongbo Zhu, Thomas Lengauer: **Using Shape Retrieval Techniques for Identifying Similar Protein Binding Sites.** (*poster at the German Conference on Bioinformatics (GCB2006)*).

Acknowledgements

I would like to express my gratitude to Prof. Dr. Peter Lipp for giving me the opportunity to work at the Institute for Molecular Cell Biology as well as for providing a very exciting and interesting subject area.

I am very thankful to Dr. Lars Kaestner for his constant availability, excellent supervision and for many valuable discussions.

For providing Figure 1.3 and his experimental data in Section 3.5.3 I would like to thank my room mate Qinghai Tian.

Many thanks go to all group members for a very pleasant working atmosphere as well as for their confidence with respect to my work as a computer administrator.

My special thanks go to all proofreaders.

For technical support during my work I would like to thank Christoph Neuhardt, Jörg Sauerbaum and Christian Schneider.

For their kind support during the connection of our software to the arivis Grid I would like to thank Christian Götze und Matthias Rust (arivis Multiple Image Tools GmbH).

I am very thankful to Jason Swedlow and the OME developer team for their hospitality during the OME developer's meeting at Dundee as well as for the interesting user's meetings at Paris.

I would also like to thank my friends and families for their loving support during the crucial phase of my dissertation.

For their patience and thoughtfulness, especially during the writing phase of my thesis I would like to thank my band Joker's Drive.

Finally, I would like to express my sincere thanks to my wife Lisa for her daily support with both, words and deeds. Her patience and her love have contributed to the success of this work.

This work was supported by the Federal Ministry of Education and Research (BMBF).

curriculum vitae



Personal Data

Surname / Forename Müller, Oliver
Address Tränkenweg 22, 66540 Neunkirchen, Germany
Phone +49 (0) 1520 / 174 92 13
E-mail oliver.mueller@uks.eu
Date of Birth 04.10.1977
Family Status married

Work Experience

Period 01.05.2007 until today
Job or Position starting with the promotion: scientific assistant
Most Important Tasks network administration, web administration and maintenance of the IT at the Institute of Molecular Cell Biology, University of Saarland
Reference Prof. Dr. Peter Lipp (peter.lipp@uks.eu)

Period 01.10.2006 - 30.04.2007
Job or Position after the master's thesis: scientific assistant
Most Important Tasks software engineering in the department Bioinformatics and Applied Algorithmics, Max-Planck Institute for Informatics
Reference Prof. Dr. Thomas Lengauer, Ph.D. (lengauer@mpi-sb.mpg.de)

Period 01.04.2005 - 30.09.2006
Job or Position student research assistant
Most Important Tasks software engineering in the department Bioinformatics and Applied Algorithmics, Max-Planck Institute for Informatics
Reference Dr. Ingolf Sommer (sommer@mpi-sb.mpg.de)

Period 01.11.2004 - 28.02.2005
Job or Position student research assistant
Most Important Tasks database management at the Chair for Computer Graphics, University of Saarland
Reference Dipl. Inf. Georg Demme (gdemme@graphics.cs.uni-sb.de)

Period 01.08.2003 - 31.10.2004
Job or Position student research assistant
Most Important Tasks software development and network administration in the company Mecadi GmbH Chemicals / Processing, biomedical centre Homburg / Saar

Period 01.07.2002 - 31.07.2003
Job or Position founder member of the students council of bioinformatics
Most Important Tasks web administration, tutor and student counsellor at the students council of informatics / bioinformatics, University of Saarland

School Education	
Acquired Qualification	university-entrance diploma
Date	15.07.1998
Basic Military Service	
Period	01.07.1998 - 30.04.1999
Academic Studies	
Period	01.05.2007 until today
Acquired Qualification	Dr. rer. nat. (prospectively in July 2010)
Main Subject	bioinformatics
Subject Area	image based high-content screening
Educational Facility	Institute for Molecular Cell Biology, University of Saarland supervised by Prof. Dr. Peter Lipp
Period	01.10.2004 - 30.09.2006
Acquired Qualification	Master of Science
Main Subject	bioinformatics
Subject Area	3D object recognition of protein binding sites
Educational Facility	Max-Planck Institute for Informatics / University of Saarland supervised by Prof. Dr. Thomas Lengauer
Period	01.10.2001 - 30.09.2004
Acquired Qualification	Bachelor of Science
Main Subject	bioinformatics
Subject Area	computer based design of peptides
Educational Facility	Chair of Bioinformatics, University of Saarland supervised by Prof. Dr. Volkhard Helms
Period	01.10.1999 - 30.09.2001
	start with the study of computer science, then change to the newly established degree programme bioinformatics
Awards and Scholarship	
	<i>October of 2007 - October of 2009</i> doctoral scholarship (Landesgraduiertenförderungsgesetz)
	<i>September of 2006</i> best poster award of the German Conference on Bioinformatics
Personal Skills and Expertise	
Native Language	- German
Other Languages	- English (fluently) - French (basic knowledge) - Luxembourgian (basic knowledge)

