A Framework for Optical Inspection Applications in Life-Science Automation

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von:

Kai Ritterbusch, Rostock, 2012

urn:nbn:de:gbv:28-diss2012-0118-2

Gutachter:

1. Prof. Thurow, Universität Rostock, IEF, Institut für Automatisierungstechnik
2. Prof. Müller, Universität Rostock, IEF, Institut für Nachrichtentechnik
3. Prof. Jovanov, University of Alabama, Huntsville, USA

Datum der Verteidigung: 5. Juli 2012

Danksagung

An erster Stelle möchte ich mich bei meinen Betreuern Frau Prof. Dr. Kerstin Thurow und Herrn Prof. Dr. Norbert Stoll für die Idee zum Thema der Arbeit, ihrer Ermöglichung und ihrer Betreuung während der gesamten Zeit bedanken. Weiterhin möchte ich mich bei Prof. Emil Jovanov für die Gastfreundschaft während meines Aufenthalts an der University of Alabama in Huntsville und für die Ideen, die durch unsere Diskussionen entstanden sind, bedanken.

Mein herzlicher Dank geht an meine Kollegen Thomas Roddelkopf, Hans-Joachim Stiller und Steffen Junginger für ihre Hilfe und unsere oft lehrreichen Diskussionen, und an Lars Woinar, Heiko Engelhardt und Grit Koch für ihre großartige Unterstützung.

Besonders möchte ich mich auch bei meinen Eltern bedanken, die mich über meinen langen Ausbildungsweg hinweg immer und aus ganzem Herzen unterstützt haben. Der vielleicht größte Dank gebührt Katja, die trotz der nötigen Umzüge, der langen Trennung und aller daraus entstandenen Anstrengungen meine Entscheidung diese Arbeit zu schreiben immer unterstützt hat und mir immer zur Seite stand.

Contents

List of Figures

List of Tables

Listings

Abbreviations

Acronyms

Physical measurements are stated in units according to the International System of Units (SI) and are not listed here.

ALA	Association for Laboratory Automation
ALP	automation labware position
AOI	automated optical inspection
API	application program interface
C++	C++ programming language (as in ISO/IEC 14882:2003[40])
CCD	charge coupled device
CIS	contact image sensor
CUDA	Compute Unified Device Architecture (Nvidia Corp., Santa Clara, CA, USA)
CV	computer vision
COTS	commercial off-the-shelf
CPU	central processing unit
DG	Data Group (part of the TWAIN protocol)
DAT	Data (part of the TWAIN protocol)

DMSO	dimethyl sulfoxide
DNA	deoxyribonucleic acid
DoF	degrees of freedom
dpi	dots per inch
DS	Data Source (part of the TWAIN protocol)
DSM	Data Source Manager (part of the TWAIN protocol)
FIS	fuzzy inference system
FDA	Federal Drug Administration (USA)
FP	false positives
GPU	graphics processing unit
GUI	graphical user interface
GUM	guide to the expression of uncertainty in measurement
HAL	hardware abstraction layer
HCS	high content screening
LIMS	laboratory information management system
LLD	liquid level detection
MFC	Microsoft Foundation Classes
MMI	manual module interface
MTP	microtiter plate
OpenCV	Open Computer Vision Library
PC	personal computer
PCS	process control system

px	pixel
ROC	receiver operating characteristic
ROI	region of interest
SAMI	Sagian Automated Methods Interface
SBS	Society for Biological Screening (now part of ALA)
SILAS	SAMI's interprocess communication interface
SIMD	single instruction multiple data
SMP	symmetric multiprocessing (a shared memory multiprocessing system)
SI	International System of Units (Système international d'unités)
TLA	total laboratory automation
TP	true positives
QA	quality assurance (also called QC)
QC	quality control (also called QA)

Symbols

Vectors and matrices are marked with a single / double underline respectively.

A	accuracy
AUC	area under curve
C	calibration pattern coordinate system
D	device coordinate system
$F1S$	F-measure (also called F1-Score)
FPR	false positive rate

I	image coordinate system
L	labware coordinate system
P	precision
R	result space
R	rotation matrix
S	scale matrix
TPR	true positive rate
W	well coordinate system
c_v	coefficient of variation
err	reprojection error
rms	root mean square
l	labware mapping function
p	point in a coordinate space
t	time
t	translation vector
x	x-Axis
y	y-Axis
z	z-Axis
σ	standard deviation
ζ	sensor-fixed coordinate system

Indices

Coordinate Systems:

c	calibration pattern coordinate system
d	device coordinate system
i	image coordinate system
l	labware coordinate system
r	robot coordinate system
w	well coordinate system
ζ	sensor-fixed coordinate system

Other Indices:

r	result

Chapter 1

Introduction

As in every field of electronics, the price for digital imaging devices trends to decrease, while their performance in terms of image quality, physical dimensions and weight increases [37]. With the increasing processing power of today's computers, computer vision (CV) systems have become widely accepted. They are used successfully in industrial automation for quality control (e.g. defect and object detection [48]) and for process control (e.g. sorting machines [65]). These applications fall into the category of automated optical inspection (AOI) tasks.

In life science laboratories however, CV is nearly exclusively used for data analysis purposes in plate readers and microscopes. Exceptions are colony pickers, which select colonies using machine vision and pick those appropriate for further processing [44] [33], and barcode readers.

The progress made in AOI was ignored by laboratory robotics equipment vendors. One may say that this is a good indicator that AOI is not needed in laboratory automation, but this conclusion is wrong. One cause for the slow adoption is that classical approaches often exist and work. However, this thesis identifies tasks within the fields of quality control and process control of automated laboratories, where machine vision can be a viable and cost-effective alternative to classical approaches.

Many applications exist where CV offers a fast and economical alternative to classical approaches, and it furthermore offers possibilities to automate tasks that were difficult or impossible to automate without [29] [71]. Furthermore, tasks belonging to sample preparation are still considered as difficult to automate. Here, CV can provide the much needed sensory input to the automation system that is required to solve this problem.

While digital cameras are available in every mobile phone today, CV algorithms still belong to the challenging topics of signal processing research and only recently have moved into the everyday lives of people (e.g. face detection in digital cameras, gesture controlled video games). That means while hardware prices decrease, one hindrance to the adoption of machine vision applications persist: The high cost of software development.

Lebak et al. state that "the cost of software development (for signal processing applications) outweighs the hardware development by a significant and widening gap" [50]. This is also true in the field of laboratory automation, where companies sell COTS hardware equipped with their own software, for e.g. barcode-reading, for as much as a tenfold of the hardware price.

While the requirements in terms of handling and robustness of the software are as high as for any professional application, the quantities in which companies sell their specialized equipment for laboratory automation is small compared to mass market software. Hence, the small market quantities typical for specialized equipment are one cause for high prices. Probably, this is one cause for the slow adoption of computer vision in laboratories.

This problem is addressed by the design of a software framework that enables developers to accelerate application development and reducing costs. The framework is a generic tool to implement plate analysis for different labware types using an arbitrary imaging device. It implements computer vision and machine learning algorithms together with providing test and simulation facilities for the developer. The interfaces to the process control system, to the imaging device and to the user are kept abstract, in a way that enables the rapid integration of different applications into laboratories.

The work on a machine vision-based liquid delivery monitoring application is presented as the second part of this thesis. It aims to monitor the delivery of smallest amounts of liquids to the widely used standard microtiter plate labware format [73] [74]. This application is an example for an AOI task and is used to validate the framework design and implementation. The algorithmic approach for the detection of drops in transparent microtiter plate wells has been tested and validated in two test runs.

1.1 Computer Vision for Automated Optical Inspection

The term computer vision (CV), as it is used in this thesis, circumscribes the computer-based understanding of images, in order to generate abstract knowledge that can be stored as a datum or can be acted upon [23]. Synonyms are *machine vision* and *industrial image processing*. Even though industrial image processing often uses one-dimensional *line scan* cameras, the area of application in, e.g., quality control has a similar program layout, algorithms and aims (i.e. generation of measurements for quality and process control).

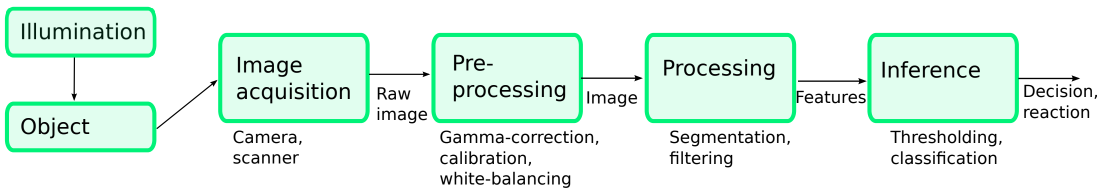


Figure 1.1: Flow scheme of a computer vision program

Computer vision in the scope of this document is used for measurement applications within automation systems and comprises the steps shown in Fig. 1.1 [41]:

1. **Sample illumination:** The setup and positioning of a light source that emits light, which is reflected by the objects of interest. The illumination setup affects the quality of an image to a great extent, hence an application whose illumination setup is controllable is advantageous and is likely to offer better results. Methods to determine optimal lighting exist [17] [54], but they rely on simplified models, such that an experimental verification is most likely necessary. Yi et al. state that active illumination design is more efficient than the application of more complex algorithms for suboptimal images [86].

2. **Imaging:** Imaging describes the spatially resolved conversion of electromagnetic waves into a signal such as voltage. Generally, any sensor able to read electromagnetic waves is appropriate, but for the scope of this document, only visible light is considered ($380nm - 780nm$). The sensor selection and positioning is carried out with consideration of the laws of linear optics (pinhole camera model). Instructions can be found in [23] and [64]. *Cowan et al.* even present an algorithm for automatic optimal positioning of cameras [16].

3. **Preprocessing** covers the correction of systematic measurement errors by calibration and other adjustments such as white balancing and histogram equalization. Preprocessing measures are described in-depth in [23] and [41].

4. **Processing:** The main part of every CV application consists of the steps image

segmentation, morphology, feature generation and object classification. Depending on the algorithms, the sequence of steps will be different to the ones presented here.

Segmentation: Generation of a map that holds the information for every pixel, whether it belongs to the object of interest (*true*) or not (*false*). Classical techniques, such as edge-detectors [14] or thresholding [62], are applied at this stage, but advanced pattern recognition algorithms are also used for image segmentation.

Feature generation: The extraction of descriptive parameters (features) for every object that was found during segmentation. Classic features used in CV are brightness levels or color, statistical moments, lines [39] and curves [20], aspect ratios, etc. For every object, its features form a feature vector.

5. **Inference:** An inference engine classifies objects and determines decisions or reactions accordingly. With feature vectors as inputs, this part forms the *intelligent* part of the system when it is trainable or when it is designed to e.g. automatically learn while operating. Example techniques used in this step include statistical methods and so-called artificial intelligence methods. Examples can be found for *artificial neural networks* (*ANN*), *fuzzy inference engines* (*FIM/FIS*), coupled *neuro-fuzzy systems*, and statistical classifiers, such as *Bayes*, *k-means*, *boosting* [75] and *support vector machines* (*SVM*). Classifiers and inference engines are classified by their learning rule into supervised and unsupervised learning. [9].

The list above describes the classical steps implemented in a CV application. Today, variations and exceptions to that sequence exist. Latest research aims to skip the application-specific image processing by implementing a fixed set of operations and feeding the results directly into a classifier [48]. For quantitative measurement, however, this is not indicated. In this case, the inference step would be replaced by a regression step to translate the algorithm output into the measurement of a physical value.

1.2 Current Situation in Laboratory Automation

This chapter provides an overview of established applications using computer vision in laboratories today. It further addresses three fields where a CV approach should be considered and presents the procedures used today.

Because the thesis' aim is the development of a CV framework for quality- and process control, tasks are grouped in sample analysis and quality process control applications accordingly. Data from sample analyses constitutes the outcome of an experiment, and therewith the target data of the whole automated process, whereas quality and process control applications support the process of generating such target data.

This chapter addresses the fact that the vast majority of applications of CV is part of sample analysis, while the number of quality and process control applications is comparatively low. Laboratory automation does not make use of the progress made in AOI and in digital imaging hardware evolution yet. The considered applications are summarized in Table 1.1.

1.2.1 Image-based Analytic Applications

Plate Readers analyze samples in microtiter plates by reading fluorescence, absorbance or luminescence. An example application is the measurement of antibiotic resistance of bacteria. The first application of bio-luminescence analysis using light-amplifying charge coupled device (CCD) sensors is dated in the late 1960's [36]. In 1980, McFadden filed a patent describing an apparatus for the sample visualization of fluorescence in microplates [56]. The evaluation itself was conducted manually. Kelly et al. filed a patent named *microtiter plate reader* [47] in 1987, that describes an apparatus to equally illuminate all wells of a plate for easy manual evaluation.

Since then, plate reader technology evolved continuously, with a high number of filed patents and publications, until a patent filed in 2000 describes the invention of imaging multiple samples in wells and consecutive automated evaluation [58]. The device moves a sample plate in a manner that allows a detector to read one sample at a time. The device uses a mask to cover other samples during imaging.

Andrews et al. patented a device that uses optical fibers to illuminate a respective fluid sample of a plurality of samples and a single detector with a plurality of optical fibers that detects the light emitted from the illuminated fluid sample [2]. The device has no moving parts, but still processes the samples serially. Volcker et al. use a cylindrical lens and prism array to image a plurality of samples in parallel [81].

Plate Imagers are an advancement to plate readers, they provide extended means for sample analysis by means of images instead of a single fluorescence measurement

[79].

High Content Screening (HCS) uses plate imagers to automatically evaluate samples in microplates visually and is also called *high-content microscopy.* It was introduced in 1996 by Biological Detection, Inc. (BDI) with their product *ArrayScan*^{TM} [84] [30]. Bushman et al. date the first validation of an automated high content screening (HCS) application to be in 1998 [13].

Protein Crystallography: A recent development is the use of automated image processing for *high-throughput protein crystallography.* Crystallization is detected by image processing with high precision that outperforms the classical manual assessment [4].

Agglutination Analysis: Blood agglutinates when antigen/antibody reactions happen. The resulting change of appearance and diaphaneity can be analyzed by visual inspection or turbidimetry. For instance, this is used in vaccine development [28] or for blood typing in blood banks. A patent filed in 1992 by Ohta et al. describes an apparatus for the digital imaging of agglutination in microplates [59]. The device is inflexible, since it uses mechanical transport means to move a sensor over a plate in a fixed motion pattern. Plates with different densities could not be used with such a system without modifications. Today, fully automated devices for high-throughput blood typing exist that rely on technology patented by Shen et al. [76]. [1]

1.2.2 Colony Picking

Colony picking is the most widespread application of CV for process control in laboratories. It is the process of selecting target cell colonies from a plate by a specific characteristic, e.g. color. It is used to grow large numbers of cell colonies for deoxyribonucleic acid (DNA)-sequencing where DNA-fragments are cloned in living cells. The colony picker selects surviving colonies for further cloning. The utilization of CV technology to automate and accelerate the picking process started in the late 1980's. The Manchester Biotechnology Centre, University of Manchester, played a strong role in the process, when its *automatic plaque selection and culture inoculation robot* (APSCIR) was developed. The robot is able to discriminate living (blue) and dead (white) cells by color and transfers selected colonies to an MTP [15].

[1]Blood bank devices from *Ortho-Clinical-Diagnostics*, `http://www.orthoclinical.com`

Jasiobedzki et al. [42] present the adaption of the APSCIR system for streptomyce detection in 1987. Further research addresses the application of CV methods to automate plant seedling growth [34]. Jones et al. present the addition of automated image processing to a previously manually operated colony picking robot in 1992 [44]. It uses a camera to present an image to an operator who selects valid colonies manually. This manual selection process is automated, such that no further supervision is necessary, and thus the process is fully automated.

The Human Genome Project was a main driver for the development of high-throughput colony picking robots: Due to the high demand of DNA-fragments, it was the key requirement for the application.[2]

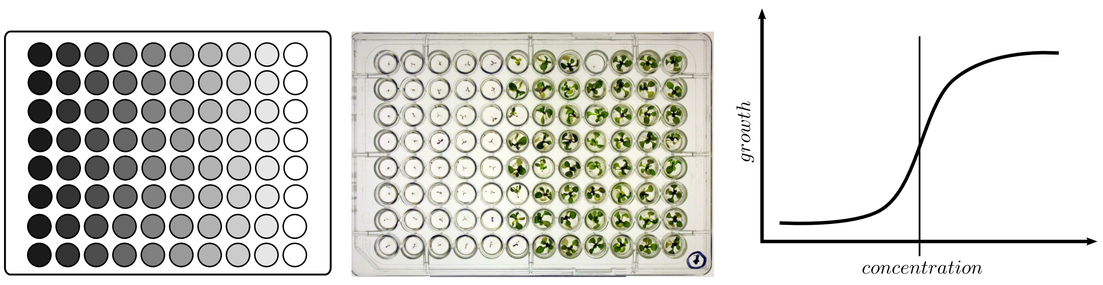
1.2.3 Plant Growth Screening

Plant growth screening addresses the (optical) inspection of plant growth. It is used to measure growth as a target information of the experiment and is conducted manually. The screening is exemplarily described for an application in a toxic resistance measurement. The plants (often *Arabidopsis thaliana*) are grown in a 96-well MTP whose wells are prepared with an increasing concentration column-by-column of a toxic substance (Figure 1.2a). The plants are seeded and stored. They are provided with equal amounts of light and nutrition media for a defined time. Afterwards their growth is measured (Figure 1.2b). The target information is the toxic concentration at which growth in inhibited (Figure 1.2c). If this information is brought into context with genetic characteristics, the gene controlling the resistance against the substance can be isolated.

Growth measurement is considered basic knowledge in plant sciences and is a very tedious task. The substitute measure for growth is weight. Prior to weighing, the plants have to be picked one by one and placed onto a paper towel. They are dried for 4 to 6 hours at 80°C and subsequently, the plants are weighed and a mean value is calculated for every column, i.e. for every concentration. Regardless of the effort it requires, this measurement technique is far from being precise: According to plant scientists, errors of up to 20% are normal. This is the worst bottleneck of such experiments and can be approached using CV.

That CV is applicable to plant growth screening was shown by Spalding [77]. Still, the

[2]Today, the Joint Genome Institute (JGI) of the Department of Energy (DoE) produces 192,000 colonies a day with four robots. See: `http://www.jgi.doe.gov/sequencing/education/how/how_5.html` (accessed December, 13, 2011)



(a) concentration **(b)** resulting plant growth **(c)** resistance of plant species

Figure 1.2: Growth as a measure for resistance

available methods are not usable to high-throughput screening, since they are designed to picture freely growing plants from the side in front of a white uncluttered background. The fact that errors of up to 20% are state of the art today and that growth screening in a similar way is already in use, makes the assumption reasonable that equal or improved results can be obtained using CV. This application is part of the research subsequent to this work.

1.2.4 Liquid Level Detection

By replacing a human operator it is possible to improve precision and decrease the number of errors, especially when boring and repetitive tasks are considered. However, the human ability to recognize a faulty outcome of an experiment step is lost. In some cases, a quality control during the experimentation, not when final results are already corrupted, makes sense. To be able to sense faults, sensory equipment is needed. In the case of low-volume liquid handling monitoring, which is presented below, CV offers effective quality control.

A patent filed in 1989 describes a mechanism for manual visual control of MTPs [52]. The device holds a plate at a predefined parallel distance to a patterned surface. The light reflected from the patterned surface is refracted through the fluid in the wells of transparent MTPs, allowing detection of fluid from distortions in the pattern or specular highlights compared to the empty wells. This method makes manual monitoring easy for the user, but it is restricted to clear, transparent microtiter plates.

Current pipettors use capacitive or air pressure-sensitive pipettes to determine a volume by detecting the fill height and calculating volume from the well geometry [66]. Drawbacks of these methods are expensive pipette tips and relatively slow measurement ($> 1min$), because every well has to be approached individually. Furthermore, those approaches will not yield usable results for volumes that apply in our case ($0,5$ - $20\mu l$ for 96-well and $0,5$ - $10\mu l$ for 384-well plates), since the fluid still forms a droplet on the bottom of the well [70]. These ranges are near the lower boundary of a pipettor operation range where precision and reproducibility usually decreases [67], hence a monitoring would be especially reasonable.

Ultrasound sensors may be used in determining volume, and also support contact-free measurement. There are sensors available that are small enough to allow column-wise well measurement for 96-well plates [19]. The sensors have to approach every column individually, hence mechanical transport of the sensing device is still required.

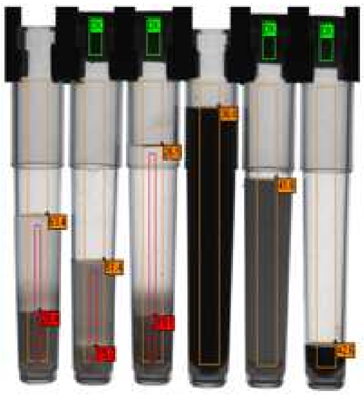


Figure 1.3: RTS Vial Auditor results

Lately, two concepts of liquid level detection (LLD) systems for quality control (QC) application were introduced. The *RTS vial auditor* uses computer vision for liquid level measurement of vials. The optical approach was chosen to enable sensing without the need to decap vials, because that would introduce a new error source, as an additional cause for cross-contamination [82]. The restriction to vials make it possible to pick one row of samples at a time and to raise them above the rack. They can be pictured from the side with an uncluttered and defined background, which eases the later image processing. The imaging from the side makes the liquid level in the vail observable (see Figure 1.3).

ARTEL's *Multichannel Verification System* (MVS) is based on a plate reader and provided dilutions, and is used to validate liquid handler precision in dedicated validation runs. The MVS is certified by the Federal Drug Administration (FDA) for the use in pharmaceutical research. It is is not able to measure sample volumes during experiments, since the dilution must be added for measurement (see Figure 1.4).

During this thesis, a liquid delivery monitoring application based on machine vision

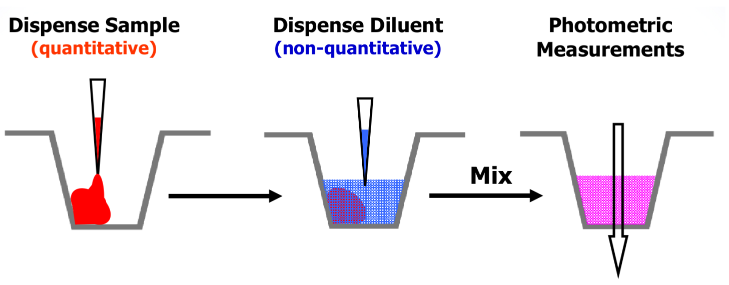


Figure 1.4: ARTEL MVS measurement procedure (from: ARTEL, Inc)

is developed. In contrast to the above mentioned methods, it is aimed at supporting microplates and lowest volumes and being usable during runtime.

1.2.5 Other Applications

Sample Preparation in Plant Sciences

Computer vision offers possibilities to automate parts of experiments that are carried out manually today. Kolukisaoglu et al. mention the field of plant sciences as being hampered by a lack of well-adapted automation technology. At worst, such a lack of high throughput laboratory equipment can constrain the level and progress of research in the field [49].

One such application of CV systems is the automation of sample preparation tasks that put high demands on laboratory staff, making them strenuous for human operators [3]. Sample preparation involves the preparation of a sample so that it can be processed by laboratory equipment, and the input of the samples into the automated experimentation system. If, for example, a leaf of a plant is to be fed into an automated process, it has to be cut into pieces that fit the labware. Due to the varying sizes and shapes of leaves, a sensor would be required to locate the right spots to cut.

Cell Culture Media Monitoring

Another example application is the control of cell culture nourishment by color detection. The culture-medium for cells changes its color during metabolization. Today, this is detected either by manual inspection or by specially designed optical sensors that measure the wavelengths of reflected and transmitted light [45]. While using optical sensing too, the current design is particularly made for that purpose and is not reusable for other applications. With respect to the low quantities typical to laboratory automation equipment, this is a drawback in terms of costs and development effort. Having a commercially available (commercial off-the-shelf, COTS) plate imaging device and a machine vision tool able to detect and classify colors would ideally reduce the development effort to the training of colors corresponding to fresh and used fluid.

Quality Control for Compound Libraries

A first report on a computer vision based QC application supporting MTPs was published in June 2011 [5]. The authors use a camera with a telecentric lens to image the plate from below in visual and infrared (IR) spectrums. Their device is able to detect empty wells and scans the compounds for precipitation and air bubbles. The authors provide a summarization of alternative methods and compare advantages and disadvantages. They state that their system mainly replaces the manual inspection as discussed in Section 1.2.4.

Table 1.1: Quality and process control tasks in laboratory automation

	Task Description	**Current Measuring Principle**
Q/P	Liquid Level Detection	pipette tips, acoustic
P	Colony Picking	computer vision
Q/P	Preparation of plant samples	manual
$Q/P/S$	Plant Growth Screening	manual weighing
P	Culture Media Monitoring	color sensor
S	Agglutination evaluation	visual/turbidemitry

Q: quality-control, P: process-control, S: Sample-analysis

1.3 Aim of this Work

As explained in Chapter 1.2, the high cost of software development is likely to be a main hindrance for the adoption of machine vision technology in laboratory automation. One way to overcome this problem and to ease software development is to establish a generic framework for machine vision applications that integrates functions common to

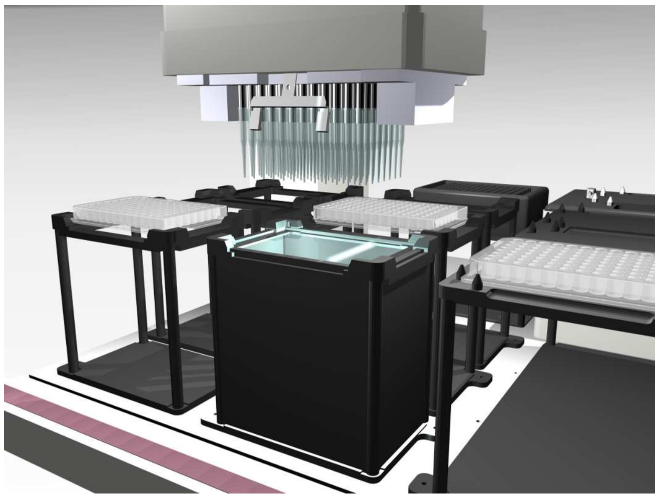


Figure 1.5: 3D model of a workstation deck equipped with a visual monitoring device and a multi-channel pipettor head

laboratory applications and offers flexible implementation of the actual algorithms. The ideal case would be a framework where the developer only has to care about the image-processing algorithm without needing to care about hardware, software, labware, user interfaces and integration into the process control. As shown in Fig. 1.6, he would design the computation run on a well, while the parallelization to and localization of all wells in the image, the labware type and the tools to setup and configure the application are readily provided, together with process and user interfaces.

To meet the requirements of high-throughput laboratories, this framework aims at imaging devices that scan a whole plate at once. Devices similar to the plate readers described in Section 1.2 were intentionally left out of this consideration. Their design is limiting the number of compatible labwares. Many readers only work with 96-well plates, often they require a specific manufacturer and type. While this is acceptable for analytic devices, such a design is too expensive and too inflexible for the use in quality and process control applications. They must be usable for different applications and compatible to experiment requirements of specific analytical devices.

This thesis defines the requirements of such a framework by taking hardware, labware and laboratory automation specific use cases into consideration. This is elaborated in the next chapter (2). According to the found requirements, concepts are designed and presented in Chapter 3. Subsequently, the developed reference implementation is covered in Chapter 4, and the last part of the thesis presents the integration of a developed AOI

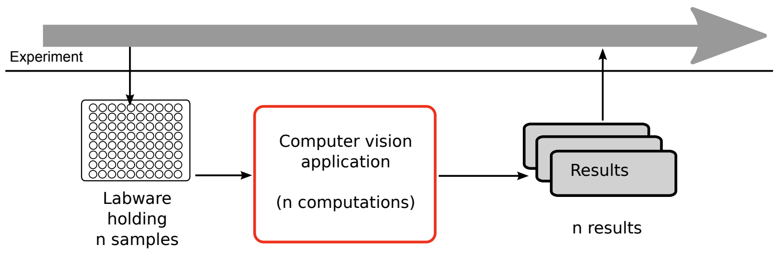


Figure 1.6: The machine vision application as a black box

application using the framework.

Chapter 2

Requirements Review

The software framework should act as a middleware between imaging devices, the process control and the user. Its main objective is to provide a generic infrastructure to developers that facilitates and accelerates development. This chapter reviews the main requirements for a machine vision framework for automated laboratories.

Today, every laboratory uses equipment that automates particular steps of an experiment. Examples of such equipment are shakers, magnetic stirrers and liquid handlers. A method would start with a first step, for example the separation of cells that stick to a flask wall using a shaker. Afterwards, a human operator transports the cell culture to a liquid handler that aspirates the medium and dispenses it into multiple vials for further processing.

Laboratory automation, as the term is used in this work[1], integrates such devices into a process consisting of multiple automated steps, often called *method*. Therewith, it replaces the steps of the workflow that were previously conducted by a human operator. The main task of laboratory automation engineering is hence the integration of heterogeneous devices into a facility while considering throughput, robustness and costs.[29] [35] [72]

2.1 Functional Integration into an Automated Laboratory

The integration is done by physical and informational means, i.e. hard- and software. The hardware is a robot that transports samples from one station to another. A software

[1]also called total laboratory automation (TLA)

controls the experiment by scheduling the experiment and commanding the transport robot and attached devices. When designing a device for optical inspection, the reachability of the plate position must be considered. Today, different robots are used within laboratories. Actuated robots usually have a vertical gripper to pick and place labware, while Cartesian robots use horizontal grippers. Models of both are shown in Figures 2.1 and 2.2.

The installed equipment in a facility defines which experiments can be conducted on that facility. Still, the exact parameters of an experiment are subject to change over time, e.g. when operators want to optimize their experiments. Furthermore, often more than one experiment runs on the same platform, e.g. day routines that need support by operators and a night routine. Consequently, a requirement on laboratory equipment is fast reconfigurability that, at least to a certain extent, should be manageable by non-technicians.

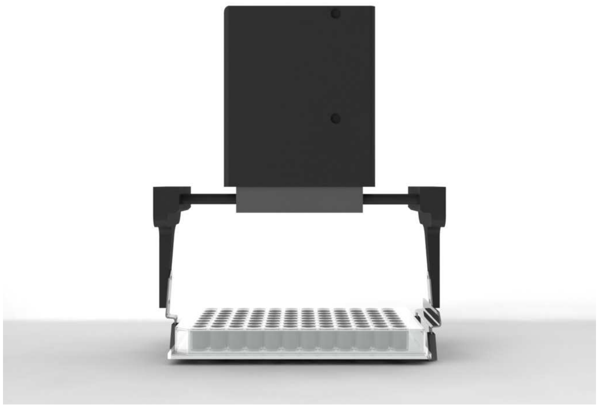


Figure 2.1: Horizontal gripper

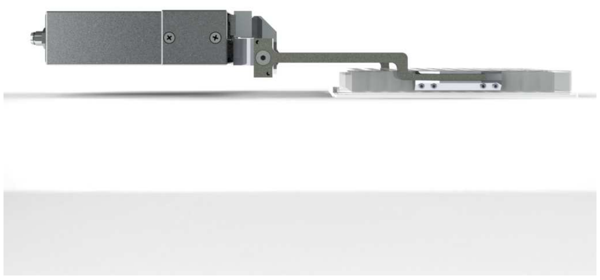


Figure 2.2: Vertical gripper

2.1.1 Classification of Use Cases

The previous section presented four different applications of CV in laboratory automation. To understand what functions and flexibility are required of the framework to support the mentioned types of applications, they are exemplarily investigated. This is done for a single sample only, although laboratories usually handle multiple samples in parallel (see section 2.4).

From the outside, the only differences between the presented applications are their input parameters and results. This is true because the framework handles a CV-application as a black box with defined inputs (images of samples, labware information, application parameters) but unknown output. The output depends on the scope of the CV application itself and must be passed on to other systems that store the data and/or react if applicable. Three use cases are introduced: *qualitative evaluation*, *quantitative evaluation* and *object detection and classification*.

Qualitative sample evaluation

In this use case, the CV system is used to answer a qualitative question about a sample depending on some numerical features x generated from the image and a threshold t (see (2.1)). One well holds a single sample, hence the result of a plate evaluation encompasses n_{Wells} binary values. Applications that belong in this group are qualitative drop detection (*"Is there a drop in this well?"*) and others, such as agglutination evaluation (*"Is the sample in this well agglutinated?"*). It is required to pass a key (name of the measurement) along with the value.

$$r(x) = \begin{cases} 1 & if \quad x \geq t \\ 0 & if \quad x < t \end{cases} \tag{2.1}$$

Quantitative sample evaluation

The quantitative evaluation results in n_{Wells} values, which can either be discrete or continuous and are in a application specific range. Examples are quantitative drop volume determination (range: $0\mu l - 10\mu l$) and plant growth screening (range $0gr - 20gr$). As for the qualitative measurement, it is required to pass a key and also a measurement unit along with the result.

$$r = f(x) \tag{2.2}$$

Object detection and classification

In contrast to the first two cases, *object detection and classification* produces complex multi-valued results for each sample. The number of results depends on the number of found objects n_{Obj} and the number of values $n_{r,Obj}$ for each object. Assuming that the location of an object, defined by two values $p = (x, y)$ is of interest, $n_{r,Obj}$ would be two, and the number of results per well is $n_{r,Well} = n_{r,Obj} \cdot n_{Obj}$. The number of results per plate is

$$n_{r,MTP} = \sum_{i}^{n_{Wells}} n_{r,i}. \tag{2.3}$$

This means that a single sample has to return $n_{r,Obj} \cdot n_{Obj}$ values per well, and a dynamically sized result container is required.

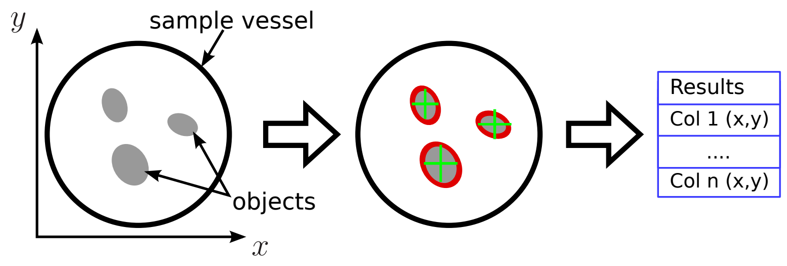


Figure 2.3: Object detection scheme

2.1.2 Requirements on computerized systems

Is a computerized system used in the pharmaceutical industry, regulations stated in *CFR21, Chapter 11*, respectively *EU GMP, Annex 11* and *EU GMP, Chapter 4*, apply. One key requirement is *data integrity*. In a comment on the above mentioned regulations, the author McDowall states: "In summary, these sections are looking for checks for correct and secure entry (both manually entered and automatically captured data) and the subsequent data processing to minimize the risks of a wrong decision based on wrong results." [55].

The tracking of system settings and states during a process of data generation, a so-called *audit trail*, is a key part of these checks, which should be considered during software development. It is not the aim of this work to deliver a product conforming mentioned regulations, still, they should be considered during the concept phase to facilitate a potential prospective certification and use in regulated environments.

2.1.3 Summary

The previous sections presented four different applications of CV in laboratory automation and their classification in use cases. Further, the need of traceable system states were mentioned. The resulting requirements are:

- Input to any CV application will be a labware with a number of samples and application dependent parameters
- To cover all use cases, the system must be able to handle three different types of results displayed in Table 2.1
- Audit trails should be considered in the concept

Table 2.1: Summary of output characteristics for the three use cases

Use Case	**Result value**	**Results per sample**	**Results per plate**
Qualitative Evaluation	binary	1	n_{Wells}
Quantitative Evaluation	floating	1	n_{Wells}
Obj. Detection and Classification	all	$n_{res,i}$	$n_{r,MTP}$

2.2 Operational Requirements

During operation, the program flow is linear and static. A command from the process control system (PCS) is received and interpreted, subsequently executed and results are returned. The user interfaces described above, however, require the library to support the nonlinear use, which is associated with the human approach to use software. Apart from the increased effort that is required to make the software robust to the different ways in which humans use software, extended application program interface (API) functionality is required for the use with the administrator graphical user interface (GUI), which implements calibration and setup functionality.

2.2.1 Operation

The software flow has a linear character in *operation* mode. The CV system acts like a sensor for the automation system that returns measurement data on a read command.

Fig. 2.4 shows the typical program flow. At startup, the CV module is initialized. During the course of the experiment, it listens for commands. When the process control system requests the evaluation of a plate, the module starts operation. Herewith, parameters such as the plate type are provided. The system acts accordingly, eventually sets parameters according to the command and starts image acquisition. As soon as the image is present in the system memory, the mapping system partitions the image into well sub-images. Then, the evaluation starts for every well and a result is computed for every well. The application passes the results to the PCS. Afterwards it sets itself to sleep, waiting for the next plate.

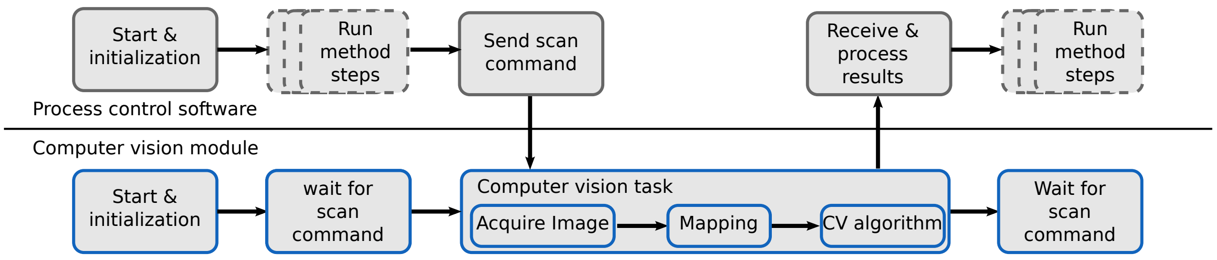


Figure 2.4: Operation flow scheme

2.2.2 Hardware Calibration

Every time the hardware or its configuration is changed, a calibration has to be run. The calibration of an optical system is the process to find optimal parameters for its projection-mapping model. As mentioned in Section 2.5, this is required for flatbed scanners, cameras and multi-camera devices.

The calibration is implemented in the GUI, hence the framework must support the iterative manner in which users set up a system. The user selects, for example, hardware settings and acquires a test image. The test image might not be optimal, hence parameters are changed and different images are taken and compared subsequently. Finally, a set of images is chosen and passed to the application. Hardware setup and calibration, but also application settings, are managed from within a GUI, leading to the requirement that the framework must be able to cope with such program flows as shown in Fig. 2.5.

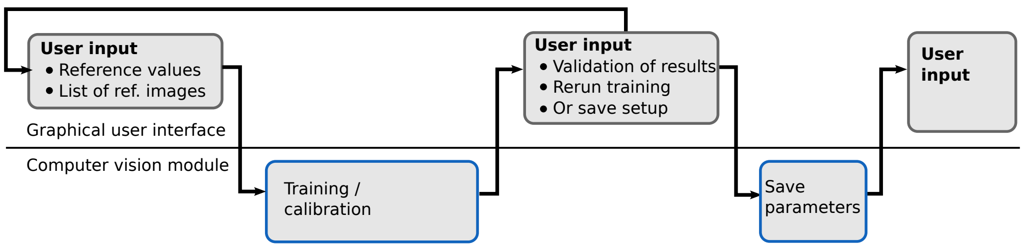


Figure 2.5: Setup flow scheme for hardware and software calibration

2.2.3 Software Calibration (Training)

Software calibration (algorithm training) is the process to find optimal parameters of the vision algorithms and for the classifiers of the current task. Since applications will use different CV algorithms and different classifiers, generic optimization routines are required. The routine should be applicable to the majority of computer vision algorithms, and following characteristics must be considered:

- number and type of parameters (input)
- continuous or discrete range / stepsize (input)
- descriptive parameters (e.g. U or V bottom) (input)
- nonlinear or unknown behavior of the algorithm.
- number and type of result values per well (output)
- number of wells (output)

Software calibration is application-specific. With respect to the use cases described in Section 2.1.1, the training routine must be able to handle the three different result types, 1. binary classification / qualitative measurement (n=2 bin classification), 2. quantitative measurement, 3. n-bin classification (prepared, but not included in this thesis).

Neither continuous differentiability nor linearity can be assumed, hence a closed-form solution is unlikely to exist [78]. A search routine might therefore terminate unsuccessfully and the framework must support a nonlinear program flow during the parameter search as shown in Fig. 2.5. The typical flow of a training operation starts with the selection of training samples. If a binary detector is trained, these will be images of plates with positive and negative samples for the given situation. If a multi-class classifier or a quantitative measurement is trained, samples of different sizes or volumes are needed as samples.

Nonlinear optimization algorithms are iterative search routines that neither guarantee successful execution nor the localization of the global optimum. As a consequence, grid searches can be regarded as a good alternative and are selected to be implemented. Theoretically, a grid search will find the optimum of a function, when the stepsizes are chosen to be indefinitely small. Thus, a grid search can be favorable especially for a smaller number of parameters or discrete parameters.

2.3 User interface

Unlike in industrial automation, the devices have to be configurable by non-technicians for their specific purpose without in-depth technical knowledge. Two user interfaces are required of which both should be easily adaptable to the current application. The first is the MMI, which laboratory devices commonly offer to manually simulate or reproduce experiment steps. It exposes the functionality and limited settings that are provided to the process control system.

The second interface is the administrator interface. It extends the range of the provided functionality by calibration and training routines and is used to edit the configuration files of the CV module. The goal remains to provide a user-friendly setup routine that does not need expert knowledge. To restrict its use to trained persons, this interface should be password protected and only be used by a trained person.

2.4 Labware

The term *labware* describes consumables that are used in laboratories to store, hold and transport samples. With the increase of throughput over the years, the microtiter plate (MTP) became popular. It holds multiple samples, allowing parallel processing and analysis. They are held in cavities called *wells*, which are arranged in a grid of rectangular shape (see Fig. 2.6). To improve interoperability and compatibility between devices of different equipment manufacturers, microtiter plates were standardized by the Society for Biological Screening (SBS) [73] [74].

The standard SBS microtiter plate labware holds multiple arrays of wells laid out in a row-column fashion. The framework must be compatible to MTPs and to custom labware or slides[2]. Microplates are available in a wide variety of materials and shapes and have 1 to 1536 wells.

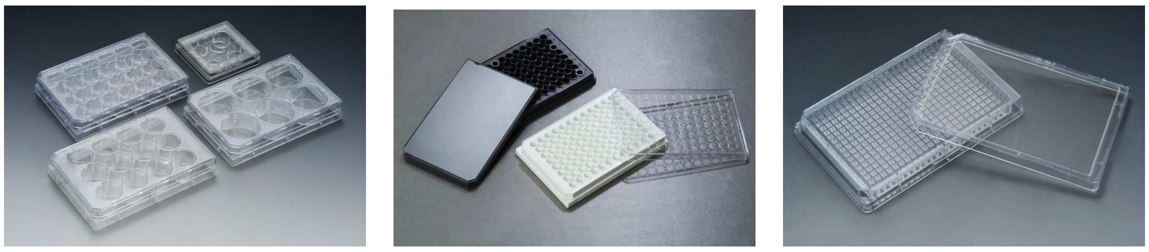

(a) 4 to 24-well microtiter plates **(b)** 96-well microtiter plate **(c)** 384-well microtiter plate

Figure 2.6: Labware: MTP types (source: `http://www.polyplastics.com`)

The samples carried on the labware are the actual subject of investigation by the CV application. Hence, the labware type affects the operation of the system in two ways. Firstly, its dimensions and well layout are needed for the mapping of wells to image regions and secondly, its material type and other geometrical information, such as the well bottom shape, can be important parameters to CV algorithms. To provide the necessary information to mapping and algorithms, the library must be able to handle labware information. The labware type should be considered as a processing parameter that is passed to the computer vision system for every plate transported to the imaging position. The system should be able to handle different labware types during operation. The resulting requirements are:

- Support of MTPs
- Provisional support of labware with unknown number and alignment of samples
- Support of changing labware types during operation

2.5 Imaging Hardware Support

Hardware support in the context of this work means support on two levels: The hardware interface layer must be able to communicate with different driver APIs, with the key part being the hardware abstraction in a way that makes higher level functions device independent. This is a main requirement for hardware integration frameworks

[2]For the sake of clarity, it will be referred to asMTP only from now on

[71] [22], and many examples, such as the Linux hardware abstraction layer (HAL) `http://www.freedesktop.org/wiki/Software/hal`, exist.

Secondly, in order to make CV algorithms device independent, they must be provided with their input data in a uniform way. Hence all optical and geometrical calculations that depend on the optical system of a device must be taken care of. For example, as seen in the previous Chapter 2.4, the labware usually holds samples in defined positions. Their positions on the image depend upon the labware and the optical system. The second part of the hardware abstraction layer is required to map the sample positions on the image to pass them on for processing.

Optical inspection devices used in laboratories consist of a digital imaging sensor, an optical system and one or more light sources. Furthermore, all devices comprise a labware position, sometimes called automation labware position (ALP), that holds the labware in a defined position. The placement of the ALP relative to the imaging device defines the relevant perspective of the images. Additionally, the position must be accessible by the transport robot's gripper. Hence, the following constraints must be considered when choosing or designing an optical inspection device:

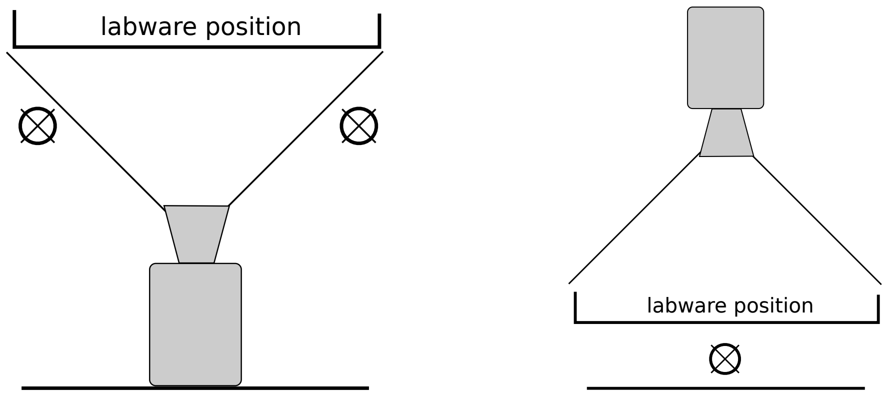
- **Optical requirements**
  - Lighting: Transmitted / reflected, spot/evenly distributed
  - Resolution
  - Perspective
  - Color channels
- **Installation requirements**
  - Installation height and space
  - Robot deck limitations
  - Robot movement to/from/over the device

Depending on these characteristics, devices are differently suitable for certain applications and a generic framework must therefore support multiple devices. Those considered in this work are described in the following. A market overview identifies four main types of imaging hardware used in laboratories listed in Table 2.2.

Table 2.2: Hardware setups used in laboratory automation

Type	Lighting	Placement	Use
camera	reflected	below	barcode reading
camera	transmitted	below	compound library QC
camera	transmitted	above	light tables, colony pickers
scanner	reflected	below	barcode reading

Reflected lighting is used when transmitted light is not applicable, e.g. when the object is not transparent or not uniformly illuminable. Hence, the setup from Figure 2.7a is often used for barcode readers. The placement of the sensor below the labware position is advantageous for integration, because it keeps the robot's path unobstructed.

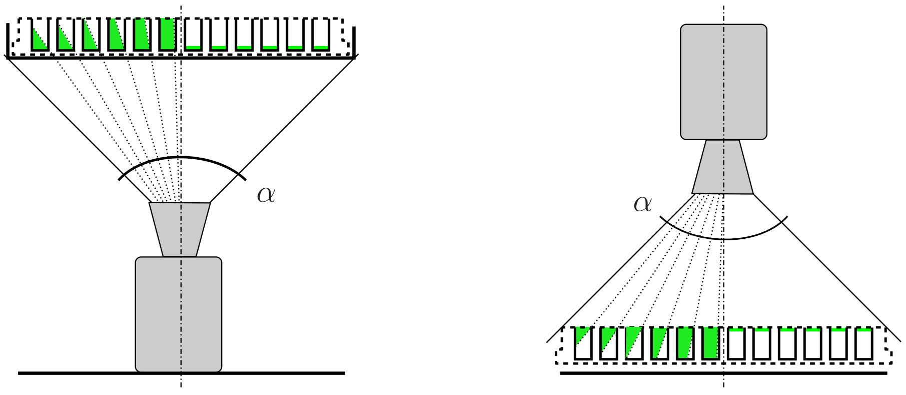


(a) reflected light **(b)** transmitted light

Figure 2.7: Two possible device setups

Transmitted light is used for transparent or translucent objects and is found in light tables but also in analytic devices such as microscopes and plate readers. Colony picking systems also use transmitted light for scene illumination. In all cases, the obstruction of the transport path has to be considered (see Figure 2.7b), since either the sensor or the light source is placed over the labware position and vertical grippers would require a conveyor from a reachable position to the imaging position (see Figures 2.1 and 2.2).

The optimal perspective depends on the application and must be considered. The primary factor is the visibility of the feature of interest. When imaging labware, the perspective change for the different samples has to be investigated further. It should be worked towards an equal perspective to every sample, because otherwise the measurement



(a) Imaging from below **(b)** Imaging from above

Figure 2.8: Perspectives for parallel imaging of multiple regions of interest

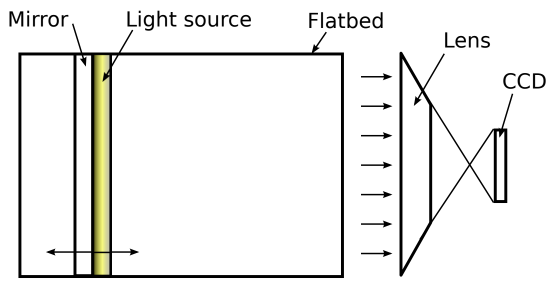
can introduce a location-dependent systematic error. This can be fulfilled by using a flatbed scanner or with telecentric lenses for camera systems. When camera systems are chosen, they are either mounted directly above or below the plate, so that the perspective symmetrically changes from the plate center towards the edges. Regarding the camera as a linear optical system, it can be shown that imaging from below the plate covers all well floors, while on the other hand, the space at the top of a well is observable from above the plate (see Figure 2.8). The lighting situation must be considered in parallel with the determination of the perspective. Available flatbed scanners, for example, do not provide transmitted lighting, although such a setup is imaginable if custom made scanners could be manufactured for an application.

2.5.1 Flatbed Scanners

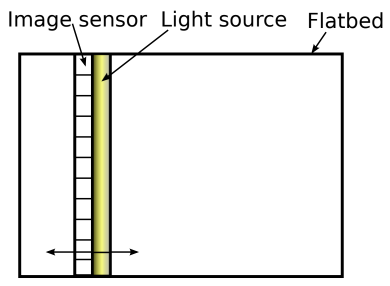
Flatbed scanners based on CCD sensors use a sensor similar to a camera, but with one-dimensional resolution, which is moved over the scan area by a mechanical transport. Such devices are similar to typical office and consumer devices, except for their small size. CCD scanners use a lens to cover the imaging area orthogonal to the drive axis, and mirrors to pass the light to the object. They use a strong light source and have a comparably large depth-of-field. Since CCD scanners use a lens in one axis, one dimensional lens effects occur. The linearity of the drive axis depends on the stepper motor precision and the timing of the sensor readings. Figure 2.9a shows a scheme of the optical system.

Scanners based on contact image sensors are also used in laboratories. Due to their design, contact image sensor (CIS) sensors allow the building of small-sized devices; they are used in facsimiles and small handheld scanners. The CIS moves near to the object plane and covers the whole width of the imaging area [89], [85]. The perspective of every point on the object is equal to all other points, although small microlenses have to be used to scan a continuous area. For applications considered in this thesis, these lenses do not have a measurable impact on the image, since their distortions are projected on a single pixel.

During image acquisition, a stepper motor moves either the sensor (CIS) or a mirror (CCD) along one axis of the image while a line scan sensor reads one line per step. The image resolution depends on the steps of the stepper motor and the number of pixels the sensor reads orthogonal to the drive axis. In laboratories today, flatbed scanners are mainly used for barcode reading.



(a) CCD



(b) CIS

Figure 2.9: Flatbed scanner schematics

2.5.2 Camera Systems

Camera systems are built up of a digital camera, an objective and light sources. They furthermore comprise a labware position that holds the labware in a defined position relative to the camera. A plate shuttle can be required to load/unload plates.

2.5.3 Camera Grids

Camera grids are devices that comprise multiple sensors arranged in a grid with their z-axes being parallel. Devices that use multiple sensors are common in laboratory automation. Therewith, manufacturers are able to keep the installation height of their devices minimal, by keeping the focal length fixed (i.e., for typical cameras, keeping the lens angle fixed) or to provide a better perspective for wells located at the border of the plate by increasing focal length (reducing lens angle). See Figures 2.10 and 2.8 for illustration.

Since the cost-performance ratio of electronic devices improved steadily during the last years, digital cameras do not add much to the price of a device. On the software side, multiple sensors require specially adapted projection mapping. Proprietary camera grid devices support a small set of plates and it is likely that they rely on simple fixed masks. This can only be assumed because proprietary systems are closed source. To the author's knowledge, no public documentation or publications mention such devices nor methods to register camera positions and projection mapping. The resulting requirements are:

- Wrapping driver APIs for unified communication to hardware.
- Providing device-independent *labware→image* mapping (see next section).
- Supporting camera based devices, but also CIS and CCD flatbed scanners.
- Supporting multiple sensors and data from a device.

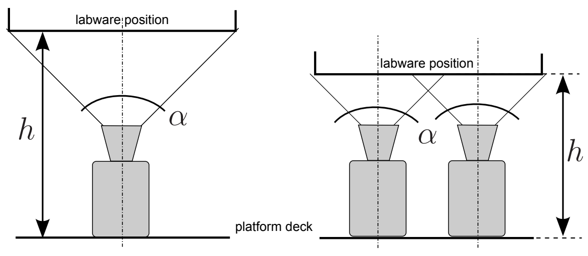


Figure 2.10: Comparison of installation height h for single- and dual-camera setup with a lens angle α.

2.6 Projection Mapping

A key requirement is the implementation of device-specific projection mapping capabilities. The section picks up the aforementioned second part of hardware abstraction, the geometrical *labware→image* mapping. The task is to map a region on a labware, e.g. a well, to a region on the image. The typical question to be answered is:

Which region on the image corresponds to well A-1 on the plate of actual type?

In order to answer this question, the projection of the three-dimensional world onto the image-plane has to be modeled. When imaging a scene, a light-sensitive sensor is exposed to light rays, which incide controlled by lenses and mirrors. Such optical systems can be described mathematically as using the laws of linear optics. Still, every part of an optical system introduces errors, some of which are nonlinear and therewith need special modeling. Main sources of errors are lenses, sensors and their mounting. Lenses introduce nonlinear rotational distortions and the mounting of a sensor relative to the optical axis can cause tangential distortions. Figure 2.11 shows the comparison of perspective of a camera and two flatbed scanner systems. The perspective differences and the so-called barrel distortion of the camera lens are clearly visible. The unequal partition of image sensors leads to pixels that are not square but rectangular.

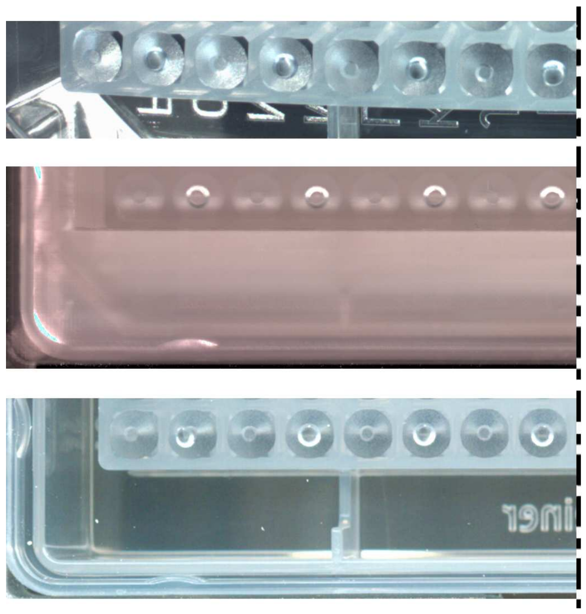


Figure 2.11: Comparison of distortions (top to bottom: Camera, CIS-scanner, CCD-scanner)

In the case of flatbed-scanners, the motor that drives the sensor, or respectively the mirror, over the image plane has placement tolerances and timing tolerances, which lead to non-equal spacing of the pixels in the drive axis of flatbed-scanners. The axis orthogonal to the drive axis, however, is affected by nonlinear lens distortions (CCD) similar to camera systems. In the special case of CIS flatbed scanners, multiple lens distortions occur orthogonal to the drive axis but can be neglected.

To ensure sufficient mapping precision, those errors have to be corrected by calibrating the system. While comprehensive documentation for camera calibration exists [12] [18] [11], only one research paper was found that deals with flatbed scanner calibration [46]. For the application in automation, such calibration procedures should be straightforward and must not require special metronomic equipment as used by the authors in [46].

Since the framework shall support hardware from external manufacturers, the calibration procedure cannot rely on any prior knowledge of device parameters. Extrinsic parameters describing the rotation and translation of the camera(s) or the scan area relative to the labware position are required. These parameters can be found by registration, a step subsequently conducted after calibration. The resulting requirements are:

- Calibration of device-specific nonlinear errors for all devices.
- Projection-mapping model incorporating calibration parameters.
- Projection mapping for all device types.
- Exposing the relevant image information to the downstream algorithm independently from the device type.

2.7 Computational Requirements

Machine vision algorithms are computationally intensive operations and their complexity increases linearly or, with higher orders, with the number of pixels of an image

In laboratory automation a system must be able to handle multiple samples at once. With a sufficient resolution for every well, this leads to rather large images. The scanners mentioned in Section 2.5, for example, provide ca. $100 \times 100px$ per well for a 384-well plate, which results in a $3.84Mpx$ image altogether. The *ABgene* device uses two cameras, with $1600 \times 1200px$ each, which too results in a $3.8Mpx$ image. As a comparison, *high definition* (1080p) video delivers $2.04Mpx$ per frame.

Computers used to control laboratory automation facilities are usually based on standard workstation PCs that offer sufficient processing power for most tasks, such as communication, task scheduling, logging and data storage. A complex signal processing application is likely to put stress on such a machine and can eventually interfere and interrupt other processes.

Therefore, it is important to optimize time efficiency of the CV system and to manage available processing power, depending on the application or task requirements. The multi-core processing units that are widespread in use in personal computers (PCs) today can help to achieve that. If fast execution is required, the computation should be distributed over the maximum number of cores, while cores should be excluded, depending on priority settings, when other operations run in parallel.

Optimum performance and a stable system is called for during operation, while training procedures, such as grid search, can be very computationally intensive, and should be accelerated as much as possible. The resulting requirements are:

- Load distribution of operation and training.
- Compatibility to different workstations that are used as control computers.
- Ability to exclude cores that are used by other processes.

Chapter 3

Conceptual Design

The requirements elaborated in the previous chapter entail a framework that wraps a computer vision algorithm for the use in laboratory automation systems. Its scope is to provide the application developer with the necessary range of functions that are not application-specific but needed by any laboratory CV application. Its key features include support of data parallelism, hardware support and abstraction, support for changing labware, and device setup routines.

The relay of generic parameter sets, from the process control system to the input side of the CV algorithm, is one task. On the output side, it has to pass the results of the CV algorithm to the process control system. In other words, the framework acts as a layer between the imaging hardware and the process control system and embeds the CV algorithm that is to run on images of the experimentation samples. It covers the process control system and the CV algorithm from the complexity of the imaging hardware and it covers the framework's hardware interfaces and abstraction modules from the application-specific inputs and outputs.

Elaborated further, a software architecture is developed that supports this key requirement but also takes into account the different use cases mentioned in Section 2.2 and the integration into a graphical user interface. This chapter presents the conceptual considerations of the framework design and algorithmic principles of the distinct parts.

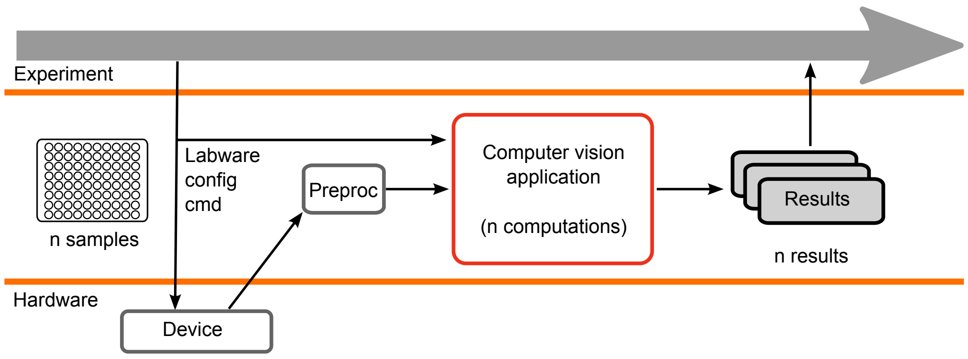


Figure 3.1: Separation of application and hardware layers

3.1 Automated Optical Inspection Tasks in Laboratory Automation

3.1.1 Task Implementation

The task itself acts on an image of an experimentation sample; it is defined for a single well by the application developer. It will have the image and additional algorithm parameters as inputs, and one or more numerical values as output (see Section 2.1.1).

Neither the algorithm parameters nor the type or number of output parameters are known beforehand, so dynamically sized structures are used. On both sides, *key value* maps with alphanumeric keys are used to implement a variadic function for arbitrary input types. By handling inputs by using keys, human readability is preserved. The parameter value does not only contain a single numerical value, but also a unit, a range and a stepsize, in which it can be changed. The latter two will be considered later in this work (in Section 3.8 considering the automated parameter search). Both input and output parameters are implemented using respective data structures.

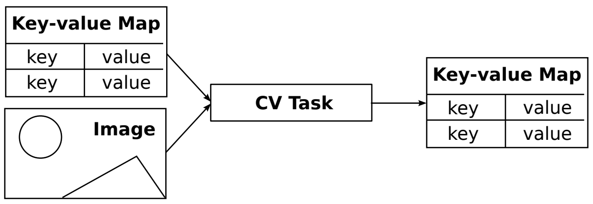


Figure 3.2: inputs and outputs of a CV task

It is technically required to connect the actual sample, identified by a plate identification plus well location (“*A-1* “), with the algorithm result to correctly reference the data later. This can be done by storing a value in an array at a position defined by the well location. If this array is handled correctly during the program flow, the value will still belong to the right well, but if any program part is erroneous, e.g. interchanges rows and columns by mistake, the reference would be wrong. Furthermore, the used configuration is of interest when evaluating results, e.g. to evaluate system performance. Significant programming effort is required to ensure correct references of input and output data. By referencing a well position (plate ID and location) with its image, and referencing this image and the input parameters to the output data, such errors can be systematically avoided.

As mentioned in Section 2.1, traceable settings and sample data can be a regulatory requirement that must be considered and which is ensured by the implementation of references throughout the application. Therewith, all necessary information can be linked to every result.

To make *audit trails* possible, the settings (input data) have to be accessed and change controlled by means that comply with regulations. This can be regarded as a main principle to follow regulations [55] [80] [24] [25], and builds a basis to fulfill the requirement of *data integrity* on a computerized system in pharmaceutical processes. Such a user management is not part of this work, but necessary prerequisites are provided by the implemented concept of references, displayed in detail in Table 3.1.

Table 3.1: Information references of output data

image references to
well location
plate ID
image file name or ID
sensor ID

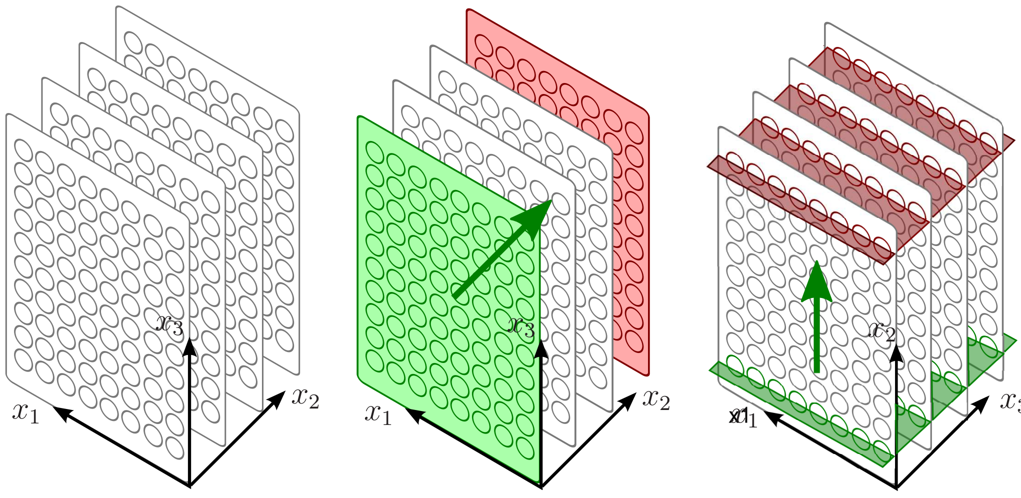
settings reference to
date of change
author

output references to
settings
image
date

3.1.2 The Result Space

While the CV task is implemented for a single well, the system itself is required to handle multiple wells on a plate, and possibly multiple plates, during parameter search or system testing. Results are generally displayed per MTP, but this is not the only supported way. If we consider multiple plates, a sample can be identified by its plate and by its position on this plate (row, column). Any plate furthermore corresponds to a point in time when the plate was processed.

Elaborated further, a well is localized by the three coordinates *row*, *column* and *plate* and in the following we consider this space for well results. Figure 3.3 shows a simple visualization of the result-space X^3, where x_2 can be regarded as the time axis.



(a) Result space **(b)** Evaluation vs x_2 **(c)** Evaluation vs x_1

Figure 3.3: The three-dimensional result space X and two evaluation paths

This approach of viewing a set of samples, their images and results is advantageous for evaluation of influences of either well coordinates or time. By considering subsets of the measurement results separately, it is possible to evaluate different influences. By considering regions on a plate separately, the robustness against changing perspective (e.g. *the first column of ten plates*) or the robustness against lighting variation over time (e.g. *the first column vs. time*) can be evaluated.

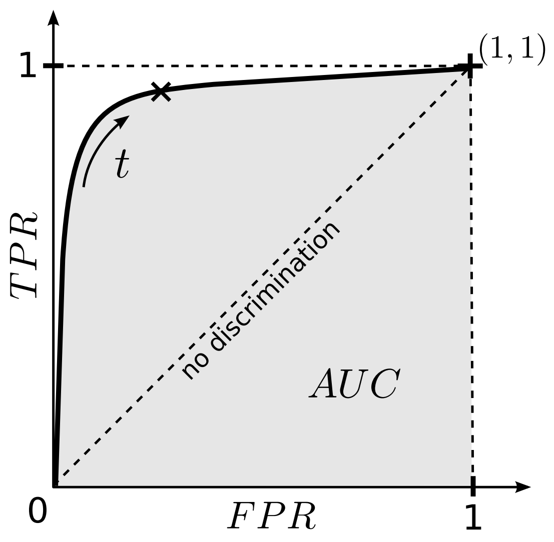
3.1.3 Performance Evaluation

The evaluation of detector performance is an important part during development and validation of any computer vision software, but also during system configuration. As pointed out previously (Section 2.3), setup and configuration are accomplished with help of the administrator user interface, and hence evaluation means are integrated into the interface. On the other hand, detector performance measures can be part of a target function for automated parameter search, which will be discussed later (Section 3.8).

The performance module is defined to have measurements and references as inputs. Therewith it picks up the results from the CV task together with its references.

Binary detectors

For binary detectors, an evaluation class was developed that provides receiver operating characteristics (ROCs) and associated performance measures [26]. The ROC displays the true positive rate (TPR) vs. the false positive rate (FPR) parametrized by a detector threshold. Figure 3.4 displays a schematic diagram.



Name	Symbol	Definition
true positive rate	TPR	$\frac{TP}{P}$
false positive rate	FPR	$\frac{FP}{N}$
area under curve	AUC	$\int_0^1 TdF$
precision	P	$\frac{TP}{TP+FP}$
accuracy	A	$\frac{TP+TN}{P+N}$
F-measure	$F1S$	$\frac{2}{\frac{1}{precision}+\frac{1}{recall}}$

Figure 3.4: ROC and related performance measures

A perfect detector lies at $(1, 0)$ or $(0, 1)$ as its inverse. The worst outcome is *not better than random guess*, which is located on the 45 deg *"no discrimination"* line. A threshold t parametrizes the curve. Many performance measures can be derived from this diagram, those further used in this thesis are listed with Figure 3.4. Area under curve (AUC) is a threshold-independent measure and can be used to describe the overall quality of an algorithm. Precision, accuracy and F-measure apply for any point on the curve individually. [26] [31]

Quantitative measurements

Statistical evaluation of quantitative measurements is covered by providing statistical properties of sets of values such as mean, minimum, maximum and standard deviation (σ) or coefficient of variation (c_v). By evaluating the result space in different manners, it is possible to calculate well, row/column and plate statistics and compare them in a typical graph where i, the index on the x-axis, denotes the axis in the result space and r the results of the set on the normal plane.

Complex tasks

This should be divided into values that can be compared with quantitative measurements or qualitative classification. In the example of colony picking that was used as an example previously, the detection of colonies would be evaluated qualitatively, whereas reference measurements of colony size or position would be investigated quantitatively.

3.2 Hardware Abstraction Layer

To provide full hardware compatibility, the framework uses an HAL with two sub-layers shown in Figure 3.5. Relationships in the interface abstraction and the perspective mapping layers are shown in this Figure. The image is received from the device by the lower abstraction layer. Concepts from Section 3.2.1 are implemented within this part. It is connected to the projection model of the respective sensor, where Sections 3.2.2, 3.4 and 3.5 will be implemented. All sensors are then connected to the sensor registration (Section 3.6).

The *scene* connects the sensors and the labware together to manage the well mapping. It outputs a map containing the locations of all wells that can be used to generate a set of well sub-images, which can then be passed to the actual CV task. At this point, the serial part of the program ends and the wells can be processed in parallel.

3.2.1 Interface Abstraction Layer

Interface abstraction is the first hardware abstraction layer. It implements the functional abstraction of all required operations. The functions that need to be implemented include a startup routine that loads device specific parameters, which were passed beforehand. Then an *image acquisition* command is required for operation.

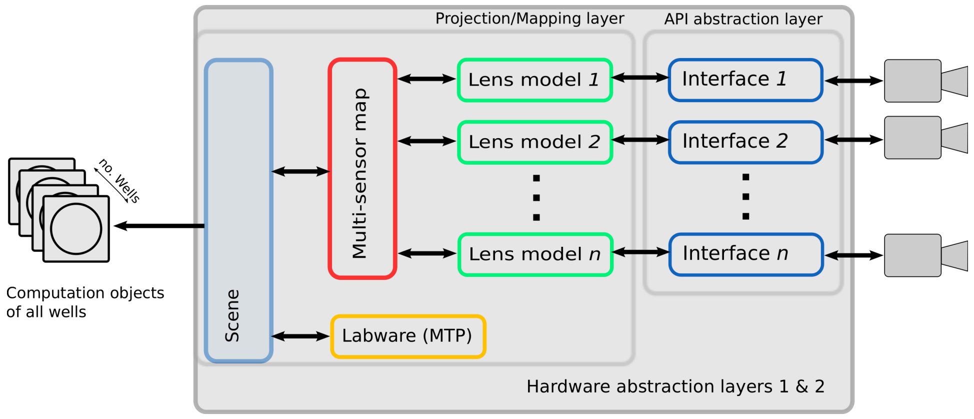


Figure 3.5: Hardware abstraction and mapping layer

Further functions to set or read parameters and operators are implemented for use with manual control. The framework supports user interfaces to interactively set, compare and store parameters without reloading the system. During operation, the configuration can be considered static. It is loaded once during initialization.

The implemented functions are summarized:

- Acquire image
- Run task
- Initialization
- Open/close
- Set/get parameters

3.2.2 Projection Mapping Layer

On top of the API abstraction, the hardware abstraction for microtiter plate-based applications and different imaging device types includes that the application must be able to reference a part of the image (region of interest) of a requested single sample position by labware well coordinates. The question that is raised is known from Section 2.6: *Which region on the image corresponds to well A-1 on the plate of actual type?* It is answered by a *labware→image* transformation. Apart from this question needed to localize samples, a perspective mapping is also required when the location of an object is of interest, for example for colony picking.

To establish a flexible method to conduct transformations for arbitrary hardware and labware, six coordinate systems are introduced. The device-fixed coordinate system D (index $_d$) is located on the robot deck coordinate system R (index $_r$). Its unit is in millimeters [mm] and it can be regarded as being two-dimensional (see Fig. 3.6 and Fig. 1.5 for an impression);

Discrete well positions (x_w: A-H, y_w: 1-12) transform to points in the labware-fixed (index $_l$) coordinate system L [mm]. The fourth system I is located on the image plane (index $_i$) and its units is pixels (I [px]). x_i is the scanner drive axis or the long side of a camera device.

Since devices with multiple camera sensors exist, it might be necessary to consider multiple image- and device-fixed coordinate systems I_i and D_i[1]. Lastly, a coordinate system C (index $_c$) is introduced for the calibration object, which is located on a calibration plate, that can also be regarded as a labware and uses the same coordinate system: C is fixed on the calibration plate L. The sixth system will be introduced later for the camera model.

A transformation between two cartesian coordinate systems is described by rotation and translation, which are denoted by a rotation matrix $\underline{\underline{R}}$ and a translation vector $\underline{t}$, respectively. Indices for the transformation between two systems (R_{ds},t_{ds}) denote destination d and source s. A point $\underline{p} = [x, y, z]^T$ can be described in either coordinate system. A special case is the transformation on a plane using a two-dimensional rotation matrix $\underline{\underline{R}}(\gamma)$ and translation vector t. To support scaling of coordinate axes, a linear transformation "scaling" matrix $\underline{\underline{S}}(\underline{s})$ is furthermore introduced:

$$\underline{\underline{S}}(\underline{s}) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}, \underline{\underline{R}}(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{3.1}$$

The rotation around three axes in three dimensional space can be expressed as the

[1]in this case the correct term would be *sensor-fixed* coordinate system, but the names are kept for now

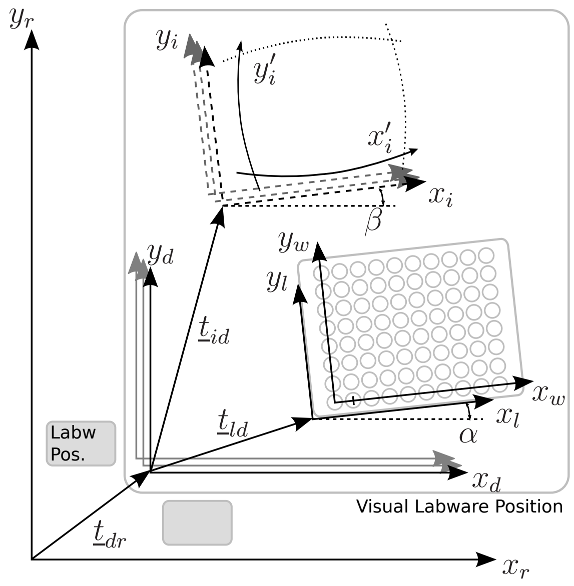


Figure 3.6: Transformations between coordinate planes

subsequent rotation around one axis by multiplication in the order of rotations[41]:

$$\underline{\underline{R}}^3 = \underline{\underline{R_\phi}}\underline{\underline{R_\theta}}\underline{\underline{R_\chi}} \tag{3.2}$$

Transformation matrices are called extrinsic in contrast to intrinsic parameters, when their respective coordinate-systems are not fixed to the imaging device (camera, scanner) itself, i.e. their position relative to the image sensor can change. As it can be seen from Figure 3.6, four main transformations must be implemented to establish the *labware→image* transformation. The robot deck coordinate system is not relevant for this work. When a location on the labware has to be approached by the robot, the relevant transformations would be done by the robot itself. The relevant ones are summarized:

1. *well→labware* , covered in Section 3.3,
2. *labware→device* , covered in Section 3.3.2,
3. *device→image* , covered in Section 3.4 respectively Section 3.5 and Section 3.6.

3.3 Labware Mapping

Labware dimensions and well positions of microtiter plates are standardized [73] [74]. Typical dimensions are listed in Table 3.2. Most labware manufacturers adhere to these standards, but exceptions exist. This means that labware mapping should not use fixed plate dimensions but provide a flexible way to integrate different well arrangements. Therewith, the design is consistent to labware configuration of other laboratory devices, which have the standard types preconfigured but also support custom designs.

Table 3.2: MTP dimensions and well positions as per [73], [74] (see Eq. 3.3 and 3.5)

Description	**Symbol**	**96 wells**	**384 wells**	**1536 wells**
overall length $[mm]$	l_{MTP}	127.76	127.76	127.76
overall width $[mm]$	w_{MTP}	85.48	85.48	85.48
margin width $[mm]$	x_{tlw}	14.38	12.13	11.005
margin height $[mm]$	y_{tlw}	11.24	8.99	7.865
well spacing $[mm]$	d_w	9.	4.5	2.25

3.3.1 Sample Map

A sample location (e.g. "*A-1*") needs to be mapped on the labware in order to provide the system with coordinates. The labware-specific coordinate systems include a discrete well coordinate system and the plate-fixed coordinate system. To map a well to a region on the plate, one has to calculate the position $\underline{p}_l$ of the well $\underline{p}_w$ on the plate. In order to do that, a well localization function $l(\underline{p}_w)$ is introduced, which transforms the well indices into a geometrical information, whereas $\underline{p}_w$ is a vector indexing a two-dimensional space.

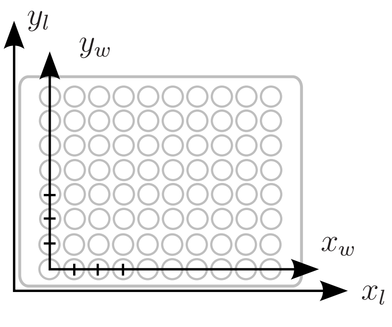


Figure 3.7: MTP-like grid pattern

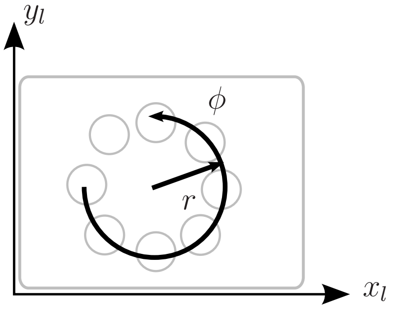


Figure 3.8: Radial pattern

For MTPs, W is a discrete Cartesian system and the point $\underline{p}_w = [x, y]^T$ on the sample grid. The point is transformed into an equally oriented real-valued coordinate system, whereas d_w is the well distance vector and as such a parameter for the scaling matrix $\underline{\underline{S}}$.

$$l(\underline{p}_w)_{MTP} = \underline{\underline{S}}(\underline{d}_w)\ \underline{p}_w \tag{3.3}$$

Other layouts are possible by altering $l(\underline{p}_w)$. For example, sample positions can be ordered shifted or in a radial pattern, which is used for slide labware for microscopic applications. The well position would be described by a two dimensional polar coordinate system $\underline{p}_w = [r, \phi]$:

$$l(\underline{p}_w)_{rad} = \underline{t_M} + \begin{bmatrix} r\cos\phi \\ r\sin\phi \end{bmatrix} \tag{3.4}$$

The function l defines the locations of sample positions relative to the zeroth well. A translation vector $\underline{t_{lw}}$ points to the zeroth well, so that the *well→labware* transformation is complete:

$$\underline{p}_l = l(\underline{p}_w) + \underline{t}_{lw}. \tag{3.5}$$

3.3.2 Labware Device Transformation

Today's automation workstations have placement tolerances $\underline{\epsilon}$ of $\pm 0.2mm$ in x_d and y_d directions. Additionally, a minimal rotational error α is introduced by the placement of the labware onto the device:

$$\underline{p_d} = \underline{\underline{R}}(\alpha)\underline{p_l} + \underline{t}_{dl} + \underline{\epsilon}, \tag{3.6}$$

whereas α and $\underline{\epsilon}$ are small compared to $\underline{p}_l$ and $\underline{t}_{dl}$. If such tolerances are not acceptable for the current application, a template matching algorithm can be used for every plate that is placed on the device. Therewith at least the translational error $\underline{\epsilon}$ can be reduced.

The *labware→device* transformation can be regarded as being a two-dimensional problem. To support imaging from above and below however, the third dimension must be considered.

3.4 Optical Flatbed-Scanner Model

3.4.1 Model

When using a CCD flatbed scanner for imaging non-flat objects, such as a microplate with V or U bottom, the perspective changes for each row in y_d-axis (see Fig. 2.11). This results from the principles of its optical system (see Figure 2.9a). The pinhole model and lens effects apply to CCD scanners in the axis normal to the drive axis [38]. Possible mapping errors can be compensated for, using a similar but simplified lens model, as for cameras. To determine if such a compensation is required, tests were run using a checkerboard.

The found distortions were similar to those found by Kangasräsiö et al. [46]. However, the deviations observed were around $0.1mm$ to $0.2mm$ and can be regarded as being negligible. The results are attached to this work in the appendix. Figure A.2 displays the positioning error orthogonal to the drive axis. The x-axis deviations are displayed in Figure A.3.

The same transformation (3.7) was used for CIS and CCD scanners during this work. If higher precision is required, for example for 1536-well plates or for colony picking applications, a calibration as done exemplarily by Kangasräsiö et al. could be implemented into the system.

CIS scanners use an image sensor that covers the whole length of the imaging area. It moves very near to the surface and has a small depth of field [1]. The absence of significant distortions leads to a linear transformation together with a correction for rotational production error (angle β) between x_d and the scanner drive axis x_i:

$$\underline{p_i} = \underline{\underline{R}}(\beta)\underline{\underline{S}}(\underline{s})\underline{p_d} + \underline{t_{id}}, \tag{3.7}$$

where $\underline{s}$ is a vector of scale factors for each axis. The required *device→image* transformation is obtained by inverting Eq. (3.7).

3.4.2 Calibration

Equation (3.7) shows that scale factors **s**, angle β and the translation t_{di} of the image corner (origin) to the device corner (origin) are required parameters for transformation. The implemented method enables the automatic determination of the combined

$$\underline{\underline{R}}(\beta)\underline{\underline{S}} = \begin{bmatrix} s_x\cos(\gamma) & -s_y\sin(\gamma) \\ s_x\sin(\gamma) & s_y\cos(\gamma) \end{bmatrix} \tag{3.8}$$

terms by least squares estimation.

The method uses the coordinate system that originates in the lower right corner of the checkerboard in the image plane, and transforms the image points into the new coordinates. The calibration is realized by using the known checkerboard corner positions $\mathbf{c}$ in object coordinates and the corresponding points in the image plane $\mathbf{c_i}$.

In order to do that, the found points have to be assigned to appropriate positions on the checkerboard by their x, y coordinates. Small rotational displacements and errors that occur during corner detection cause variations in the coordinates, such that the correct row and column position must be determined by observing intervals for each row and column. The tolerance bands are shown in Figure 3.9. The equation to be solved is

$$\underline{c}_i^* = \underline{c}_i - \underline{t}_{ci} \tag{3.9}$$

$$\underline{c} = \underline{\underline{R}}(\beta)\underline{\underline{S}}(\underline{s})\underline{c}_i^*. \tag{3.10}$$

Solving each row of (3.7) for β and $\mathbf{s}$ with a least squares approach is straightforward. If the rotational displacement is regarded insignificant ($\underline{\underline{R}}$ will be an identity matrix), (3.7) is solved for s_x and s_y. The equation is reduced to:

$$\underline{c} = \underline{\underline{S}}(\underline{s})\underline{c}_i^*. \tag{3.11}$$

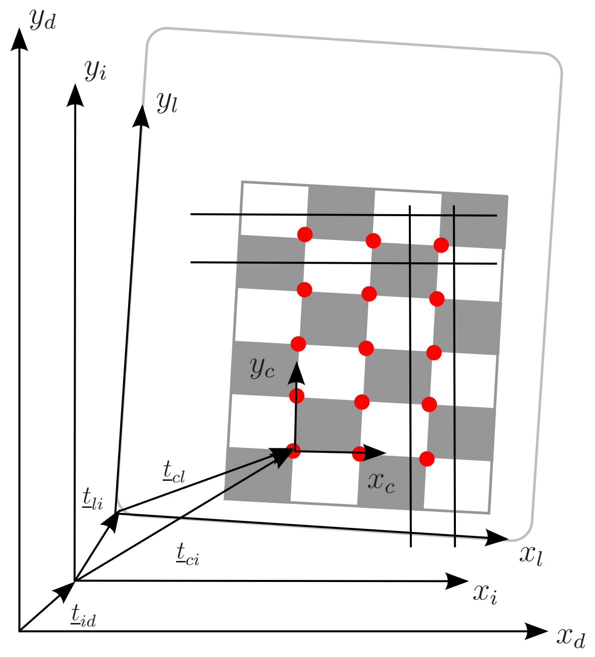


Figure 3.9: Calibration plate and coordinate systems

The translation $\underline{t}_{di}$ is found by setting the device coordinate system to the checkerboard origin $D = C$ and calculating:

$$\underline{t}_{id} = -\underline{t}_{ci} = -(\underline{t}_{li} + \underline{t}_{cl}). \tag{3.12}$$

Furthermore, $\underline{t}_{dl}$ is obtained from the checkerboard position $\underline{t}_{dl} = \underline{t}_{cl}$.

3.5 Optical Camera Model

3.5.1 Model

The camera model describes the *device→image* transformation for camera-based systems. A point relative to the device has to be mapped to a point on the image. Currently, the region of interest (ROI) is defined in device coordinates. The first step is to transform points from the device to an intermediate coordinate system ζ, which is fixed to the camera:

$$\underline{p}_{\zeta} = \underline{\underline{R}}_{\zeta d}\underline{p}_{d} + \underline{t}_{\zeta d} \tag{3.13}$$

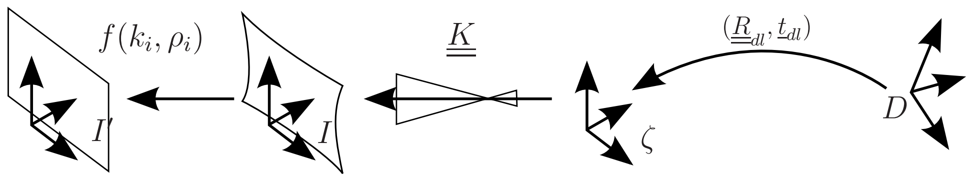

$f(k_i, \rho_i)$
I'
I
$\underline{\underline{K}}$
ζ
$(\underline{\underline{R}}_{dl}, \underline{t}_{dl})$
D

Figure 3.10: Converting from object to camera coordinate system

Subsequently the pinhole projection and distortion corrections follow. The pinhole model uses the *Intrinsic* matrix $\underline{\underline{K}}$ for linear transformation [11]:

$$\begin{aligned} \underline{p}_{i} &= \underline{\underline{K}}\underline{p}_{\zeta} \\ \begin{bmatrix} x_i \\ y_i \\ w \end{bmatrix} &= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{bmatrix} \end{aligned} \tag{3.14}$$

where $f_{x,y}$ denotes the focal length, $c_{x,y}$ the lens center and $z = w$ the distance from the lens to the labware plane. Additionally, nonlinear distortions (barrel and tangential) caused by the lens and its mounting are taken into account ($x'_i \leftrightarrow x_i$) [12]:

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} 2\rho_1 x_i y_i + \rho_2 (r^2 + 2x_i^2) \\ 2\rho_2 x_i y_i + \rho_1 (r^2 + 2y_i^2) \end{bmatrix} \tag{3.15}$$

Where k_i and ρ_i denote barrel and tangential correction coefficients respectively, and $r^2 = x_i^2 + y_i^2$. Figure 3.10 shows the steps required to map a point from the device-fixed coordinates to the image. The camera model is summarized

$$\underline{p}_i = \underline{\underline{K}}[\underline{\underline{R}}_{\zeta d} | \underline{t}_{\zeta d}] \underline{p}_d \tag{3.16}$$

$$\underline{p}'_i \xleftarrow{(k_i, \rho_i)} \underline{p}_i \text{ by eq 3.15} \tag{3.17}$$

3.5.2 Calibration

Camera calibration is needed when the internal parameters of a camera are unknown to the user, but also to find current values and deviations to the specified values. The determination of camera intrinsics, i.e. focal length and lens center, uses Zhang's method [87]. The calculation of the distortion coefficients uses Brown's method [12]. Both methods are state-of-the-art and do not have to be adapted for the current case. The methods await multiple images of a reference checkerboard as inputs.

The calibration is extended by a registration step which automatically detects the coordinate-system positions and sets their transformations as presented in Section 3.6 below.

3.6 Multi Sensor Support

3.6.1 Mapping

A system uses M cameras to image a plate with N wells, whereby the image areas of the respective cameras can but do not have to overlap as, for example, in stereoscopic setups. That implies that a registration (calibration of extrinsic parameters) method cannot accordingly rely on corresponding points. In order to avoid duplicated functionality in the library, and hence complicated code management and maintenance, multi-sensor devices are implemented as a collection of imaging devices.

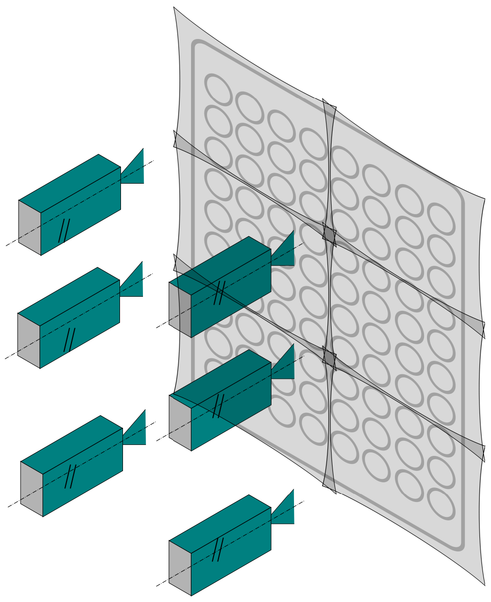


Figure 3.11: A 6-camera manifold device

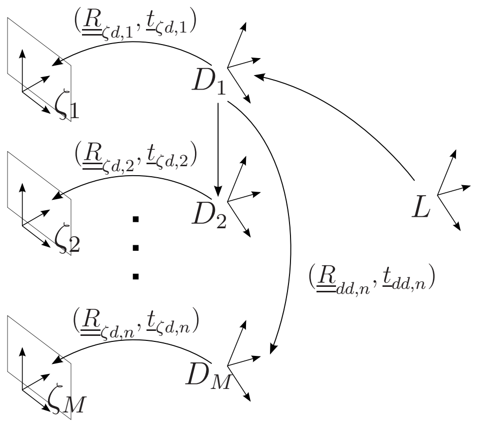

$(\underline{\underline{R}}_{\zeta d,1}, \underline{t}_{\zeta d,1})$
D_1
ζ_1
$(\underline{\underline{R}}_{\zeta d,2}, \underline{t}_{\zeta d,2})$
D_2
ζ_2
L
$(\underline{\underline{R}}_{dd,n}, \underline{t}_{dd,n})$
$(\underline{\underline{R}}_{\zeta d,n}, \underline{t}_{\zeta d,n})$
D_M
ζ_M

Figure 3.12: Multi-sensor object to camera coordinate system conversion

For the integration of multiple cameras, the result of (3.6), $\underline{p}_d$, is renamed to $\underline{p}_{d,1}$, a point in D_1, a predefined first device coordinate system. To transform a point on the device to the image of any other camera $c_j \in \mathcal{C}$ $(j = 1, \dots, M)$, an intermediate step is inserted before (3.13), as shown in Figure 3.12:

$$\underline{p}_{d,j} = \underline{\underline{R}}_{dd,j} \underline{p}_{d,1} + \underline{t}_{dd,j} \tag{3.18}$$

Eqs. (3.13), (3.14) become for camera j:

$$\underline{p}_{i,j} = \underline{\underline{K}}_j (\underline{\underline{R}}_{\zeta d,j} | \underline{t}_{\zeta d,j}) \underline{p}_{d,j} \tag{3.19}$$

and the undistortion $\underline{p}'_{i,j} \leftarrow \underline{p}_{i,j}$ analogously. Now, the well position $\underline{p}_w$ is known in M image coordinates, from which not all are valid image coordinates, i.e. within the pixel range of the image.

Which samples are covered by which camera is defined by the system's geometry and optical layout. In order to establish a mapping method, we introduce a set of cameras $\mathcal{C}$ with cardinality M comprising all cameras of a system. These cameras inspect a microplate comprising a set of wells Ω with cardinality N. Subsets of Ω are captured by each camera: $\omega_j \subseteq \Omega, j = 1, \dots, M$, where subsets can but do not have to overlap each other.

The association of a well with one or more cameras that cover it is required. As mentioned before, the set of wells Ω is distributed over $\mathcal{C}$ in subsets ω_j. We introduce a subset $c_i \subseteq \mathcal{C}, i = 1, \ldots, N$ for every well w_i. The relationship $\Omega \to \mathcal{C}, w_i \to c_i$ is obtained by two steps:

It is possible to consider for any point, if the resulting $\underline{p}'_{i,j}$ for a camera j lies within the pixel range of the image. Furthermore, if $|c_i| > 1$, it is possible to determine the nearest camera from the calculated pixel positions on the planes $I_1, \ldots, I_M$ from the distance of $\underline{p}_i$ to $\underline{c} = [c_x, c_y]^T$.

$$dist = \|\underline{p} - \underline{c}\| \tag{3.20}$$

All images where a well, defined by two points $\underline{p}_{l,1,2} \to \underline{p}_{i',1,2}$, lies within the size of the image can be used for inspection. A range check tests for which sensors $\underline{p}_{i',1,2}$ are in the range of the image size (see Listing 3.1). The distance (3.20) is calculated for its center $\underline{c}_{i'}$.

Listing 3.1: Range check

```
bool inRange[nSensors](false);
for( i=0; i<nSensors; ++i){
     if (p_i > range.begin && p_i < range.end)
       inRange[i] = true;
}
```

$$\underline{p}_{i',c} = \underline{p}_{i',1} + \frac{1}{2}(\underline{p}_{i',2} - \underline{p}_{i',1}) \tag{3.21}$$

Image regions, sorted by their distance to the image center can now be passed to the subsequent computer vision algorithms.

3.6.2 Registration

Important for the further integration of multi-sensor devices is that the camera axes are aligned in a grid and parallel to each other (see Fig. 3.11. Furthermore, they don't necessarily have a large image-region overlap, which is a fundamental difference to a stereo camera, which is usually subject of multi-camera calibration and registration.

Approaches used for stereoscopic vision calibration were not considered, since they need corresponding points imaged by both cameras [32]. For devices used in lab automation, the geometry and the size of the overlapping region is unknown, differs and is likely to be kept small by device manufacturers; hence the calibration procedure must be independent to corresponding points.

The registration of the different relative camera positions requires the use of a custom reference plate with the outside dimensions of a microtiter plate. According to the rough layout of the sensor grid, M checkerboards Cj are placed on the plate (see Figure 3.13). The absolute position of the first checkerboard on the plate $\underline{t}_{cl,1}$ and the relative positions ($\underline{\underline{R}}_{cc,j}$,$\underline{t}_{cc,j}$) of the other checkerboards are chosen during manufacturing of the registration plate (see Figure 3.14).

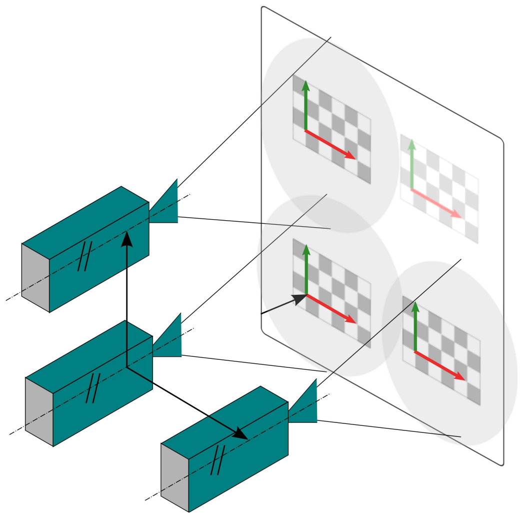


Figure 3.13: Registration of multiple sensors

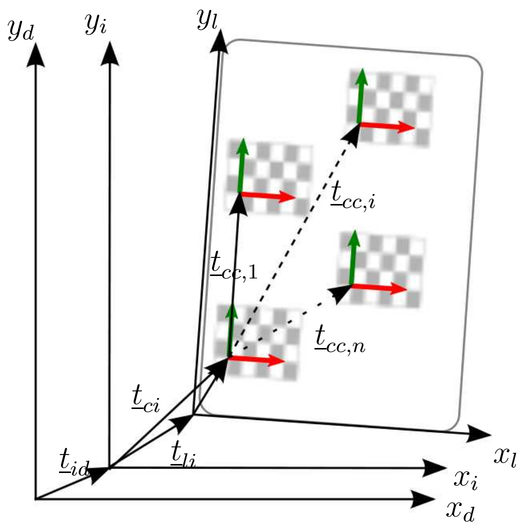


Figure 3.14: Calibration plate for registration of multiple sensors

To keep the calculation simple, rotational displacement should be zero between all checkerboards. Fig. 3.14 displays the required translation vectors from the plate origin to the first checkerboard and to the subsequent checkerboards. The developed registration approach is based upon deriving the position of one camera relative to a second via known relative rotation and translation of two reference objects. The plate is placed on the device and all cameras take an image of the plate, find their checkerboard corners and solve the homography (as implemented in OpenCV library [10]) with a-priori known object dimensions to yield $\underline{\underline{R}}_{\zeta d,j}$ and $\underline{t}_{\zeta d,j}$.

The checkerboards C_j are chosen as reference objects, since we know their relative positions. Let

$$Dj = Cj, \quad j = 1, \ldots, M, \tag{3.22}$$

when the reference plate is located on the labware position of the device. Now all transformations from figure 3.12 are defined.

The $L \rightarrow D$ transformation leads to the first device coordinate system, which is now equal to the position of the first checkerboard. A $L \rightarrow C$ translation vector is calculated for every sensor j:

$$\underline{t}_{cl,j} = \underline{t}_{cl} + \underline{t}_{cc,j} \tag{3.23}$$

Now (3.6) turns into

$$\underline{p}_{d,j} = \underline{p}_l - \underline{t}_{cl,j}, \tag{3.24}$$

with α and ϵ set to 0. The point is further transformed into camera plane coordinates by using the rotation matrix $\underline{\underline{R}}$ found by the respective single camera calibration, added to the translation vector t_{cl} found by registration.

$$p_d^* = p_l - \underline{t}_{cl,j} \tag{3.25}$$

$$p_i = \underline{\underline{K}}(\underline{\underline{R}}|\underline{t})p_d^* \tag{3.26}$$

To optimize computation time when processing power is limited or the number of cameras is high, the steps in Listing 3.1, Eq. (3.20) and (3.21), required to find the well camera assignment, are replaced by a static lookup table $f : \Omega \rightarrow \mathcal{C}, w_j \rightarrow (c_j), j = 1, \ldots, N$:

$$\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \begin{bmatrix} (c_j, dist), \ldots \\ \vdots \\ \vdots \end{bmatrix} \tag{3.27}$$

The calculation is run once for every well during initialization of the application, and the respective sets c_j are stored in N lists sorted by their distance. It is now possible to request the list of appropriate cameras for a well of interest by a table lookup.

The developed procedure does not support access to regions that are spread over more than one image, i.e. a sample must be available as a whole on at least one image.

However, it would be possible to extend the procedure using image stitching techniques. The devices covered in this thesis did not encounter such a problem for the tested plates and applications, hence it was not considered further.

3.7 Operation and Administration User Interfaces

In the *operation* use case, the results are returned to a caller, i.e. the PCS. But for setup and configuration, a GUI calls the library functions. Returning results have to be received and presented to the user in a proper way to allow interpretation of the data. As shown in Section 2.3, a user interface is an important part of a laboratory automation device.

The designed user interface has the same requirement to be application independent and must be able to handle different application types appropriately. This is made possible by defining result displays within the application part of the framework, comprising graphical and numerical presentation of data.

To form a unified interface, it is necessary to define a common output format for all result types. The result types are binary (qualitative), quantitative and complex, as described in Section 2.1.1.

The most appealing display will be the graphical display. It should give a quick and clear impression of the measurement result. From other devices that work with MTPs, workers and scientists are accustomed to work with a scheme of the plate viewed from above, where results are denoted as colors or numerical values. To ensure consistency, this straightforward approach is similarly chosen for the here relevant user interfaces (see Figures 3.15 3.16).

The actual design is use case dependent and is left to the application developer. During implementation of his application, the developer defines the graphical display. He is provided with a frame of the same size as the sample region in the original image, and may define paintings that interpret his results. Regarding a binary detector, this could be achieved by a simple mark in two colors denoting either positive or negative results. Quantitative evaluation would probably need a range of colors, e.g. from green to red. It would also be beneficial for users to have a short numerical value presented along with the graphics. For the complex use case, e.g. a localization task, n crosses at given points on the sample image would be appropriate.

The detailed information display is designed as a tabular view with variable rows and columns. Each row comprises a *key*, a *value* and a *unit*. Those details may contain further information that cannot be displayed graphically. This tabular view is placed in the administrator interface as and additional window.

It is practical to present data in such a way, since it enables fast copying of data for transfer (drag and drop) to other software. The result overlay mask is made adaptable to the different result types mentioned in Section 2.1.1 by polymorphism. The subsequent two sections each describe one type of the user interfaces as they were developed during this thesis.

3.7.1 Manual Module Interface

The framework is designed to provide two user interfaces. The first is the MMI, which enables the user to execute all functions exposed to the PCS. The user can execute image acquisition and the CV application. Further, all parameters that the PCS can pass to the module are editable. This spans a general system config and information about the current labware.

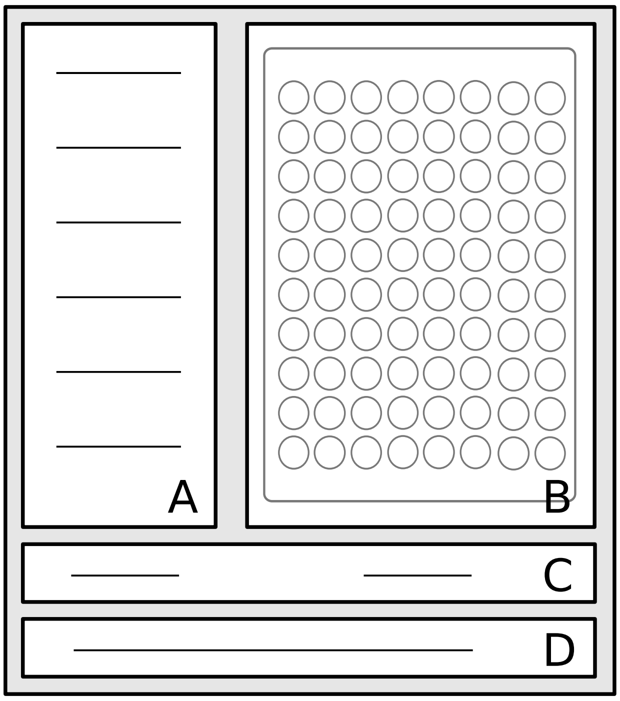


Figure 3.15: Manual module interface

Figure 3.15 shows the GUI layout. Field **A** contains the fixed list of buttons to execute API commands, excluding training and calibration. Field **B** displays the current image and overlaid results. Below, field **C** offers space to set parameters to predefined sets of values (configurations). A typical way to offer a limited range of options to a user is e.g. a pull-down menu. Here, also labware type configuration and expected execution time are set. The field **D** on the bottom is the standard read-only status bar, which is commonly used in applications. It is used to show tooltips to the user or to display log messages during and after execution.

3.7.2 Graphical Administrator Interface

The administrator GUI offers setup, calibration and training methods. It is designed to support, for example, a technician during system configuration. A scheme is shown in Figure 3.16.

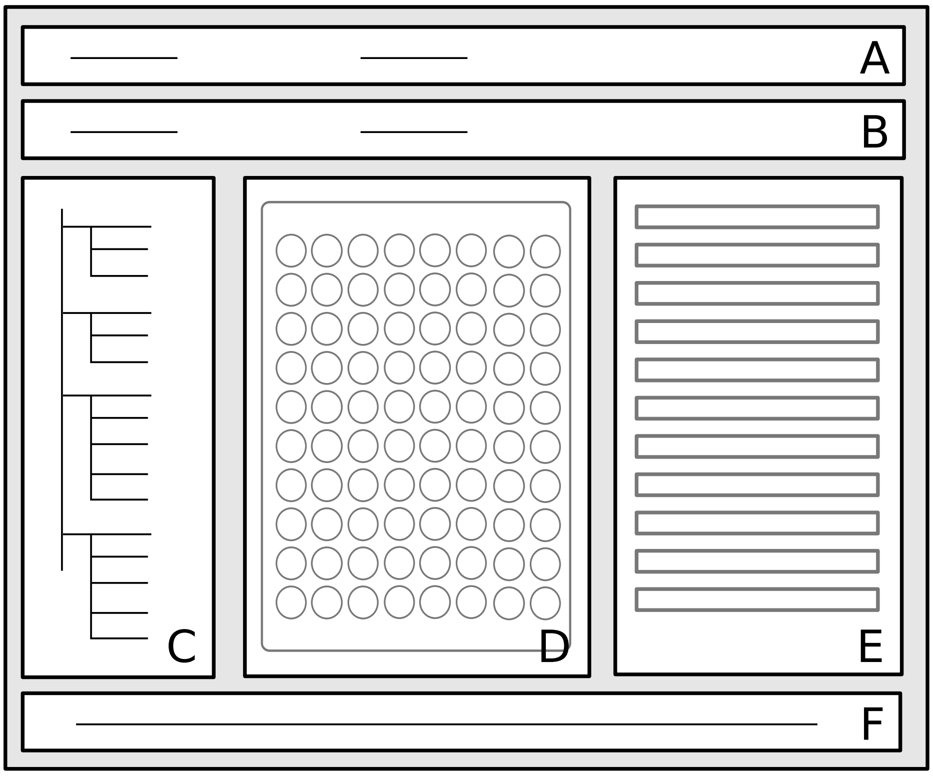


Figure 3.16: Administrator interface

Field **A** holds the API commands, as in the MMI, including training and calibration. Field **B** provides further access to the hardware interface directly to acquire images. In addition to the parameters accessible by the API, all system parameters are view- and editable in the tree view list in field **C**. This comprises hardware and labware-specific parameters and application specific parameters, such as algorithm parameters. The tree further lists images for training, calibration and testing. They are added to the list when they are loaded from disk or acquired by the application. **D** provides a plate image with overlaid results as in the MMI. Field **E** displays additional results. This field is necessary when a large number of results for every well would be difficult to display on the overlaid mask. The field can be used to display locations and other values of found objects, but also for training results and statistics. Again, field **F** at the bottom is the standard status bar.

3.8 Generic Algorithm Optimization

Algorithm optimization is application specific and applicable to CV algorithms used with the framework. This section describes the functionality that the framework exposes to the application developer. The requirements are stated in Section 2.2. The optimization is aimed at computer vision algorithms and hence nonlinearity is considered.

3.8.1 Algorithm Interface

The framework's algorithm optimization is implemented to work with arbitrary algorithms. Using the interface defined at the beginning of this chapter (Section 3.1), it considers the algorithm as a black box that it feeds with input variables and the actual image (i.e. a group of well images), and from which it receives outputs. To support an unknown number of parameters with unknown type, the input value is a dynamically sized key-value map in the form shown in Listing 3.2.

Listing 3.2: Entry of the input list

```
key, {type, value, min, max, stepsize, unit}
```

For every value, its name (key), type, range and stepsize is passed. Thereby all entries except the value are constant for an input value, such that they can be stored together with the algorithm itself. In an object oriented language, a possible way is to define an algorithm class that implements a member function that returns a reference input value with keys, ranges and stepsizes.

The output is a dynamically sized list of matrices with n_{Wells} elements, so that multiple results for every well are possible.

To operate the framework's optimization tool, the supervised-learning rule needs images with reference results of the same type as the algorithm results.

3.8.2 Target Function

The target function $T(x)$ computes a measure of the algorithm's performance from its outputs and provided references. The target function varies, depending on the type of the algorithm and the aim that the optimization should fulfill. By varying $T(x)$, it is possible to weigh specific criteria, such as precision or robustness, according to one's needs during the optimization. The performance measures described in Section 3.1 are provided to the developer, who may use them to design a target function.

3.8.3 Nonlinear Optimization

Nonlinear optimization techniques are iterative procedures: A first set of parameters is initialized and the algorithm under consideration is run with a reference input. After computation, the actual output is compared with the reference output by a target function and a performance measure is calculated. According to a rule set, new input parameters are calculated. The algorithm is executed again in an iterative fashion until its performance reaches a threshold.

Many different rule sets upon which new parameters are calculated from older ones exist. Unfortunately, these techniques are never equally suitable for different algorithms. The common problem is that an algorithm can treat a local minimum for a global minimum or does not converge. Furthermore, these search algorithms are not trivial to parallelize, since the input parameters are not known beforehand and depend on each other, and the strict separation of setup and computing is hence not possible.

Nonlinear optimization is a broad field and further research could aim to find appropriate algorithms for use with the framework. For applications with a very large number of parameters, this could make sense.

3.8.4 Grid Search

During this work, a grid search is implemented, since it is compatible to any algorithm and offered appropriate training times for the tested applications after it was parallelized to use multi-core machines. The grid search was implemented for an unknown number of parameters with a function that calls the (unknown) algorithm in question recursively and iterates over the parameter grid. The basic form of a recursive loop is attached to this work in the appendix in listing A.1.

3.8.5 Linear Optimization

When linear equations are part of the algorithm, it is indicated to obtain optimal parameters by linear optimization, i.e. a least squares approach instead of running time-consuming search algorithms. An Adaline, for example, is a linear function that computes a decision from a set of input values.

3.8.6 Regression

When quantitative measurement is the use case of the application, no inference is needed. The algorithm working on the image returns results that can be regarded as measurement signals, like voltage acts as the signal carrier of a thermistor. The signal has to be transformed into the target unit by a function that depends upon the measurement principle.

The thermistor is considered as an example for clarification: The physical temperature (measured variable) affects the resistance of the thermistor, $\rho = f(t)$. This translates into a voltage that acts as the signal carrier. Subsequently, nonlinear behavior of f is compensated during transformation of the voltage into a measure of temperature $T = g(V(R))$, where the compensation would ideally be $g = f^{-1}$.

3.9 Concurrency Concept for Parallel Region Handling

3.9.1 Basics

Flexible load distribution is required to ensure stability while optimally utilizing available resources. As pointed out in Section 2.7, PCs with multiple cores are a standard today and hence likely to be available for laboratory control. With respect to the application in laboratories, parallelization is only considered for typical workstations with multi-core central processing units (CPUs). This restricts the design to symmetric multiprocessing (SMP) systems, which use shared memory for their cores, and excludes server clusters or similar systems (i.e. systems coupled over network, distributed memory systems).

With the recently established GPU computing interface Compute Unified Device Architecture (CUDA), Nvidia Corporation offers an established interface to make use of a PC's graphics processing unit (GPU) computing resources. This way, it is possible to use a personal computer for high performance parallel computing. But since CUDA is a proprietary interface, and as such is bound to the company's hardware, the advantages can only be exploited with specially equipped computers. If a future application is time-critical and demands high processing power, it can be considered to add CUDA support to the framework and to provide a dedicated system with the CV application. To account for the application area and its restrictions, CUDA is not considered in this thesis.

Data parallelism is immanent in the parallel character of high throughput laboratories and in the character of images as input data, which consist of a fixed number of equally sized and typed datums (picture elements, pixels). Hence, data parallelism is considered in this work.

Image processing is counted among the trivially parallel problems[6], because it mostly consists of independent and equal steps, e.g. the so called *single instruction multiple data* (SIMD) operations. For example when the same calculation (single instruction) is performed on every pixel (multiple data) of an image. A simple example for single instruction multiple data (SIMD) operations is given in the appendix in Listing A.3.

If parallelization is considered for a computationally intensive program, it must be determined which parts of it are *threadable*. *Threadability*, the suitability for multithreading, is given when a program consists of multiple steps that are independent of each other. If parallel execution leads to access conflicts, some problems can be reorganized [6].

A concept to guarantee threadability is the strict separation of a computation into a setup and compute phase. Lebak et al. call this concept *early binding* and also mention it as one of VSIPL's[2] key characteristics [50].

Applied to image processing, it is sufficient to allocate all buffers prior to starting the calculation operation to ensure threadability for a large number of image processing algorithms. When neighborhood operations are considered, overlapping read access must further be provided. Furthermore, when a calculation is started in a loop with an unknown number of repetitions, for example during a grid search where the parameter sets are calculated using recursive iteration, access conflicts occur. It is indicated to reorganize the program into the setup phase where the recursive loop stores all sets in memory, and into the computation phase where the computations are run in parallel.

When threadability is assured, speedup but also efficiency have to be considered. Efficiency in the scope of distributed programming denotes the computation speedup versus the processing power respectively the number of processors:

$$speedup(n_{data}, n_{proc}) = \frac{t(n_{data}, 1)}{t(n_{data}, n_{proc})} \tag{3.28}$$

$$efficiency = speedup(n_{data}, n_{proc})/n_{proc} \tag{3.29}$$

Doubling processing cores will never cut processing time to 50%, this is caused by the serial parts of the program that cannot be parallelized. *Amdahl's Law* provides values for a theoretical maximum speedup depending on the parallel/serial ratio of work and the number of cores. It can be regarded as an upper limit for speedup and efficiency, since it only considers the serial/parallel ratio. Additionally, other causes for efficiency losses are possible ([6] [21]):

1. sequential parts, such as initialization and I/O operations
2. thread initialization
3. communication and synchronization between threads
4. load imbalance (threads are waiting for a slower one to finish)

The first cause points to parts of a program that cannot be threaded, such as the memory allocation of example listing A.2. The input data partitioning for the distribution on multiple cores must also be handled by a single thread.

[2]VSIPL: Vector Signal Image Processing Library

The second point adds to the sequential parts of the program, because every parallelized application needs to spawn other threads from the main thread. This initialization is additional work that adds to the unthreaded program. If the parallelized portion is small compared to the thread initialization work, the efficiency could theoretically decrease by parallelization.

Communication and synchronization overhead is minimal for SIMD operations, when no communication of intermediate data is required during the computation. Considering the fourth point, synchronization of identical instructions is often trivial, because every instruction finishes in the same number of clock cycles. Algorithms whose computational complexity depends on the input data, such as the Hough transformation [20], are exceptions that should be considered.

The parallel character of laboratory tasks offers possibilities to distribute the processing load with respect to the four points mentioned above. The optimal utilization of multithreading is given when:

1. Optimal efficiency and speedup and
2. Hardware compatibility

are achieved. The framework supports three key concepts of parallel programming.

3.9.2 Concepts for Parallel Processing

As mentioned above, the first concept is the strict separation of computations into a setup and compute phase. Listings A.4 and A.5 in the appendix show the separation of serial and parallel tasks using the example from above. This separation is catered for by the definition of threadable parts of the application. Wherever possible, the program is reorganized to maximize the threadable part of the application.

As pointed out above, the efficiency is strongly influenced by the complexity of the CV algorithm, which is executed in parallel, as well as the image size. The second concept allows parallelization with different minimal chunk sizes, that are defined as different parallelization levels listed in Table 3.3. The levels are implemented to enable flexible development of efficient code.

By defining a well sub-image as the smallest chunk size, the parallelization is completely kept away from the application developer, who develops his algorithm on a single well image. During parallelized operation within the framework his algorithm will execute

Table 3.3: Parallelization levels

Level	No. Plates	No. Images	No. Wells
Plate Stack	≥ 1	≥ 1	≥ 1
Plate	1	≥ 1	≥ 1
Image	≤ 1	1	≥ 1
Well	≤ 1	≤ 1	1

the way he designed it. The application developer does not need to care about access problems during development.

As shown in Section 2.5, hardware exists where one plate is captured with more than one imaging sensor. When considering the parallelism of the data flow, an image is hence the next larger set in which pixels can be grouped. This level is relevant for multi-camera devices.

The plate itself is a set of well images from one or more sensors, and a plate stack consists of one or more plates. A plate stack can be relevant during algorithm training, where multiple plates have to be regarded as a single input to a training algorithm. Plate stacks are not handled during operation (see Figures 2.4 and 2.5). Figures 3.17 and

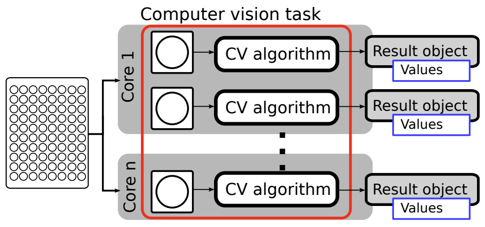


Figure 3.17: Well-based threading

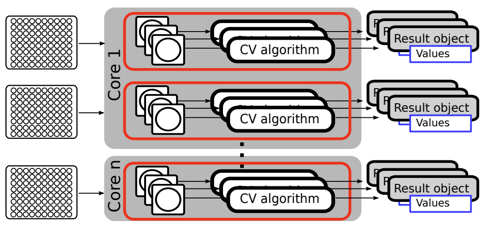


Figure 3.18: Plate stack threading

3.18 show a schematic view of parallelization on two levels. When a plate is processed concurrently, and it is inside a plate stack, e.g. during training, nested parallelism occurs.

The third concept is *thread pooling*, a prevalent pattern in software development [68] that offers large possible performance gains in computationally intensive tasks [69]. A thread pool keeps a number of threads on standby to overtake tasks when parallel processing is possible. This way, efficiency losses due to thread creation and destruction can be minimized. The performance gained with a thread pool depends on the task type and the number of processors, whereas the number of threads in the pool influence the efficiency to a large extent [51]. The number of threads must therefore be configurable by the application developer or user.

3.10 Communication Interface to Process Control Systems

3.10.1 API Description

The API defines the function signatures with whom the framework is controlled and results are obtained. Besides the functions required during operation, the framework can be used in the context of hardware and software calibration and respective extended functionality must be provided. Necessary functionality is described in Section 3.7 and the API functions are summarized in Table 3.4.

Table 3.4: API summary

Function	**Input**	**Output**	**Description**
compute	labw. type	list of results	Executes the application (*Operation* use case)
init	config	-	initializes the application
acquire	-	-	acquires image
open/close	-	-	dummy for plate loader
compute	file	list of results	Executes the application with image from disk
calibrate	list of files	parameter set, plate info	starts calibration, returns the parameter set and results for each plate
training	list of files	parameter set, plate info	starts training, returns the parameter set and results for each plate
command_list	-	key-value map	outputs names of the application and its actions for user interfaces
undistort	-	image	outputs the current plate stiched and corrected for user interface

3.10.2 Integration into Message-Based Process Control Systems

To expose the API to the PCS, a small translator module between the framework API and the control system is required. This way it is possible to use the framework with different control systems by only changing the translator module. Hereby, the control system's specific characteristics have to be taken into consideration.

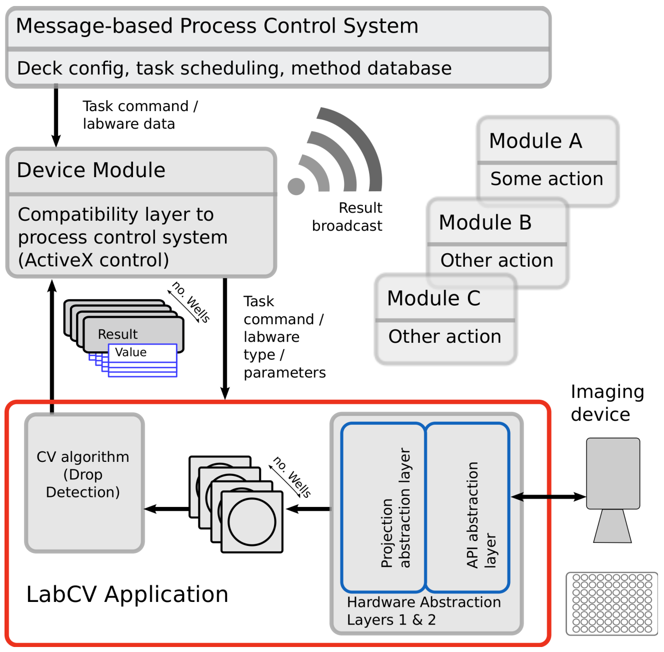

Message-based Process Control System
Deck config, task scheduling, method database
Task command / labware data
Device Module
Compatibility layer to process control system (ActiveX control)
Result broadcast
Module A
Some action
Module B
Other action
Module C
Other action
no. Wells
Result
Value
Task command / labware type / parameters
Imaging device
CV algorithm (Drop Detection)
no. Wells
Projection abstraction layer
API abstraction layer
Hardware Abstraction Layers 1 & 2
LabCV Application

Figure 3.19: Framework integration into process control software

Fig. 3.19 shows the implementation using a translator module ("Device Module") for a messaged-based process control system. In this system, messages are broadcast to all devices. The devices are configured according to their task to listen to specific commands. It can be defined within the system which module receives broadcast messages from a specific type or origin. In order to use the results from the CV application, at least one module has to be set to listen.

3.11 Summary of Concepts

The concepts for a computer vision framework for laboratory automation that are presented in this chapter can be summarized in two main parts. The first part is the hardware abstraction layer, consisting of technically related hardware abstraction, i.e. a driver interface wrapper, and optical abstraction in terms of geometrical projection mapping and lens correction including calibration and registration.

The second part is the parallel character of labware. Elaborated further, this leads to regarding wells as data objects and a plate or an image as a collection of such objects, which are an input to a CV application that is defined for a single sample. The parallelization concept is based on this fact because data independence can be assured, and parallel handling on plate-, image-, or well level is possible.

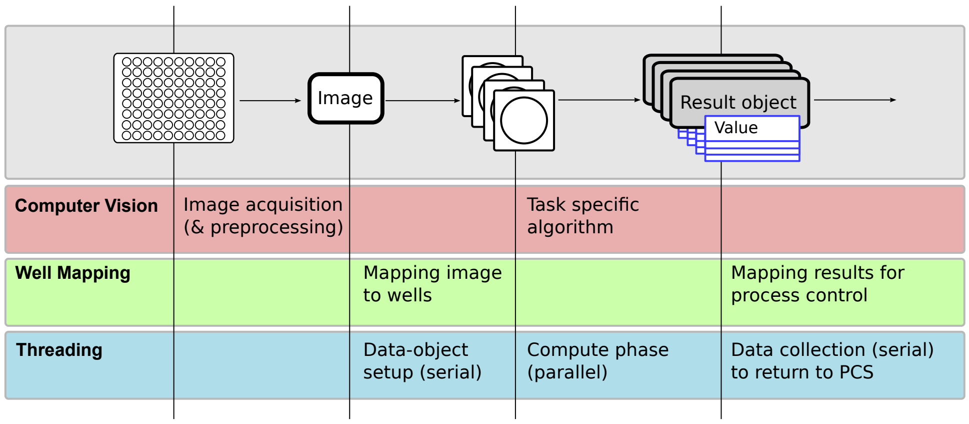


Image
Result object
Value
Computer Vision
Image acquisition (& preprocessing)
Task specific algorithm
Well Mapping
Mapping image to wells
Mapping results for process control
Threading
Data-object setup (serial)
Compute phase (parallel)
Data collection (serial) to return to PCS

Figure 3.20: Summary of the application signal flow

A result type was defined that is compatible to the presented applications, and is expendable by evaluation functions for training and testing purposes for application-independent parameter optimization. In the same manner as the input objects, every processed well delivers an independent result object. All results for a plate are collected, either plain or extended, and can be passed on to the PCS or to a GUI for display or further processing. See Figure 3.20 for a schematic display of the developed application flow.

Furthermore, two GUIs, a manual module interface and an administrator tool, were introduced - both easily adaptable for different applications. To be used by laboratory automation systems, a PCS specific translator layer was introduced.

Chapter 4

Reference Implementation

This chapter describes the software-related implementation of the concepts presented in the previous chapter. A software framework called *LabCV* was developed as a reference implementation for this work. The concepts are generic and not bound to a programming language or paradigm, still, they are especially suited for object oriented programming. The reference implementation is realized in the C++ programming language (C++). The following sections correspond to the respective sections of the previous chapter wherever suitable.

4.1 Software Architecture of the Framework

The software architecture aims at implementing the concepts summarized in Section 3.11 and Figure 3.20. Polymorphism is used to integrate hardware types and labware characteristics into the library in a flexible way. The used concept for separating data and functionality and used libraries are described below, while the following sections will cover the software functionality of the different parts.

4.1.1 Separation of Data and Functionality

It is considered good practice to keep data and functional objects separated. Often, developers choose an architectural concept where the data is kept by a base class, from which a class is derived that keeps the functionality, e.g. hardware communication and algorithms. For the reference implementation, a similar architecture is chosen: At startup, a tree-like configuration object is created that holds the configuration information of all program parts in parameter classes. The parameter-tree is then passed to an application object that gets initialized according to the configuration structure. This approach turned out to be practical for the integration in the graphical user interface, where the user configures the application settings in a tree-structured interface and is then able to launch test runs of the application.

4.1.2 Dependencies

The implementation makes use of platform-independent libraries that are listed in Table 4.1. All libraries are licensed in a way that allows academic use free of charge.

Table 4.1: Used software libraries

Name	Version	Licence	Use
Boost	1.4x	BSD[a]	memory management, file system
OpenCV	2.x	BSD[a]	CV and maths, camera interface
OpenMP	2.5	various[b]	concurrency
TWAIN	2.0	TWAIN[c]	Hardware interface reference implementation
Lumenera	5.0	comm.	Proprietary hardware interface

[a] see [60], [b] open standard, licence depending on implementation,[c] open standard

From the libraries listed in Table 4.1, the Open Computer Vision Library (OpenCV) provides the important image processing and linear algebra part [10]. *Boost* provides memory management and file management and is also part of the core functionality. On the other hand, the implemented hardware interfaces, *TWAIN* and *Lumenera's Lucam* are exemplarily implemented driver modules and could be replaced or extended by other drivers.

4.2 Hardware Interface Abstraction Layer

The framework is required to support any imaging device that delivers an image of a microtiter plate. So far, camera based systems, flatbed scanners and manually operated light tables with a mounted digital camera are supported with the implemented interfaces. All devices that were tested to work with the framework are listed in Table 4.2 and shown in Figures 4.1 and 4.2.

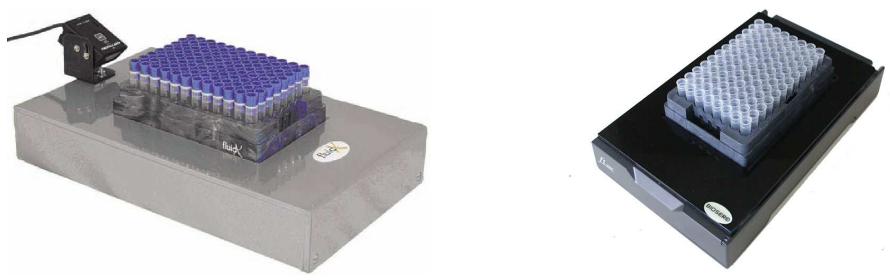

(a) CCD, *XTR96 MK I*, *FluidX Ltd., Cheshire, UK*
(b) CIS, *A6 Rack Scanner*, *Ziath Ltd., Cambridge, UK*

Figure 4.1: Flatbed scanner

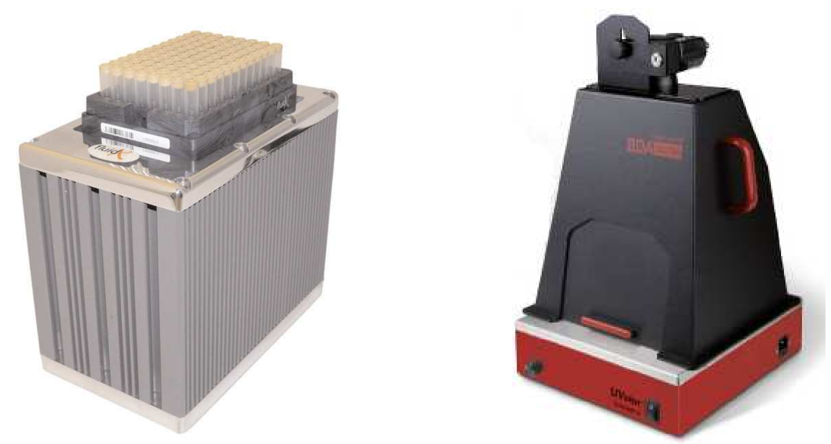

(a) *XTR384Pro*, *Ziath Ltd., Cambridge, UK*
(b) Light table with top mounted camera, *BDA*

Figure 4.2: Camera systems

The hardware interface is defined by a pure virtual base class, which implements all function definitions that are required to be realized with the chosen driver API. The base class is overloaded by an instance of a derived class, depending on the actual hardware (see Figure 4.3a). Table 4.3 lists all function definitions that must be implemented in order to use a hardware device.

The defined functions are based upon the requirements that are imposed by the mapping modules. The differences of flatbed scanners and cameras must be considered. The complexity behind a function from the table depends on the optical system. The frame size, for example, defines the size of the imaged area in object coordinates. A flatbed scanner is able to provide this information by a query to the firmware, since the object plane is defined to be on the flatbed of the device. Whereas a camera does not have such

Table 4.2: Examples for imaging hardware used in laboratory automation

Model	**Manufacturer**	**Sensor(No.)**	**Type**	**Position**	**Lighting**
XTR96-MK1	fluidX, Ltd	flatbed	CCD (1)	below	reflected
A6	Ziath, Inc	flatbed	CIS (1)	below	reflected
Smarscan 96	Thermo Fisher	camera	CCD (2)	below	reflected
XTR384pro	FluidX, Ltd	camera	CCD (8)	below	reflected
Light Table	BDA	camera	CCD (1)	above	transmitted

Table 4.3: Interface operations

function	**scanner**	**camera**
initialization	x	x
acquireImage	x	x
(max)frame	g/s	g
resolution	g/s	g
pixeltype	g/s	g/s
bitdepth	g/s	g/s

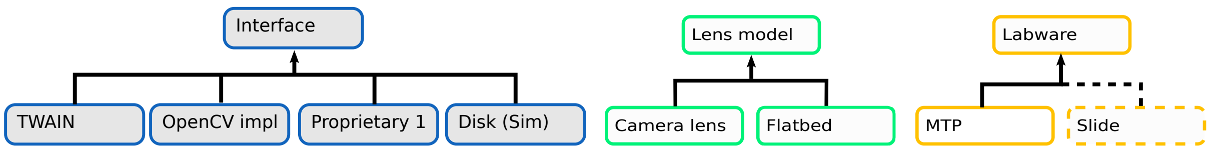
g : get, s : set, x: supported

a defined object plane, hence it is necessary to define the object plane and to consider the perspective to calculate the dimensions of the area covered by the image.

Parameters that only apply to either flatbed scanners or cameras cannot be required to be alterable by the overlying modules. Such parameters still have to be set up by the application developer and hence initialization sequence must be provided for every hardware API to set up driver and device-specific parameters. During operation, the configuration remains static, and no *set* operations are allowed.

Parameters that will not be considered further are compiled in the list below. They are set during the initialization phase.

- Resolution (scanner)
- Frame size (scanner, camera possible)



(a) Interface objects **(b)** Lens model objects **(c)** Labware objects

Figure 4.3: Polymorphisms

- Bitdepth (scanner possible, camera possible)
- Pixel type (greyscale, color)
- Exposure (camera only)
- Light source setting (scanner only)
- Color correction / White balance (scanner, camera)

To evaluate the feasibility of an implementation with different driver APIs, three examples are considered. The *TWAIN* image acquisition standard is designed to support flatbed scanners but also cameras; *OpenCV* itself wraps operating system-dependent interfaces[1], mainly for cameras; and *Lumenera* is a proprietary API. The implementation of a simulation interface, which loads available images from disk, is the fourth implemented derived class.

4.2.1 TWAIN Interface

TWAIN is a popular standard API for Windows. The *TWAIN* Working Group consists of delegates from device manufacturers and software companies. The *TWAIN* standard is based on the consensus that an open standard is beneficial for all parties [20]. By supporting *TWAIN*, it is possible to cover a large amount of devices.

The *TWAIN* standard uses a central interface management, the Data Source Manager (DSM), to interface a compatible device, which is called Data Source (DS). Together with the application, a DSM and a DS form the system. The communication protocol defines message triplets of the form: 1) *Group*, 2) *Data/command type*, 3) *Message*.

The Data Group (DG) defines the message's high-level membership to a command group such as DG_CONTROL for control or DG_IMAGE for image-specific settings. The

[1]Used interface for a) MS Windows: DirectShow, b) Linux: V4L

second part is the command type, e.g. DAT_PROCESSEVENT (process an event) or DAT_STARTXFER (start native transfer). The message is terminated by the actual command, such as MSG_GET, MSG_SET, MSG_ENABLE. A declaration of message origin and target and a data container is added to the triplet. Altogether, this results in six-part procedure calls:

Listing 4.1: *TWAIN* Procedure Call

```
origin, destination, group, type, command, data buffer
```

Table 4.4 shows the key triplets that are used to implement the required operations from Table 4.3. An operation usually consists of multiple calls, e.g. status requests or unit checks; the table lists only the most relevant calls.

4.2.2 OpenCV Interface

With its implementation of the *DirectShow*, resp. *V4L* API, OpenCV supports the majority of cameras dedicated to be used with PCs. The majority of this kind of cameras are used as webcams, but manufacturers of professional cameras sometimes provide a DirectShow interface, too. By providing both *TWAIN* and *DirectShow*, a huge number of imaging devices is covered. The OpenCV interface supports basic frame grabbing and get/set operations for a number of parameters, as shown in Table 4.4.

4.2.3 Proprietary Interface

It is possible to use the library with other APIs. The derived class must implement the operations described in Table 4.3. As an example for a proprietary interface, the *Lumenera* camera driver is considered. Table 4.4 shows the implementation of interface operations *Lumenera* cameras.

4.2.4 Disk-Interface

By implementing file reading to open images from disk, a simulation, testing and offline batch-processing interface can be created.

Table 4.4: Three implementations of the interface class

device function	***TWAIN***	***Lumenera***	***DirectShow* / OpenCV**
initInterface	DG_CONTROL/CONTROL_DSM/MSG_OPEN DG_CONTROL/CONTROL_DS/MSG_OPEN DG_CONTROL/DAT_XFER/MSG_NATIVEXFER	LucamEnumCameras LucamCameraOpen LucamSetProperty(...)	VideoCapture::ctor(id)
acquireImage	DG_CONTROL/DAT_USERINTERFACE/ MSG_ENABLEDS	LucamTakeSnapshot LucamConvertFrameToRgb24	VideoCapture::operator>>()
get/set Resolution	DG_CONTROL/ICAP_RES/MSG_GET (DG_CONTROL/ICAP_FRAMES/MSG_GET)	read image size perspective transform calculate resolution	VideoCapture::get(W/H) perspective transform calculate resolution

**get/set
parameter**
	DG_CONTROL/PARAMETER/
	MSG_GET / MSG_SET
		LucamSetProperty(..)
		LucamGetProperty(..)
			VideoCapture::set(..)
			VideoCapture::get(..)

parameter is a implementation specific identifier for frame (image size in pixels), pixeltype or bitdepth			

4.3 Projection Mapping Layer

When an image is taken, the image data is written to a data buffer by the hardware interface. The next step is the mapping of samples, as presented in Sections 3.3, 3.4, 3.5 and 3.6.1. This is divided into the projection model and the sensor map (see Figure 3.5).

By overloading the lens model and the labware type (see Figures 4.3b and 4.3c), the system is set to a configuration. During operation, the models are addressed by a *Scene* class to calculate the transformations. To find a well on the image, it first requests the rectangle on the labware from the labware class. Then it forwards the rectangle to compute the *labware→device* transformation and the *device→image* transformation, which are part of the registration part and the loaded lens model.

This mapping computation results in a rectangular ROI, which can be used to define a sub-image, i.e. a sub-array of the image. Figure 5.1 shows the found sub-images that are passed to the application. If the wells of the current plate have a circular shape, it can be indicated to further specify the area that represents the well and leaves out the four corners. In order to do that, a circle, with its center point corresponding to the well center and its diameter corresponding to the size of the well, can be applied as a mask. The mask can be created by the downstream algorithms if necessary.

However, the same transformations are important when the location of a found object is required in device or robot coordinates. Hence, all transformations are implemented for point objects, and a rectangle is defined by a set of two points. The transformations are controlled by the Scene class.

Copying of data is a computationally expensive task. Hence, the generation of sub-image objects is realized in a way that does not require any copying or movement of image data. This is achieved by objects that do not hold the image data but reference to parts of the original image. A header containing management information and references to the data is the only memory overhead of the objects. This design maximizes performance, since no data has to be moved in memory. When a new image is loaded in place of the older one, the sub-images must not necessarily be created again when the labware is of the same type. The objects can be viewed like a mask above the image that is replaced by a new one. By using a set of objects instead of an array of rectangles, it is easy to parallelize the computation on a well basis.

4.4 Calibration and Registration

The device setup includes interface setup and calibration and registration. The necessary steps to set up a hardware requires a driver installation and the configuration of proper system settings. In the administrator user interface, Figure 4.8, the device controls and settings are displayed in the top left of the window. The application settings can be changed in the tree view on the left.

To meet the requirement of a "one-click" device installation, a straightforward method for hardware calibration, i.e. the determination of the model parameters, is needed. The determination of calibration parameters is done with a checkerboard that acts as the reference object.

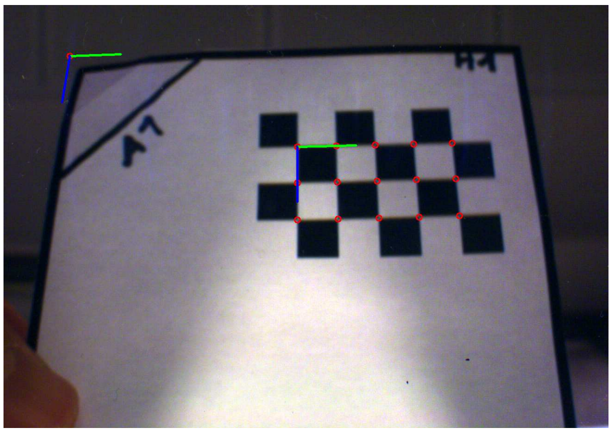


Figure 4.4: Camera calibration

The first step is to find all corner points of the checkerboard in the images. This is done by binarizing the image using adaptive thresholding and subsequently running a corner detection algorithm that searches for a binary checkerboard structure by evaluating logical relationships. By averaging the found locations of the checkerboard corners, the found intersections have sub-pixel accuracy. The subsequent steps are device dependent and implemented in the corresponding classes (either camera or flatbed model). The procedure follows the steps described in Section 3.4.2 and Section 3.5.2.

More than one image is required for camera systems, so that the plate must be repositioned multiple times. This also applies for flatbed scanner calibration, although less parameters have to be determined and hence fewer images are sufficient. With any device, the user acquires an amount of images, and checks for every image if the checkerboard is found. A number of 5-10 images was found to be a good reference for a camera system.

For registration, one image has to be acquired with the registration plate, a purpose-built plate that fits into the labware position, being positioned on the labware position. The image is then used for the registration of coordinate systems and multiple sensors if applicable. With the plate being positioned properly, equations from Sections 3.4.2 and 3.6 are satisfied. One has to keep in mind that tolerances mentioned in (3.6) apply.

Figure 4.4 shows an image taken during the camera calibration procedure, where the checkerboard was found and the origin of the labware coordinate system was properly placed in the image. The plate is held in its position by the user.

Figure 4.5 shows the image taken for registration. Here, the plate's dimensions are equal to a microplate, and the plate aligns with the device's labware position. Now that these points are known to the system, Equations (3.23), (3.25) and (3.26) can be solved.

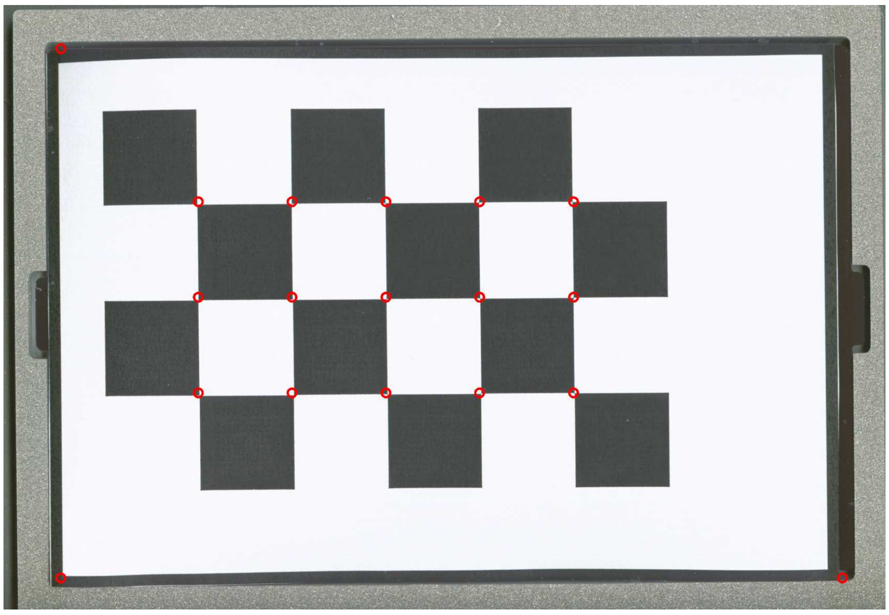


Figure 4.5: CCD scanner calibration, plate at the registration position

Labware based Calibration of the Optical System

Instead of a checkerboard as the calibration object, it is also possible to use other objects for calibration as long as their geometry is known. The geometry of a microplate is known and standardized, and round wells appear on the image as grid-wise ordered circular objects. As such, they can be used for calibration. In fact, pattern of circles lately seem to supersede checkerboards as popular calibration objects within the robotic vision community, because they often provide more robust results. Microplates, however, are not ideally black and white and their features are more difficult to detect on the image. When an algorithm can be found to robustly detect their locations, they can be used as inputs to the calibration algorithms, i.e. as c_i for (3.9) for the flatbed scanner model.

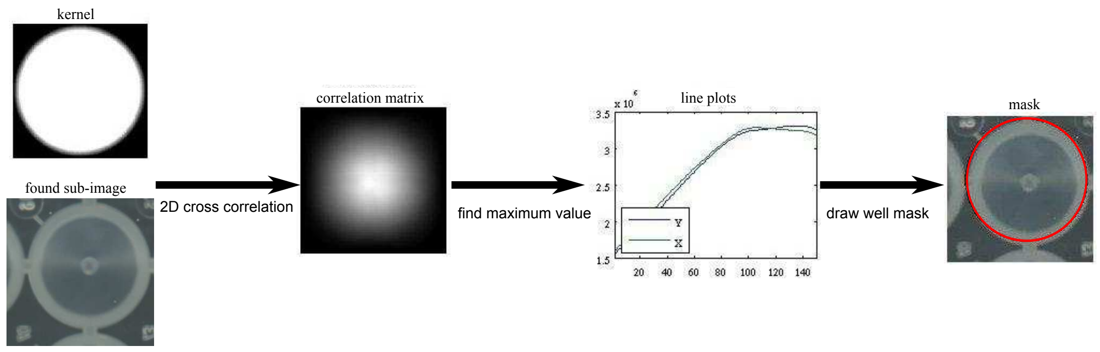


Figure 4.6: Well center detection principle

With the algorithm whose principle is shown in figure 4.6, it is possible to detect a circular-shaped well of a microplate. Subsequently, the algorithm runs a cross-correlation with a circular-shaped kernel $h(x, y)$ over the sub-image $g(x, y)$. The result is a matrix with values corresponding to the similarity of the kernel and the image for different positions of the kernel. The maximum value depicts the best match of both functions $g(x, y)$ and $h(x, y)$.

The current algorithm was tested with two parametrized kernels on three different plate types. Figure 4.6 shows the filled type created according to Figure A.4b. It was observed that the outermost wells could not be detected reliably, but apart from that, the algorithm proved to be able to work with different plates and also with filled wells. The approach should be investigated further in the future, to find out whether special calibration plates could be rendered obsolete. An excerpt from the first test run is attached to this thesis in Appendix A.6.

4.5 Generic Algorithm Optimization

4.5.1 Grid Search

The grid search implementation makes use of recursive loops to support variable numbers of parameters, ranges and stepsizes. The classic form of a recursive loop is shown in the appendix in Listing A.1.

Listing 4.2: Implemented recursive loop to generate input data using container classes

```
p_act   // ITERATOR POINTING TO param_storage
algorithm_params::recursion(
                container<algorithm_params >::iterator  p_act ,
                container<algorithm_params>& param_storage )
{        /* BASE CASE */
        if(p_act == this->end())
                param_storage.push_back(*this);
                return;
        else{
            n_steps = calc_steps(p_act)
            for(i = 0 ; i < n_steps){
                /* RECURSION STEP */
                recursion(++p_act , param_storage);
                p_act = set_param();
            }
        }
}
```

Listing 4.2 shows the implemented form. It was adapted to the object-oriented paradigm of the *LabCV* implementation and implemented using container classes and their iterators. Herewith, it is compatible with the defined algorithm inputs.

With respect to multithreading support, it outputs a list of parameter sets to be tested, together with a reference to the actual algorithm. From this list, the computations can be dispatched to the available threads. Each parameter set is used to evaluate a plate stack.

4.5.2 Target Function

The evaluation classes described in Section 3.8 can be used for performance evaluation and to define appropriate target functions.

4.6 Communication Interface to Process Control Systems

Every application is an inherited class of the *LabCV* application class. It defines the interfaces to the process control system as predefined virtual commands to trigger actions.

The interface provides functions to initialize the system, start an image acquisition and to run image evaluation, calibration and training routines. For the integration into the PCS only the subset of functions displayed in the first part of Table 3.4 is implemented.

The adaption to a specific control software is done via a layer between the library and the control software. The layer translates the messages between both systems, the current mapping is displayed in Table 4.5. The implemented translator is compatible to a SAMI/SILAS process control system.

4.6.1 The SAMI / SILAS System

The reference implementation is integrated into Beckman Coulter, Inc's SAMI/SILAS [63] message-based process control system. The Sagian Automated Methods Interface (SAMI) schedules and runs experiments (called "methods") and SILAS is its interprocess-communication interface. SILAS uses *ActiveX* controls and defines a message protocol. The results are returned using a SAMI *data message*[2], which is broadcast to all modules loaded in the system.

Table 4.5: SAMI - API command mapping

SAMI (Parameters)	**LabCV(Parameters)**	**Description**
initialize	init	load module
open	initLabwareType???	happens before a plate is placed on the device
close	-	
execute	compute	run application and return results

All modules being part of a SAMI/ SILAS system can register capabilities they implement to the PCS. Capabilities that apply for imaging devices are:

[2]Its format is shown in the appendix in Table A.2

SAMIStation: The device is able to hold a single stack of labware.

PositionHost: Describes physical labware positions, and the actions required to reach them. For example when a lid has to be opened or a shuttle is required to transport a plate to the position.

ManualControl: This capability indicates that a manual module interface is available.

Drawing: If implemented, a graphical representation of the device can be presented to the user.

The communication between module and overlying system is done via *device messages* and *consumer messages*. *Device messages* control the device and include configuration and commands. The CV device module communicates results of a measurement via *device data* messages. *Consumer modules* are data-processing modules. They are set up for a run and listen for device data messages of a specific type or origin. The configuration of such consumer modules herewith defines the way the system processes the results. It is, for example, possible to setup a database writer module to realize a documentation functionality. Or, by setting up a liquid handler module to listen to the messages, a feedback control can be realized.

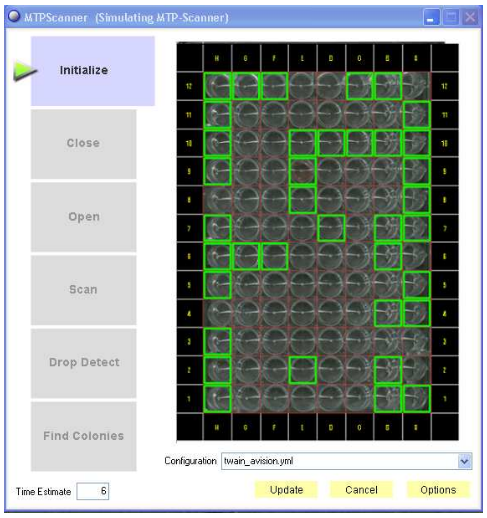
4.7 Operation and Administration User Interfaces

The standard operating system for process control computers in laboratory automation is still *Microsoft Windows XP*. The native libraries for GUI development are part of the Microsoft Foundation Classes (MFC).

By using the native implementation, a familiar interface guarantees fast learning and comfortable use. Techniques such as drag and drop and standard behaviors of mouse and keyboard inputs can be implemented.

4.7.1 Manual Module Interface

As the implementation uses the commercially available SAMI/SAMI's interprocess communication interface (SILAS) from Beckman Coulter, Inc., the MMI is implemented as a SAMI module. The displayed button text is defined by the loaded application, such that the module itself is usable for any application implemented using the framework.



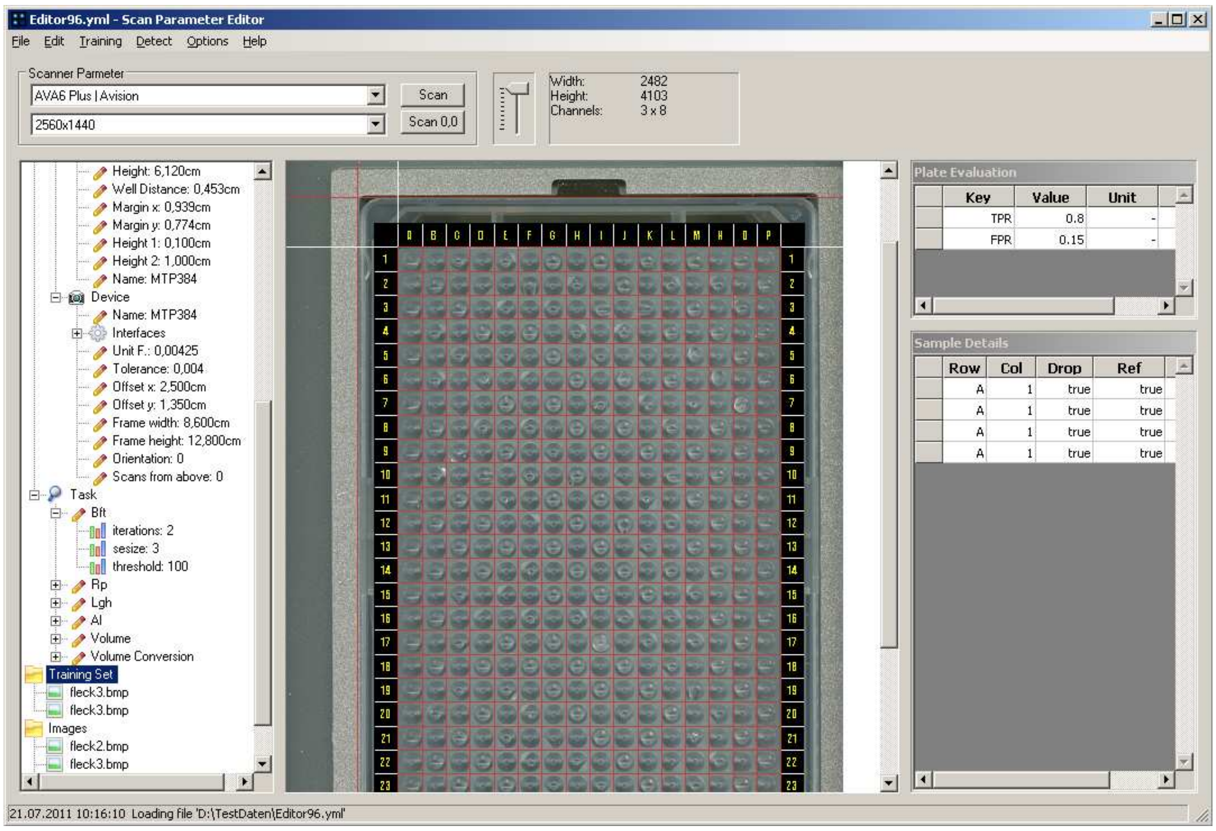
Configuration twain_avision.yml
Time Estimate 6

Figure 4.7: MMI screenshot

The plate image is also delivered by the application. The image is rectified, and possibly multiple images are stitched together in the case of a multi-sensor device. It is further possible to display results graphically as described in the concept. The module itself was designed such that it does not display extended information or statistics.

4.7.2 Administrator Interface

An administrator interface was written following the concepts and layout of Section 3.7. A screenshot is shown in Figure 4.8. During this work, the administrator interface was designed to be independent from a PCS. If the system is used in a regulated environment, however, it would become necessary to integrate the software into a system providing user administration and audit trails for change tracking.



Scanner Parmeter: AVA6 Plus | Avision; 2560x1440

```
Width:      2482
Height:     4103
Channels:   3 x 8
```

```
        Height: 6,120cm
        Well Distance: 0,453cm
        Margin x: 0,939cm
        Margin y: 0,774cm
        Height 1: 0,100cm
        Height 2: 1,000cm
        Name: MTP384
    Device
        Name: MTP384
      Interfaces
        Unit F.: 0,00425
        Tolerance: 0,004
        Offset x: 2,500cm
        Offset y: 1,350cm
        Frame width: 8,600cm
        Frame height: 12,800cm
        Orientation: 0
        Scans from above: 0
  Task
    Bft
        iterations: 2
        sesize: 3
        threshold: 100
    Rp
    Lgh
    Al
    Volume
    Volume Conversion
Training Set
    fleck3.bmp
    fleck3.bmp
Images
    fleck2.bmp
    fleck3.bmp
```

Plate Evaluation

Key	Value	Unit
TPR	0.8	-
FPR	0.15	-

Sample Details

Row	Col	Drop	Ref
A	1	true	true
A	1	true	true
A	1	true	true
A	1	true	true

```
21.07.2011 10:16:10  Loading file 'D:\TestDaten\Editor96.yml'
```

Figure 4.8: Administrator user interface screenshot

Chapter 5

Framework Validation

Validation tests were run to investigate the performance of the framework's projection mapping capability and of its load distribution method. Both tests, their results and discussion can be found in this chapter. The overall usability of the framework was demonstrated by an example application that was integrated using it. This is found in Chapter 6.

5.1 Validation of Projection Mapping

The developed calibration and registration procedures have to be validated to provide sufficient accuracy and precision. A typical camera calibration is validated by the reprojection error (5.1). This error is calculated using the corner positions on the image plane p_i^*, as they are found by the corner detection algorithm, and projected points from the object coordinate system $p_i = (R_{ic}|t_{ic})p_c$. The usual measure is based on the root mean square (rms) L^2-norm:

$$rms = \sqrt{\sum_{n_c} \frac{1}{n_c} \|p_i^* - p_i\|^2} \tag{5.1}$$

Flatbed Systems

If this error is observed over all calibration images, it gives a good sense of the error introduced by the *labware→image* transformation. The errors computed with Eq. (5.1) are displayed in Table 5.1. To support the findings, an uncertainty analysis according to the guide to the expression of uncertainty in measurement (GUM) was conducted. It can be found in Section5.2.

Table 5.1: Reprojection errors

	$\mathbf{n_{img}}$	**rms [px]**	$\sigma_{\mathbf{L2}}$**[px]**
FLXCCD	1	3.4572	1.6443
ZIACIS	1	4.3056	2.0749

Camera Systems

The developed method was evaluated with a dual camera system, of which both cameras have a resolution of $1600x1200px$. Their intrinsic characteristics are displayed in Table 5.2. The distance between the cameras is ca. $6cm$, arranged on the central long axis of the labware position, below the plate. The device is the commercially available *ABgene Smartscan 96 (Thermo-Fisher Inc.)*. Found intrinsic parameters are displayed in Table 5.2. A result mapping is illustrated in Figure 5.1.

Table 5.2: Camera characteristics

		Cam 1	**Cam 2**	
Focal length x	f_x	1.673e+3	1.837e+3	px
Focal length y	f_y	1.690e+3	1.843e+3	px
Center x	c_x	7.850e+2	7.996e+2	px
Center y	c_y	5.461e+2	5.990e+2	px

The rms, computed with (5.1), are displayed together with mean and standard deviation in Table 5.3. The first part ($C \to I'$) of the table shows the reprojection errors during single-camera calibration, where object coordinates were given in a unique checkerboard coordinate system (C) for each of five images.

The second part shows the overall system error that will be faced during operation for each camera and overall, when points on the labware L are mapped. The camera's mean errors show a constant offset of approx. $3px$ in x-axis and $0.5px$ in y-axis directions. Standard deviations amount to $2px$ and $0.3px$ respectively. As for the flatbed model, an uncertainty analysis is available below.

Table 5.3: Reprojection errors, dual-camera system

		Cam 1 [px]		**Cam 2 [px]**		**Overall [px]**	
		x	y	x	y	x	y
	mean	7.3e-4	2e-4	-0.40	0.89		
$C \rightarrow I'$	σ_{L2}	0.45	0.28	0.68	0.72		
	rms	0.44	0.28	0.78	1.14		
	mean	3.20	-5.7e-3	3.83	6.05-e3	3.52	1.4e-4
$L \rightarrow I'$	σ_{L2}	1.91	0.33	2.21	0.55	2.08	0.45
	rms	3.72	0.33	4.42	0.55	4.08	0.45

5.2 Theoretical Mapping Uncertainty and Discussion

The analysis evaluates the uncertainty in which a position on the labware can be mapped. The first step ($L \rightarrow D$) is hardware independent. Subsequently, the estimation is split for flatbed and multi camera systems. The approach is to calculate the device-independent part first and use the result for the different subsequent steps by the result uncertainty $u(p_d)$. Gaussian error propagation is used in the calculation:

$$u(F)^2 = \sum \frac{\partial F}{\partial x_i}^2 u(x_i)^2 \tag{5.2}$$

Unfortunately, many input values are only available as type-B uncertainties, i.e. values from documentation or from experience. Even though the significance is reduced because of this lack of information, the models can be used to determine the influence of a certain parameter x by the calculated slope $^{\partial u}/_{\partial x}$. A rectangular shape is assumed for all type-B uncertainties according to GUM principles [43]. The values of the following calculations are presented for one axis (x).

5.2.1 Transformation Labware - Device

We regard the transformation of a point p_l into the device coordinate system. The first investigated transformation is (3.6):

$$p_d = R(\alpha)p_l + t_{dl} + \epsilon \tag{5.3}$$

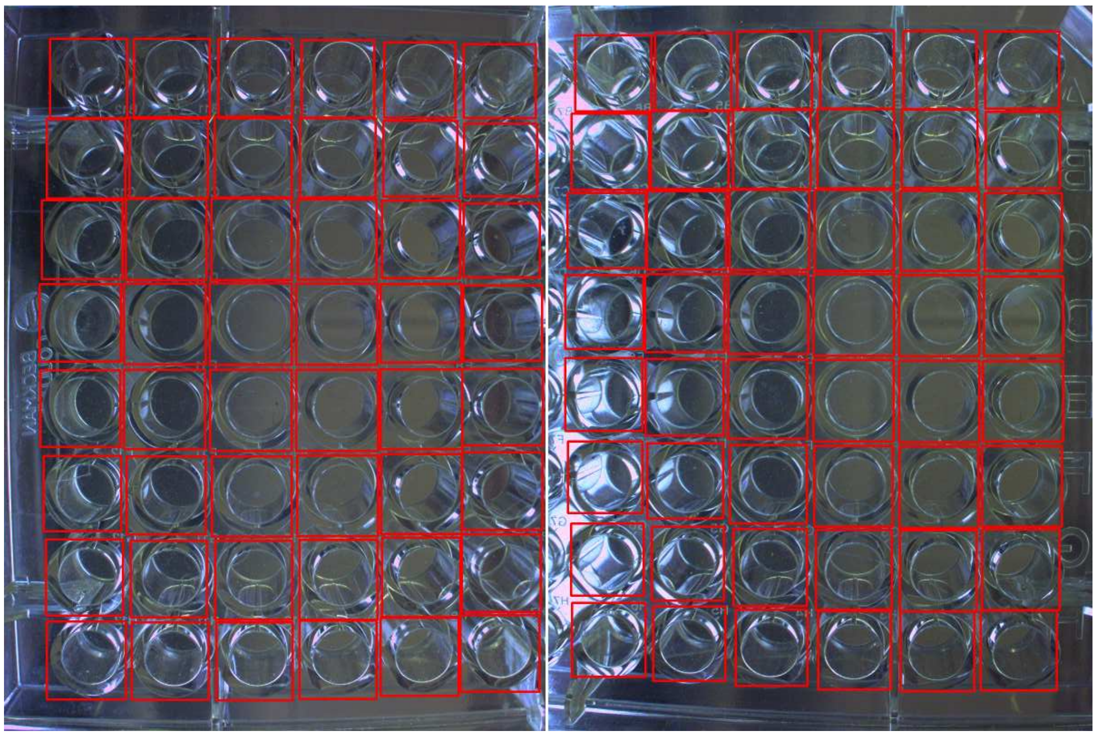


Figure 5.1: Resulting sample map

p_l is the system input (the point to be transformed to the image plane), and is assumed to be an ideal value of 60mm, half the length of a microplate. Angle α describes the rotational error with which the plate is placed on the device's labware position. It is correlated with ϵ such that only either one can have maximum error at a position.

Vector $t_{dl} = t_{cl}$ is influenced by the placement of the registration plate (setting D), and the placement of the actual labware L. In both cases, the labware position uncertainty of $0.1mm$ apply. Additionally, small uncertainties introduced by plate manufacturing, e.g. with a high precision printer ($1200 - 2400dpi$), come in as well. The relative printer uncertainty is estimated to be ${}^{1}/{}_{1200}{}^{2} + {}^{1}/{}_{1200}{}^{2}$ =1e-3 in mm. Note that also the absolute precision of the printer, relative to the paper margins, influences t_{cl}, but is neglected herein. If applicable, other manufacturing options such as laser cutting can provide better results here. The error introduced by the corner detection algorithm used during registration is furthermore neglected, since it is supposed to be on a subpixel level and regarded insignificant. All influences add up to

$$u(t_{dl})^2 = 0.1^2 + 0.1^2 + {}^{1}/{}_{1200}{}^{2} + {}^{1}/{}_{1200}{}^{2} \tag{5.4}$$

Table 5.4: Uncertainty budget $L \rightarrow D$ transformation

Symbol $\mathbf{x_i}$	**Source of uncertainty**	**Value**		**Distribution**	**Divisor**	**$\mathbf{u(x_i)^a}$**
p_l	System input	120	mm	-	-	0
t_{dl}	Plate position, printing	-	mm	rectangular	$\sqrt{3}$	0.103
ϵ	Labware position tolerance	-	mm	rectangular	$\sqrt{3}$	0.1
α	Labware position tolerance	0	deg	rectangular	$\sqrt{3}$	0.05
p_d		-	mm	normal		0.103

[a] in the unit of the respective value

The resulting model is (5.5), and the input values and resulting $u(p_d)$ are displayed in Table 5.4.

$$u(p_{d,1})^2 = u(t_{dl})^2 + u(\epsilon)^2 + R(\alpha)' \cdot p_l \cdot u(\alpha)^2 \tag{5.5}$$

5.2.2 Flatbed Scanner

This step investigates the transformation from $D \rightarrow I$, (3.7). Using the found uncertainty $u(p_d)$, we retrieve the overall uncertainty of the L to I transformation.

$$p_i = R(\beta)S(s)p_d + t_{id}, \tag{5.6}$$

Angle β reflects production tolerances of the labware position to the scanner drive axis, which also is one image axis. p_d is set to 40mm, because D is set somewhere on the microplate.

The scale factor depends upon the resolution of the scanner. It is set to 600dpi, and the uncertainty to 5%. The uncertainty of t_{id} results from the calibration using the calibration plate ($0.1mm$) given in px. The values are listed in Table 5.5

$$u(p_i) = \sqrt{\left(R'(\beta)S(s)p_d\right)^2 u(\beta)^2 + \left(R(\beta)p_d\right)^2 u(s)^2 + \left(R(\beta)S(s)\right)^2 u(p_d)^2 + u(t_{id})^2} \tag{5.7}$$

Table 5.5: Uncertainty budget L to I transformation with flatbed scanner

Symbol x_i	**Source of uncertainty**	**Value**		**Distribution**	**Divisor**	**$u(x_i)^a$**
p_d	Point on device (table (5.4))	40	mm	normal	1	0.103
β	Manufacturing tol.	0	deg	rectangular	$\sqrt{3}$	0.1
s	Scale factor	0.042	$\frac{mm}{px}$	normal	1	0.002
t_{id}	transformation vector	-	px	rectangular	$\sqrt{3}$	2.4^b
p_i			px	normal		2.97

^{a} in the unit of the respective value, ^{b} uncertainty of D relative to I is estimated to be equal to $u(p_d)/s$

with the matrices

$$R'(\gamma) = \begin{bmatrix} -\sin(\gamma) & -\cos(\gamma) \\ \cos(\gamma) & -\sin(\gamma) \end{bmatrix} \text{und} \qquad S'(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.8}$$

The result of $u(p_i) = 2.97px$ converts to approx. 0.125mm and can be compared to standard deviations σ of measurements. That the Ziath CIS scanner has larger ALP dimensions compared to the CCD system explains the slightly worse results from Table 5.1. The measurements are attached to this work and can be found in Table A.1.

5.2.3 Multi Camera System

During development, only a single commercially available device was available for testing. To provide a more universal estimation of the mapping uncertainties in the system, an analysis was conducted here, too. This can also provide proof that the number of cameras does not influence mapping uncertainty.

The following analysis is confined to the transformations $D \to D_j$ for multi-camera setups. The transformation to the image plane, i.e. the uncertainty of the camera model is not covered here. Investigations of the used model can be found in [88] and in the literature Zhu et al. reference therein.

The investigated transformation is (3.18). Here, the placement of the registration plate has no influence on the translation vector $t_{dd,j}$, since it is a relative measure on the plate. The printer precision, however can have an influence and is set analogously to $u(t_{dl})$. The

Table 5.6: Uncertainty budget $L \rightarrow D_j$ transformation, x-axis

Symbol x_i	**Source of uncertainty**	**Value**		**Distribution**	**Divisor**	**$u(x_i)^a$**
p_d	See table 5.4	40	mm	normal	1	0.103
$t_{dd,j}$	Rel. trans. between checkerboards	-	mm	rectangular	$\sqrt{3}$	0.003
β	Rel. rot. between checkerboards	0	deg	rectangular	$\sqrt{3}$	0.1
$p_{d,j}$		-	mm	normal	-	0.11

^{a} in the unit of the respective value

estimation uses the following model:

$$u(p_{d,j})^2 = R(\beta)' p_{d,1} \cdot u(\beta)^2 + R(\beta) \cdot u(p_{d,1})^2 + u(t_{dd,j})^2 \tag{5.9}$$

The result of $u(p_{d,j}) = 0.11mm$ is approximately $2px$ on the image plane and can be compared to standard deviations σ of measurements. When compared to the results presented in Table 5.3, it is obvious that the very high mapping quality in the y-Axis is most likely not representative. A check of the ALP geometry revealed that it is not perfectly according to the standard dimensions. Table A.1 shows that the ABGene Dual-Camera system's ALP has an uncommon short long dimension (a).

5.3 Validation of Implemented Concurrency

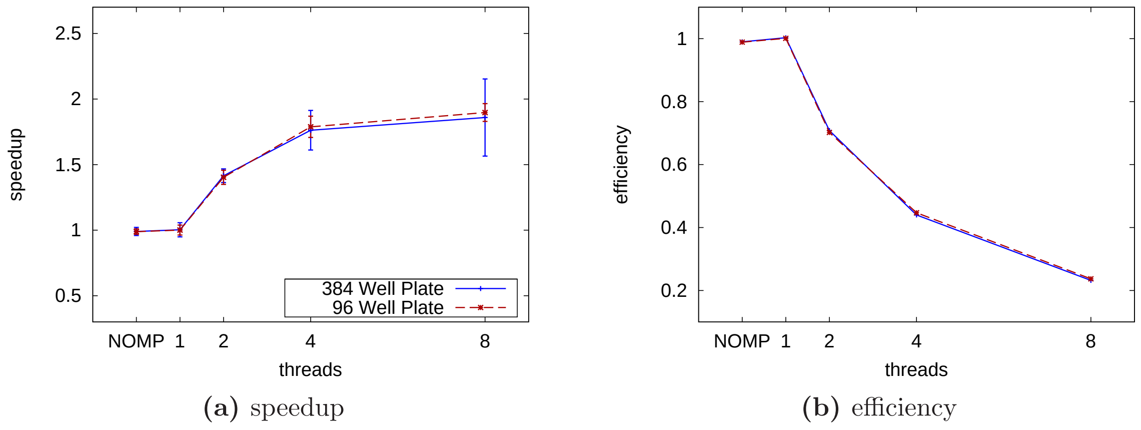
The concepts for load distribution are implemented using *OpenMP*. It uses a "fork-join" model that suits the base concept mentioned in Section 3.9 and shown in Figure 3.17, where an SIMD instruction is distributed over multiple threads. Nested SIMD instructions, as they appear when parallelizing an already parallel algorithm for multiple plates (Figure 3.18) are also supported by *OpenMP*.

However, implementation-specific limitations exist, since laboratory control computers usually are constrained to *Microsoft Windows XP* and *VC 8.0* and therewith to the *OpenMP* 2.5 specification. Here, only the outermost parallelization is conducted, while nested forking directives are ignored. When the outermost instruction can be divided such that all available resources are working to capacity, this is not a drawback. But, if only two plates shall be processed on an eight-core Windows XP system, six cores would not be used.

5.3.1 Validation

Test Setup

To test the performance gain obtained by parallelization, ten images of 384 and 96 well plates respectively were used. Using the CCD flatbed scanner, this results in a chunk size of $100x100px$ for every well for the 384-well plate, and $200x200px$ chunks for the 96-well plate. The system used for the test is an *Intel i7* based workstation, whose quad-core processor exposes eight threads to the operating system.[1]



(a) speedup **(b)** efficiency

Figure 5.2: Results well-wise partitioning

The test was conducted with five software versions with a different hardcoded number of threads each and without *OpenMP*. Parallel execution of a plate stack (partitioned in single plates) and of a single plate (partitioned in wells), were investigated. The reachable speedup was exemplarily investigated for two filter operations and a subsequent Hough transformation for circle detection. Hereby, the first filter operation is part of *preprocessing*, which is finished prior to the well mapping and hence a serial part (see Figure 3.20).

In the first step, twenty plates were processed with two parameter sets, with all five software versions. The execution times of the whole task and also of the parallelized parts were stored for the determination of the reached speedup. The second part investigates the performance gains with a partition on plate level that is possible during training, where a number of plates is processed. Here, a whole plate is assigned to a thread. Each plate is processed with up to 40 parameter sets. Subsequently, all results are collected

[1] using *Intel*'s *Hyper-Threading* Technology

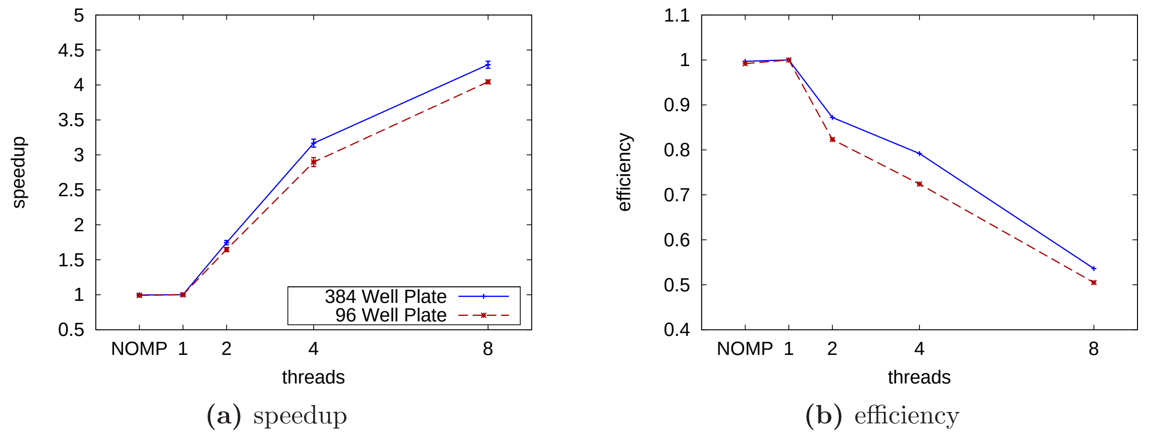


Figure 5.3: Results plate-wise partitioning

(serial part) and a performance value is calculated in a second parallel part. The last step compares the performance values and returns the optimum parameter set.

Results and Discussion

The raw results are attached to this work in the Appendix A.4. Table 5.7 displays the absolute times of the OpenMP-1-thread run together with the serial/parallel ratio for plate and well partitions. The time per well τ denotes the overall time divided by the number of wells. When comparing τ of both partitioning methods, one has to consider that the plate-wise partitions are used within a training run and that additional calculations are added in the serial part compared to the simple evaluation. Besides, the table lists the speedup reached with two threads.

Here, only two graphs shall be discussed. Figures 5.2 and 5.3 show the mean speedup together with its standard deviation (a) and the mean efficiency (b), relative to the OpenMP-1-thread with standard deviation σ for one, two and eight threads and the *no-OpenMP* version (NOMP). The graphs show that the 1-thread and the NOMP versions perform equally. This shows that OpenMP does not have a significant impact if used on a single core machine, where its not possible to take advantage of its threading capability.

Furthermore, it is obvious that the speedup increases with n_{thr} in a saturation curve. With well-wise partitions, σ grows for the 4- and 8-thread runs. A possible explanation is that the system is not able provide dedicated resources for more than three threads, because it has to run system services on one of the four cores besides the application.

During the tests, around 300 threads belonging to the operating system and system services were running in the background. This of course affects the 8-thread run as well, since the four additional threads are virtual. The very large σ of the 384-well run (with plate partitioning) is caused by a single very large outlier. The variations are equally large for plate and well partitions (ca. $\sigma = 0.12s$) but are not recognizable in Figure 5.3a because of the much higher absolute runtime.

When comparing reached performance with theoretical maximum values derived according to *Amdahl's Law*, it should be considered again that the system does not provide four dedicated processing cores. With two threads however, the performance can be compared. The ratio of parallel code is relevant for the determination of the maximum speedup. It is 99.6% for a training run that partitions a plate stack and only 62% when partitioning a plate. Preprocessing for example is done prior to the partitioning of the image data and is hence a serial part (see Figure 3.20). The different ratios lead to different maximum speedups according to *Amdahl's Law*:

$$speedup = \left(serial + \frac{parallel}{n_{proc}} \right)^{-1} \tag{5.10}$$

With 99.6% parallel code the maximum speedup with two cores is 1.99 while it is only 1.45 for 62%. With a speedup of 1.41, the theoretical maximum is nearly reached with well partitioning, while still being smaller than for the highly parallel stack processing (1.75). The relatively large serial part of the tested algorithm reduces the impact of parallel processing significantly.

The speedup for four cores is far from the theoretical values (3.17 to 3.96 and 1.76 to 1.86), but as explained above, this is most likely due to the other processes running on the system. For eight cores, the speedup compared to four cores is 10% respectively 25% and therewith on the lower side of the 15%-40% improvements possible with *Hyper-threading* technology [21].

Altogether, both partitioning strategies proved to work as expected. The stack partitioning (plate-wise) offers a high ratio of parallel code and is a good way to offer optimum performance during training. During operation however, only a single plate is present on the system and the plate has to be partitioned well-wise. Here, the efficiency depends largely on the design of the algorithm, and a conclusion is that preprocessing should be kept minimal - all operations that can be implemented for a sub-image should be implemented as such. Together with this design rule, the presented approach offers effective

Table 5.7: Results

	well-wise	**plate-wise**
Overall Exec Time 1-thread 96 wells (τ)	$0.49s$ $(5e-3s)$	$79s$ $(1e-2s)$
Overall Exec Time 1-thread 384 wells (τ)	$0.49s$ $(5e-3s)$	$61s$ $(7.8e-3s)$
Ratio parallel/overall	61.9%	99.6%
Speed-Up with $n_{thr}=2$	1.41(1.45)	1.7(1.99)

load distribution that does not require parallel programming of the application developer.

Chapter 6

Application Pipettor Monitoring

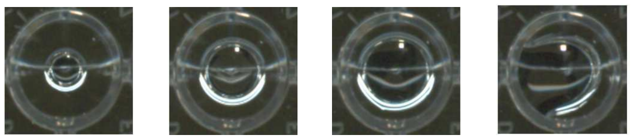
6.1 Motivation

Expensive and time-consuming processes force an increase in efficiency of the research industry. Together with the increasing need for documentation, this introduces the problem that pipetting robots operate blind. The operator provides reagents and substances for the experiment, and receives results in the form of value measurements only at the end of the experiment - when identifying the root cause for outliers is difficult and corrective actions are no longer possible. Since pipettors are very reliable today [27], a monitoring system must meet high requirements in terms of sensitivity and specificity in particular, in order to reduce false alarms.

This can pose difficulties in finding the problem that is responsible for the faulty result. It is furthermore economically disadvantageous to process a plate when it is faulty to begin with. The already deficient plate would eat up processing time on an expensive system and will also hinder a qualitative plate to pass on faster. For this reason it is a common business principle to have QC checks during an operation, especially prior to any bottlenecks of a process. [8] [53] The here presented approach to pipettor monitoring is based on CV and was implemented using the reference implementation of the framework.

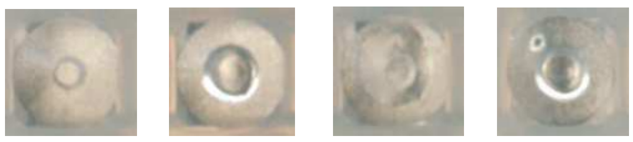
6.2 Drop Detection Concept

The detection of transparent objects has been a topic of previous research [7] [57] [61]. The authors of [57] evaluate distortions of a background texture to recognize transparent



(a) $3\mu l$ (b) $7\mu l$ (c) $24\mu l$ (d) $32\mu l$

Figure 6.1: Growing drop connecting to the wall between $20\mu l$ and $25\mu l$ in a 96-well plate



(a) empty (b) good visibility (c) bad visibility (d) satellite

Figure 6.2: Examples of $1\mu l$ drops in a VPP384 plate, taken with a CCD scanner

objects, similar to the approach by Litt et al. (see Section 1.2).

The authors of [61] use specular highlights to recognize shiny and transparent objects. A similar approach was chosen for the current task. The special challenge to detect drops in microtiter plate wells is that a used plate is translucent or transparent and produces specular highlights itself (see Figures 2.11, 6.1, 6.2 for examples). To suppress such artifacts caused by the plate, background subtraction is used in algorithms one and two (see Sections 6.2.1). A picture of an empty plate is taken for each plate type and subtracted from the actual plate. This technique cancels out the specular highlights of the plate.

To cope with a multitude of different microplates with different materials and hence luminance and transparency values, different well-bottom and well-border shapes, an adaptive algorithm is used that was inspired by cascade classifiers [75]. Cascade classifiers use a cascade of multiple weak classifiers to create a strong one.

Different hardware devices are supported by the framework and are tested for their suitability for the system. However, because the camera system proved to be too sensitive to ambient light in preliminary tests, the current investigation is limited to the two presented flatbed scanners. They picture a plate in ca. $10s$ at $600dpi$, still fast enough compared to classical approaches.

The drop detection is carried out by a two step procedure, which starts with a cascade of parallel algorithms. The three algorithms evaluate an image simultaneously and their eight outputs are weighted and merged by an Adaline [83] to generate the final output.

6.2.1 Algorithm Cascade

The pipettor monitoring application takes an image as input and detects drops in wells. It processes the image with a number of algorithms in parallel, and a final decision is subsequently computed from the outputs.

The application has to support a multitude of different plate densities and materials and the subsequent addition of new types has to be possible for users. These requirements on flexibility lead to the approach: The use of a-priori knowledge, to set parameters according to the labware type, and an automatic training for a plate type, to generate the parameters of the algorithm cascade. The automatic training includes furthermore the calculation of optimal weights for the fusion step. These weights are plate type dependent. This approach is inspired by cascade classifier approaches such as *AdaBoost*.

Algorithm 1

Detecting objects from comparison with the template image, this algorithm is called *BlobsFromTemplate()*. The difference image is binarized and further eroded and dilated iteratively to reduce noise. By using a circular structuring element, circular patterns are emphasized on the image. It consists of the following steps:

1. Calculate difference image $d(x, y) = g(x, y) - h(x, y)$ for all channels
2. Mask to remove noise
3. Thresholding: Binarization according to a plate specific threshold.
4. Morphological Opening:
5. The sum over all pixels equals the object size in px.

Algorithm 2

This detector uses the template image, too. However, it is possible to calculate the reference values from the template image during setup and before operation to save time. The algorithm compares calculated statistical moments of a well to these reference values, and outputs their differences. The higher the differences, the higher is the statistical probability that the image changed because of a drop in the well.

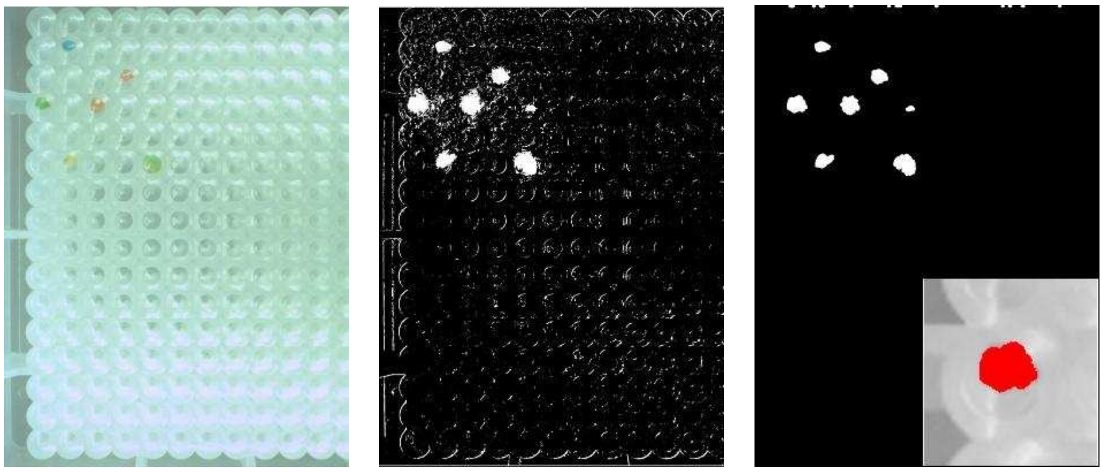

(a) Original (b) After thresholding (c) After morph. opening

Figure 6.3: Images between steps

A Sobel-based edge detection in direction of the lighting can emphasize reflections on drops as seen in Figure 6.4. A column profile is calculated from the area shown in the figure. From the one dimensional data, statistical moments are calculated.

The moments variance, skew and kurtosis are being used. Their difference is calculated and returned. The steps of the algorithm are summarized:

1. Edge detection with a Sobel operator
2. Mask to reduce noise
3. Calculation of column profile
4. Calculation of variance, skew, kurtosis
5. Difference
6. Output

Algorithm 3

A Hough transformation detects lines or curves in images. The implementation used here detects circular curves. Apart from drops, well borders can have a circular shape, too. In order to discriminate both shapes, a simple fuzzy inference system (FIS) with two input variables and two rules is used. The Hough transformation searches for circles with

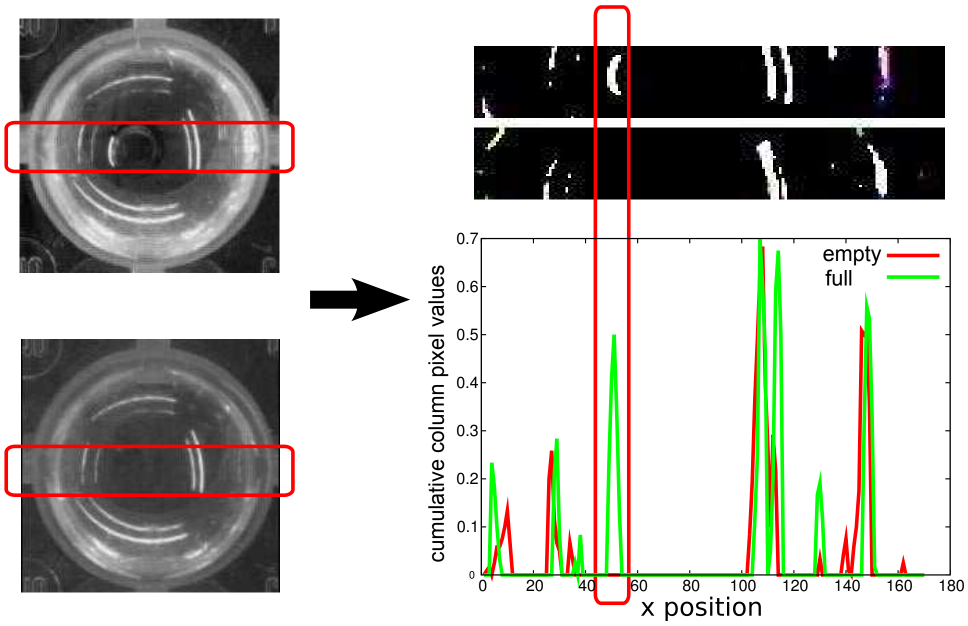
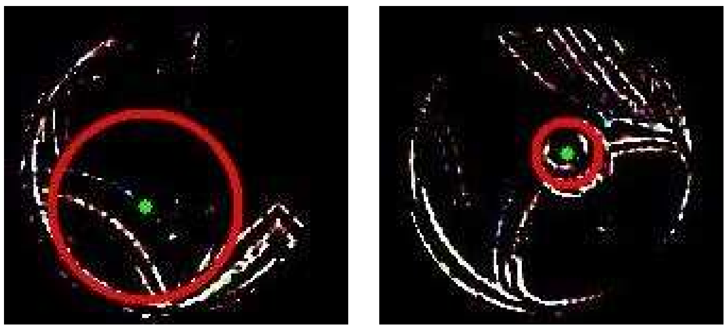


Figure 6.4: Calculation of the integral column profile

a radius in a specified range. If the detected circle is close to r_{max} it is likely that the algorithm found a well border. The system will tend to classify the circular shape as a well border when it is tangentially to the well border.



(a) Wall detected **(b)** Drop detected

Figure 6.5: LGH result pictures

A true drop is rather small and centered in the well. To adapt the algorithm to the used plate and pipetting technique, it is possible to set $[r_{min}, r_{max}]$ and train the rules. If for example the drop is placed near to the border using Tip-Touch[1], it is possible to

[1]A pipetting technique where the pipettor moves the tips along the wall to make use of cohesion. That way it is possible to pipette smallest amounts that otherwise would remain on the tip.

decrease the weight of the location of the circular shape. The rule set is the following:

$$IF \quad r = small \qquad AND \qquad d_t = small \qquad THEN \qquad y_{LH,1} = small \tag{6.1}$$

$$IF \quad r = r_{erw} \qquad AND \qquad d_t = large \qquad THEN \qquad y_{LH,1} = large \tag{6.2}$$

The labels “small“ and “large“ represent plate-specific values and the awaited drop radius r_erw can be set according to the commanded volume. However, the values are set automatically during training. The fuzzy inference is integrated into the inference step and trained globally.

1. Edge-detection using a Laplacian-of-Gaussian (LoG) Kernel (Size: $5x5$, $\sigma = 1,4$).
2. Mask to reduce artifacts
3. Hough transformation detects circular pattern.
4. FIS using two inputs and rules
5. Output

6.2.2 Inference

The calculation of a decision is done by a linear fuzzy inference element (Adaline)[83]. Its inputs are the algorithm outputs for every well of the plate. Their eight outputs x_i are weighted with β_i^* and merged to generate the output y:

$$y = \sum_i \beta_i^* \cdot x_i, \text{ where } \beta_i^* = \frac{\beta_i}{\sum_i \beta_i}. \tag{6.3}$$

6.3 Detector Training

As pointed out in Section 2, training should be a ”one-click” solution not needing specialized experience. Plates have to be prepared and acquired for training. They are captured beforehand with the administrator tool. Subsequently, the positions and volumes of drops have to be declared by the user before the run.

The first step of the training run is to find optimal algorithm parameter sets for the algorithms described in Section 6.2.1 using the framework’s grid search routine. To find the optimal parameter set θ_{opt}, we use the target function

$$|AUC(\theta) - 0.5| \to \max . \tag{6.4}$$

It considers $AUC = 0.5$ as the worst value [26]. Then, the Adaline weights β are found by least-squares estimation. The last step is to find an optimal threshold t_{opt} where

$$(TPR(t) - FPR(t)) \to \max . \tag{6.5}$$

The user accepts a found setup by a confusion matrix that notes true positives (TP) and false positives (FP) .

6.4 Detector Validation Tests and Conclusions

6.4.1 Development Tests

Test Setup

Each device was set up on the robot platform for testing. Fluids were delivered with the pipetting system. The plates were then transferred to each optical system; an image was taken of each plate with covering (CIS, camera) and without (CIS, CCD, camera). The camera system was not included for evaluation for the reasons stated above.

To generate a representative amount of test data, the plates were used several times, accumulating amounts of substance inside. Two pipetting approaches were used for the examinations. For the 96-well plates a certain volume was dispensed to each of the 8 wells in one row whereas the following row was left empty (6 rows filled, 6 rows empty). For the 384-well plate a certain volume was dispensed to every second well (checkerboard pattern). These approaches were selected in order to measure the droplets spread over the plate. There were also empty wells distributed across the plate for position effects to be detected as well. Dosages of $1 - 40\mu l$ were taken for the 96-well MTPs, but only $1 - 12\mu l$ max. dosage for the 384-well plates due to the smaller well volume. Tighter steps were used for $1 - 8\mu l$ dosages to generate more test data. The algorithms developed were used afterwards on the data collected.

The tests were run on 96-well polystyrene (PS) and polypropylene (PP) plates with V-bottoms (V), a 96-well polystyrene plate with a flat bottom (F) and a 384-plate polypropylene plate with a V-bottom. In the following, the plates are being referred to by the above abbreviations. Images were taken in three to four runs each, for four plates in eight to twelve volume steps in three types of setup: CCD, CIS with, and CIS without covering. In total, more than four hundred test images were taken.

Results

Table 6.1 summarizes the true and false positives for the plates examined for each device. Wide-ranging examinations were performed on system sensitivity to various influences in order to gauge the usefulness of the system in real-life conditions.

Table 6.1: Results in percent

	CIS open		**CIS lid**		**CCD open**	
	TP	FP	TP	FP	TP	FP
VPP96	99.54	0.00	96.05	0.00	95.21	0.00
VPS96	97.80	0.00	86.69	0.00	96.64	0.35
FPS96	88.75	1.04	86.31	5.36	72.57	2.95
VPP384	94.17	7.50	96.46	4.24	95.35	0.56

Influence of resolution on efficacy

The tests were performed at a maximum resolution of $600dpi$ ($0.042mm/px$). Comparative tests on lower resolutions of $300dpi$ ($0.085mm/px$) also yielded good results. Lowering resolution in the volume ranges tested above $1\mu l$ showed no significant negative effect.

The system's measurement range

There are upper and lower volume limits for detecting droplets. The lower limit is subject to several factors, such as light distorting the image in the middle of V-bottom wells at the cylinder tip or V-tip, and the droplet needs to be larger than the bottom tip. For all plates the lower limit was found to be below $1\mu l$.

Figure 6.1 shows how droplets tend to stick to the walls of a well in recognizably increasing numbers when growing in size. This reduces the TP rate for larger volumes, especially with a CIS scanner (please see Appendices B.1.1 and B.1.2), since the reflections disappear, and hence leads to the upper boundary of the measurement range. The ideal measuring range for 96-well plates is between 1 and $24\mu l$, while volumes of up to $6\mu l$ are recognizable in 384-well plates with $> 0.9TPR$. Figure 6.6 shows the degradation of measurement quality with two different plates.

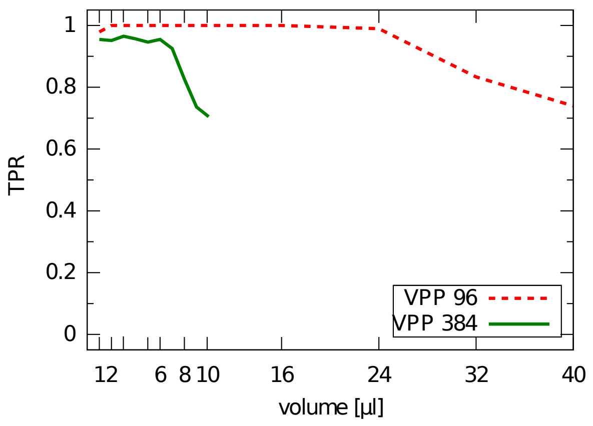


Figure 6.6: Degradation of measurement accuracy with growing sample volume

Operation under changing lighting conditions

The system would be used in a variety of conditions in real-life operation; ambient light reaching the plate varies depending on the system's location as well as the season and time of day, so the system would have to be tested for robustness with regard to light conditions. To test this, test images were taken in three different typical lighting situations as follows: 1. Full direct sunlight on the device, 2. Indirect daylight, 3. No natural lighting - lab lighting. Two images were taken for each run using the CIS sensor to test its particular sensitivity towards ambient light. Each plate was imaged once with, and once without the covering (microtiter plate lid) in each ambient light setting and with every volume. The lid shields the plate from ambient light during the imaging process.

Fig. 6.7 shows an example of how direct sunlight ruined the image. Similar overexposure also affected images made using the CCD sensor, albeit less strongly, since the CCD sensor is less sensitive to ambient light. Overall, no images taken with the CCD sensor were affected by direct light such that they became useless.

The results for the CIS scanner with lid are represented by the hashed area in Figure 4. The solid area includes results without the lid. The figure shows the improved relationship between true positives against false positives on the one hand, and the smaller fluctuation

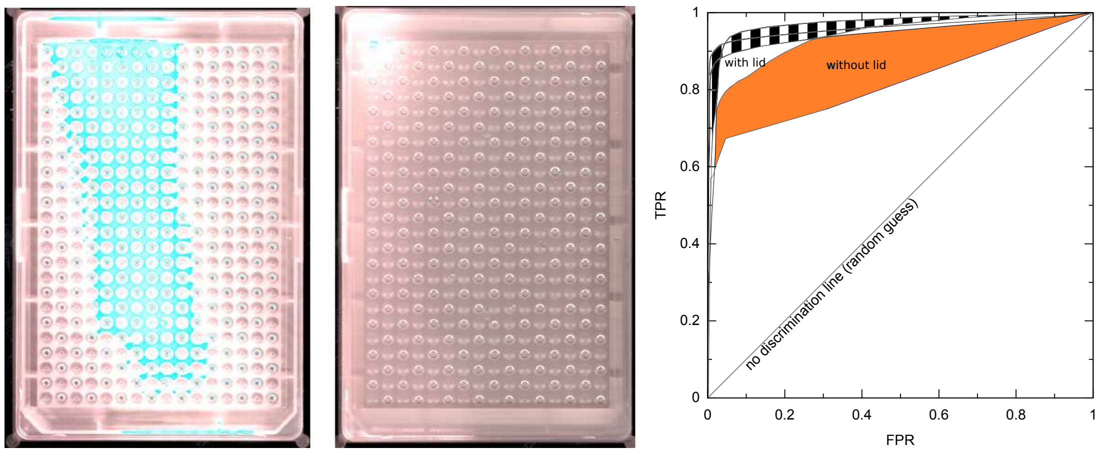


Figure 6.7: Overexposure without lid, with lid and results, V-PP-384 (CIS)

range on the other, using the lid. The benefits of covering therefore consist of improved results and robustness to ambient light. The smaller fluctuation range is equivalent to a smaller area on the TP/FP plane. The bounded area $\Delta AUC_{o,CIS}$ was 0.132 without covering compared to $\Delta AUC_{l,CIS}$ at 0.018 with covering (lid)- only around 13.6%. The CCD sensor yielded more robust results: The difference in area $\Delta AUC_{o,CCD}$ was 0.034 for the CCD sensor, 25% of $\Delta AUC_{o,CIS}$.

Influence of sample substance and pipetting technique

Fluids with varying levels of viscosity and surface tension are dosed using different pipetting techniques. If the cohesion effects between the fluid and pipette are too strong, as it is the case with dimethyl sulfoxide (DMSO), the droplets have to be placed into the well using techniques such as Tip-Touch. Viscosity also affects the way samples of a given volume spread across the well.

One test addressed functionality with filtered water and pure DMSO. The ideal pipetting technique for water was used for the droplets of the same shape to be placed into the middle of the well at a very high level of reproducibility. DMSO was pipetted using Tip-Touch[1], which is not ideal for the substance; this increased the negative influence of droplets clinging to the well edges. Dosing DMSO also showed an increasing frequency of off-center droplets at increasing volume, suggesting an operational range depending on substance. Fig. 6.8 shows how a bad pipetting technique leads to uncentered drops clinging to the walls. As described above, this leads to a decrease in measurement accuracy. However, this decrease is not as severe as the influence of the scanner type: Comparing

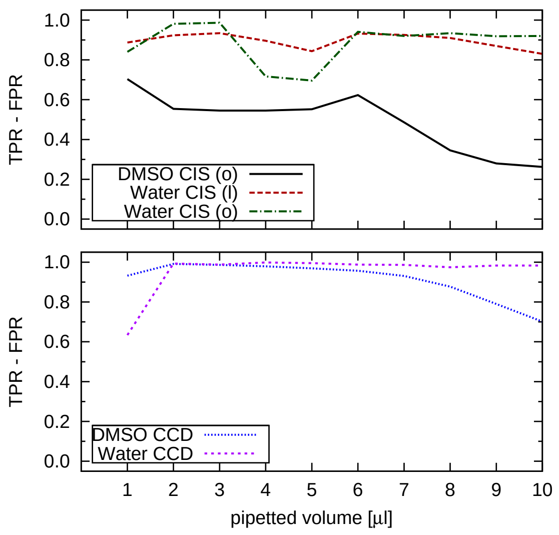


Figure 6.8: Influence of pipetting technique, top: DMSO, bottom: water

the results of CCD and CIS systems in Figure 6.8, it is clearly visible that CIS sensors are not robust and not suitable to detect DMSO. The accuracy decreases earlier and more rapidly for DMSO than it does for water using either sensor type.

Conclusion

The system's functionality has been confirmed in this test sequence. Using a basic flatbed scanner simplifies application for pipetting monitoring and improves results while increasing efficiency compared to the camera solution used up to now.

The pipetting monitoring presented can be implemented using barcode readers based on CCD or CIS flatbed scanners that the lab may already have at its disposal. Efficacy with regard to sample volume and substance properties as well as robustness to ambient light was tested. Depending on lab lighting conditions, a covering may improve results.

The results of the tests taken up to now have revealed a variety of possible improvements for the system. Result quality showed heavy dependence on the algorithm parameters used, so automatic parameter optimization is implemented for the reliable detection of the best parameters. Additionally, this enables the user to configure the device for new labware on his own, because all settings can be found automatically.

In addition, adaptation to ambient light should be optimized to ensure that the system can be used without covering wherever possible. There is further room for improvement at high volume ranges or for badly positioned droplets; the potential for quantitative volume estimation from dimensional droplet projections, fluid properties and well shape warrant further investigation.

6.4.2 Qualitative and Quantitative Validation

Test Setup

Although the pipettor monitoring is designed for the use with multi-channel pipettors, the test was conducted with a *HTC PAL* (*CTC Analytics*, *Zwingen*, *Switzerland*) using a $10\mu l$ syringe with a precision of < 1 % relative standard deviation (RSD). Two labware types were tested: A flat-bottom (F), polystyrene (PS) 96-well plate and a V-bottom, polypropylene (PP) 384-well plate.

Three plates of each type were pictured with a CCD and a CIS scanner, covered with a lid (l) and open (o), for volumes between 1 to $10\mu l$ (384 wells) and 1 to $25\mu l$ (96 wells). Only 6 rows of each plate were pipetted to reduce the time before imaging and hence reduce evaporation. The preparation of a 384-well plate took ca. 10 minutes. The first of three sets was used for training, the other two were evaluated.

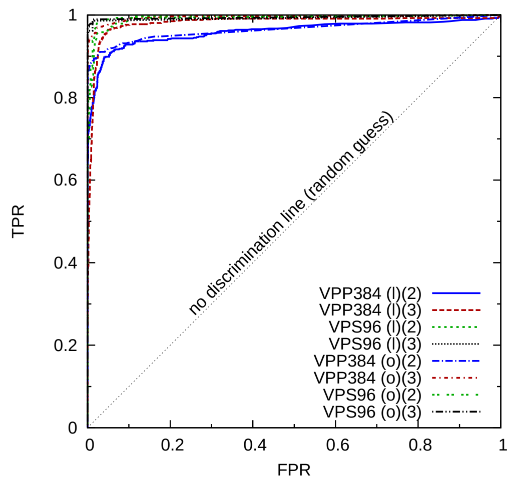


Figure 6.9: Results using a CCD scanner

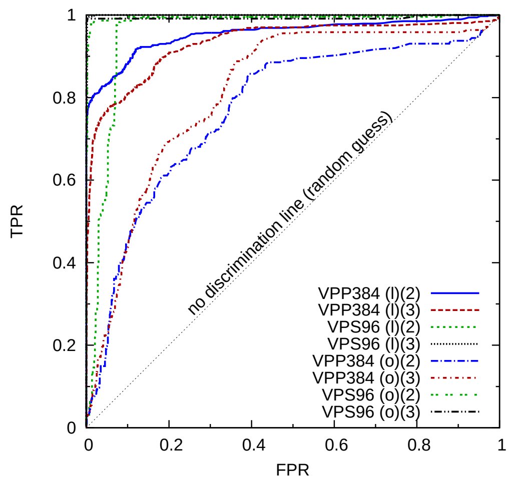


Figure 6.10: Results using a CIS scanner

Results

The qualitative results are shown in Fig. 6.9 and Fig. 6.10. Overall, the CIS scanner performed best (FPS96-3: $AUC = 0.994$) but also worst (VPP384-3: $AUC = 0.772$).

The large differences of two datasets of the same plate/device combination are a measure for the low robustness of CIS scanners. The low robustness compared to the CCD system was expected from our previous test, because CIS scanners use a weaker light source and hence ambient light accounts more to the lighting of a scene.

The worst run using the CCD scanner was near ($TPR = 0.9$, $FPR = 0.1$, $AUC = 0.965$). The modified training procedure accounts to the fact that the CCD system performance during uncovered operation is not categorically worse when compared to covered operation. The variation of ROC curves of two similar datasets is smaller compared to the CIS system, which is a sign for higher robustness compared to the CIS system. With the V-bottom 384-well plate, the CCD scanner outperformed the CIS scanner.

Furthermore, the system's capability to quantify drop volumes was examined. Figure 6.11 shows the first detector's output and its standard deviation over the volume that was commanded to the liquid handler as reference. In addition, a fitted curve $a\sqrt{x/b}+c$ is shown, that was later used to calculate volumes from the outputs for the validation datasets (second graph of Fig. 6.11).

Below $5\mu l$, the conversation formula provides appropriate results (ideal values would lay on the dotted line with a slope of 1). With the 384-well plate, the slope of the curve has a maximum above $5\mu l$. This corresponds with our finding in [70], where the increased clinging to the wall with growing drop size (see Fig. 6.1) was identified as a cause for a decrease in detector performance. Evaluated quantitatively, this effect causes too low values.

6.4.3 Improvements to the Drop Detection Algorithm

Adaline Training

Two areas were found during validation where the algorithm can be further improved. The first issue is the performance of the inference step when one algorithm clearly outperforms all others. The cascade-classifier approach to build a strong classifier by combination of weak ones could be observed to work. It was recognizable from ROCs like drawn in Figure 6.12.

However, when a single descriptor clearly outperforms all others, the used approach does not deliver optimum performance anymore. By using the algorithm tuning described in Section 6.3, it was often the case that the first descriptor could be improved so much that its performance was much better compared to the other two. During the tests, it

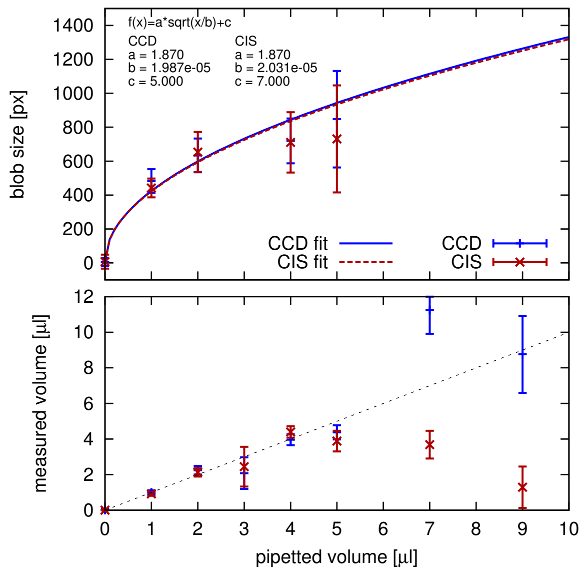


Figure 6.11: Quantitative evaluation of the first detector

became apparent that the chosen least-squares based Adaline training does not deliver optimal performance in such cases, as it reduces overall performance as in the case shown in Figure 6.13. This is a clear drawback of the system, since it is obvious that weighing algorithms 2 and 3 with zero would improve outcome r to be equal to y_1.

The system was originally developed with a camera system in mind that was discarded later in the process, when flatbed scanners showed superior suitability for the task. The results further show that the first algorithm performed best with both flatbed scanners. As pointed out above, this can lead to suboptimal system settings. To improve its behavior, the training algorithm should be investigated and other approaches should be tested.

Robustness against ambient lighting

Additionally, it is possible to take action to further improve descriptor 1. Its cancellation of surrounding objects and lighting effects by template subtraction relies on the fact that the template and the actual image are equal except for the subject of measurement. This assumption is never fully met due to interfering factors that comprise:

- Placement of every plate within the position tolerances of the robot
- Labware manufacturing tolerances or type-differences
- Lighting changes

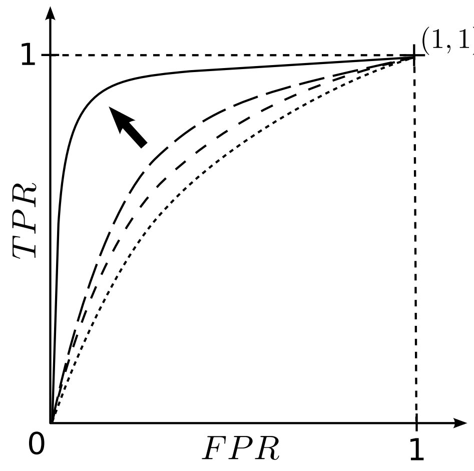


Figure 6.12: Weak descriptors build a strong one (weak: broken lines)

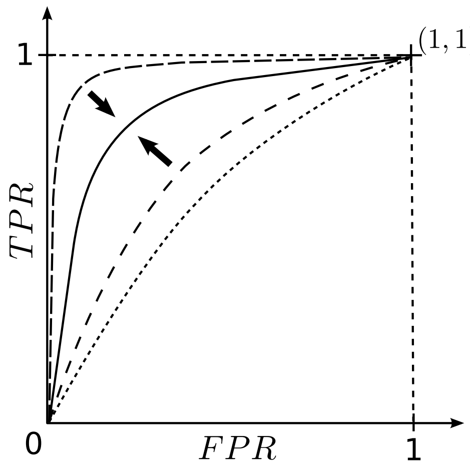


Figure 6.13: Performance is lost due to the inference step (weak: broken lines)

The first factor is not avoidable, but a possibility to minimize its translational portion was mentioned in Section 3.3.2. As long as labware of the same type is used, the second point can be considered insignificant. This leads to the requirement that a new template image must be recorded for every new plate, even when the plate type is the same from the experimental perspective.

The third factor was minimized by using a lid to cover the plate from ambient lighting. Here a different approach can be chosen that allows open operation. When regarding the artificial light to be constant over time (i.e. the lights in the lab are turned on during working hours), the differences between two lighting situations $\Delta g(x, y)$ mainly result from daylight changes. Relative to the time of a run, this can be regarded as being minimal.

To make use of this fact, it may make sense to operate the liquid handler directly on the visual labware position, which is possible due to its open design. The operation would be changed such that an empty plate is moved to the imaging position for dispensing. Before pipetting, a template image is taken for every run. It is possible to acquire the first image while the pipettor still aspires liquid, hence this procedure may even be faster compared to lid By doing so it is possible to minimize the lighting changes and improving outcome of all template based descriptors. The validity of this approach was already tested manually. Needed prerequisites are summarized:

- The imaging position must be reachable by the pipetting head.

- Depending on the experiment, multiple imaging devices could be required, e.g. when multiple plates need to be prepared at the same time.
- Controlled artificial lighting must be ensured during operation (lab lighting).
- The additional time required for the second image should be considered.

6.5 Integration into the Framework

The designed algorithm was integrated into the framework as proposed in the concept in Figure 3.1. The input is a sub-image showing a single well and two values per well can be returned by the application. The first value is the binary decision, the second would be the quantitative guess if it is requested. The results are returned as a SAMI data message, which can be utilized by modules that are set up appropriately.

Chapter 7

Conclusions and Future Work

7.1 Framework

7.1.1 Conclusions

The developed concepts for a CV-framework for laboratory automation proved to be a valid approach: One specific characteristic of totally automated laboratories is the use of labware that provides high throughput through parallelization. The need for parallel execution of CV algorithms arises, and was successfully catered for. The application developer creates a standalone algorithm, whereby the framework takes care of all needed prerequisites for its integration, which comprises interfacing to multi-sensor devices, calibration of cameras or flatbed scanners and labware mapping.

A suitable calibration routine for flatbed scanner devices was not found in the literature. The thereupon developed approach proved its feasibility. Flatbed calibration results in reprojection errors in the range of 3.4 to 4.3px, with standard deviations of ca. 2px. The uncertainty analysis supports this finding. For cameras, the implemented transformations result in an overall reprojection error of ca. 4px, equivalent to 0.2mm. The standard deviation is within the range of the estimated uncertainty.

While a large portion of research covering multi-sensor devices deals with stereo vision, devices in laboratory automation use camera grids to increase the resolution of the regions of interest, only superimposing in small regions at the image borders. Because of that, available registration procedures are not suited for their registration. A straightforward multi-sensor registration approach was hence introduced and has already proven its validity. One limitation of the current implementation is that a sample position is not allowed to be distributed over images. This means that a dual-camera system is not yet compatible to one well labware.

Calibration and registration are implemented into a graphical user interface that is adaptable to the application. This interface can further be used to run the implemented grid search algorithm optimization and to test and setup a device. The parallelism of the laboratory ("parallel samples") is used in a similar way for data partitioning to implement multi-processor-compatibility. Using two cores, multithreading speedup is at 1.41 for well-wise partitioning of a plate, reaching 97% of the maximum value according to Amdahl's law. For plate-wise partitioning of a plate stack, the speedup is at 1.7 (85%).

The framework's PCS interface was exemplarily linked to a proprietary, state-of-the-art control system. The reactions on measurement results can be defined by the setup of other connected device modules.

7.1.2 Future Work

Sample Mapping

Currently, samples are not allowed to be spread over multiple images. By finding corresponding regions on the borders of images and by correcting their perspectives, it is possible to stitch images together. The stitched images can then be passed on to the image processing algorithm. This would make devices with many cameras compatible to low-density plates (e.g. a two-well plate with an eight camera system). The current implementation chooses the camera with the best perspective (well near image center) from the lookup table for further processing. Instead, it is possible to pass more than one image for every well to the algorithm. Then it is possible to evaluate a specific sample from different perspectives (from multiple cameras). Herewith, perspective errors could be reduced and even three-dimensional information can be calculated. Finally, Appendix A.6 shows an excerpt of the labware calibration test. The algorithm has to be modified so that it runs reliably on all wells, and two areas of application should be investigated: Calibration of the complete optical model or fine-tuning of a prior calibrated system to map wells more accurately.

CCD Flatbed Scanner Model

The test of CCD flatbed scanner image distortions showed an oscillating distortion in the drive axis and an s-shaped distortion orthogonal, similar to the literature. Here, future work could address automatic calibration of the nonlinear effects of flatbed scanners for automation systems, to establish a common method similar to the available camera calibration method. The first step will be to investigate a set of devices regarding their distortions and how much they resemble the behavior shown in Figures A.2 and A.3. In case that a generic rule can be derived, it should be implemented in an automatic calibration routine and be used in the system as such.

Multithreading

The multithreading routines could make use of automatic sizing of the thread pool and thread priority should be considered.

7.2 Pipettor Monitoring Application

7.2.1 Conclusions and Future Work

The pipettor monitoring application was validated successfully with two comprehensive tests. While the qualitative drop detection works with sufficient performance, more work must be carried out to enhance the quantitative measurement.

Two areas are presented in Section 6.4.3 that would benefit much from future work. This includes the software calibration (training) of the inference step, where the used algorithm can lead to suboptimal results under certain circumstances, i.e. when a single weak descriptor is much better than the others. And it furthermore includes the use of per-plate templates to limit the influence of changing ambient conditions. This is done by reducing the time between two image acquisitions. With cameras, the time between the two images could be reduced further, resulting in a stream of images showing the dispensing process. This would make the use of established background-subtraction algorithms possible. Future research should investigate whether cameras can compete with flatbed scanners using this approach. The aim should be to measure the volume change over time ($\Delta V(t)$). Using this information, erroneous dispensing could be found by extrapolation.

The support of multi-camera systems paves the way to find out how multiple cameras can be used to determine a liquid level in a sample vessel. The basic idea here is that two cameras are required to determine the position of a point in 3D space. This approach should be investigated in the future.

Bibliography

[1] E. E. Anderson and W. Wang, "Novel contact image sensor (CIS) module for compact and lightweight full-page scanner applications," in *Cameras, Scanners, and Image Acquisition Systems*, H. Marz and R. L. Nielsen, Eds., vol. 1901. San Jose, CA, USA: SPIE, May 1993, pp. 173–181.

[2] J. P. Andrews, C. V. O'Keefe, B. G. Scrivens, W. C. Pope, T. Hansen, and F. L. Failing, "United states patent: 6043880 - automated optical reader for nucleic acid assays," Mar. 2000.

[3] M. Arhoun, "An automatic system for stepwise treatment of solid samples and application to pollution evaluation by measuring ion lixiviation rates in lichens," *Laboratory Robotics and Automation*, vol. 11, no. 2, pp. 121–126, Jan. 1999.

[4] A. Azarani, B. W. Segelke, D. Toppani, and T. Lekin, "High-Throughput protein crystallography," *Journal of the Association for Laboratory Automation*, vol. 11, no. 1, pp. 7–15, Feb. 2006.

[5] P. Baillargeon, L. Scampavia, R. Einsteder, and P. Hodder, "Monitoring of HTS compound library quality via a High-Resolution image acquisition and processing instrument," *Journal of Laboratory Automation*, vol. 16, pp. 197–203, Jun. 2011.

[6] W. Baumann, "Einfuhrung in die parallele programmierung mit MPI," Zuse Institut Berlin, 2006.

[7] J. Beck, "Perception of transparency in man and machine," *Computer Vision, Graphics and Image Processing*, vol. 31, no. 2, pp. 127–138, 1985.

[8] J. Blackstone, "Theory of constraints - a status report," *International Journal of Production Research*, vol. 39, no. 6, pp. 1053–1080, 2001.

[9] H. Bothe, *Neuro-fuzzy-methoden.* Springer, 1998.

[10] G. Bradski, “The OpenCV library,” *Dr. Dobb’s Journal*, pp. 120–125, Nov. 2000.

[11] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O’Reilly Media, Inc., Oct. 2008.

[12] D. Brown, “Close- range camera calibration,” *Photogramm Eng*, vol. 37, no. 8, pp. 855–866, 1971.

[13] P. J. Bushway, M. Mercola, and J. H. Price, “A comparative analysis of standard microtiter plate reading versus imaging in cellular assays,” *Assay and Drug Development Technologies*, vol. 6, no. 4, pp. 557–567, Aug. 2008, PMID: 18795873.

[14] J. Canny, “A computational approach to edge detection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[15] P. Courtney, M. S. Beck, and W. J. Martin, “A vision guided life-science laboratory robot,” *Measurement Science and Technology*, vol. 2, no. 2, pp. 97–101, 1991.

[16] C. Cowan and P. Kovesi, “Automatic sensor placement from vision task requirements,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 3, pp. 407–416, 1988.

[17] C. Cowan and B. Modayur, “Edge-based placement of camera and light source for object recognition and location,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1993, pp. 586–591 vol.2.

[18] F. Devernay and O. Faugeras, “Straight lines have to be straight,” *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[19] D. Dossenbach and S. Jess, “Wenn der μ-Liter eine rolle spielt,” *Mikrofluidik*, vol. 2009, 2009.

[20] R. O. Duda and P. E. Hart, “Use of the hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[21] J. Duffy, “Using concurrency for scalability,” *MSDN Magazine*, Sep. 2006.

[22] F. Echtler, “Tangible information displays,” Ph.D. dissertation, Technische Universität München, 2009.

[23] A. Erhardt, *Einführung in die Digitale Bildverarbeitung: Grundlagen, Systeme und Anwendungen*, 1st ed. Vieweg+Teubner, 2008.

[24] European Union, “European union GMP annex 11 computerised systems,” Jun. 2011.

[25] ——, “European union GMP chapter 4 documentation,” Jun. 2011.

[26] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[27] M. Felton, “Liquid handling: dispensing reliability.” *Analytical chemistry*, vol. 75, no. 17, 2003.

[28] C. for Disease Control and P. (CDC), “Serum cross-reactive antibody response to a novel influenza a (H1N1) virus after vaccination with seasonal influenza vaccine,” *MMWR. Morbidity and Mortality Weekly Report*, vol. 58, no. 19, pp. 521–524, May 2009.

[29] H. G. Fouda, “Robotics in biomedical chromatography and electrophoresis,” *Journal of Chromatography B: Biomedical Sciences and Applications*, vol. 492, pp. 85–108, Aug. 1989.

[30] K. A. Giuliano, R. L. DeBiasio, R. T. Dunlay, A. Gough, J. M. Volosky, J. Zock, G. N. Pavlakis, and D. L. Taylor, “High-Content screening: A new approach to easing key bottlenecks in the drug discovery process,” *J Biomol Screen*, vol. 2, no. 4, pp. 249–259, Jun. 1997.

[31] J. Hanley and B. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, Apr. 2004.

[33] T. Hartley, C. Stewart, R. Stewart, and D. Munroe, “Cost-Effective addition of High-Throughput colony picking capability to a standard Liquid-Handling platform,” *JALA - Journal of the Association for Laboratory Automation*, vol. 14, no. 1, pp. 22–26, 2009.

[34] W. B. He, M. S. Beck, and W. J. Martin, "Processing of living plant images for automatic selection and transfer," *Computers and Electronics in Agriculture*, vol. 6, no. 2, pp. 107–122, Oct. 1991.

[35] G. E. Hoffmann, "Concepts for the third generation of laboratory systems," *Clinica Chimica Acta*, vol. 278, no. 2, pp. 203–216, Dec. 1998.

[36] C. E. Hooper, R. E. Ansorge, and J. G. Rushbrooke, "Low-light imaging technology in the life sciences," *Journal of Bioluminescence and Chemiluminescence*, vol. 9, no. 3, pp. 113–122, 1994.

[37] D. Hopper, "The long perspective for robotic vision," *Assembly Automation*, vol. 29, no. 2, pp. 122 – 126, 2009.

[38] R. Horaud, R. Mohr, and B. Lorecki, "On single-scanline camera calibration," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 71–75, 1993.

[39] V. Hough and C. Paul, "United states patent 3069654: Method and means for recognizing complex patterns."

[40] ISO, "Programming languages – c++," ISO International Standards Organization, Standard ISO/IEC 14882:2003, 2003.

[41] B. Jähne, *Digitale Bildverarbeitung*, 6th ed. Springer, Berlin, Apr. 2005.

[42] P. Jasiobedzki and W. J. Martin, "Processing of bacterial colony images for automatic isolation and transfer," *Journal of Physics E: Scientific Instruments*, vol. 22, no. 6, pp. 364–367, 1989.

[43] Joint Committee for Guides in Metrology, "Guide to the expression of uncertainty in measurement," JCGM, Tech. Rep. JCGM 100:2008, 2008.

[44] P. Jones, A. Watson, M. Davies, and S. Stubbings, "Integration of image analysis and robotics into a fully automated colony picking and plate handling system," *Nucl. Acids Res.*, vol. 20, no. 17, pp. 4599–4606, Sep. 1992.

[45] S. Junginger, "Automated cell culture," Ph.D. dissertation, Universität Rostock, 2011.

[46] J. Kangasrääsiö and B. Hemming, "Calibration of a flatbed scanner for traceable paper area measurement," *Measurement Science and Technology*, vol. 20, no. 10, p. 107003, Oct. 2009.

[47] J. E. Kelly, D. C. Jones, F. R. Tuck, W. A. Zimmermann, and K. R. Clark, "United states patent: 4710031 - microtiter plate reader," Dec. 1987.

[48] H. I. Kim, S. H. Lee, and N. I. Cho, "Automatic defect classification using frequency and spatial features in a boosting scheme," *Signal Processing Letters, IEEE*, vol. 16, no. 5, pp. 374–377, 2009.

[49] U. Kolukisaoglu and K. Thurow, "Future and frontiers of automated screening in plant sciences," *Plant Science*, vol. 178, no. 6, pp. 476–484, 2010.

[50] J. Lebak, J. Kepner, H. Hoffmann, and E. Rutledge, "Parallel VSIPL++: an open standard software library for High-Performance parallel signal processing," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 313–330, 2005.

[51] Y. Ling, T. Mullen, and X. Lin, "Analysis of optimal thread pool size," *SIGOPS Oper. Syst. Rev.*, vol. 34, no. 2, pp. 42–55, Apr. 2000, ACM ID: 346320.

[52] G. J. Litt, "United states patent: 4824230 - visualization device," Apr. 1989.

[53] V. Mabin and S. Balderstone, "The performance of the theory of constraints methodology: Analysis and discussion of successful TOC applications," *International Journal of Operations and Production Management*, vol. 23, no. 5-6, pp. 568–595, 2003.

[54] E. Marchand, "Control camera and light source positions using image gradient information," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 417–422.

[55] R. McDowall, "Is GMP annex 11 europe's answer to 21 CFR 11?" *Spectroscopy*, vol. 26, no. 4, 2011.

[56] F. J. McFadden, "United states patent: 4221867 - optical microbiological testing apparatus and method," Sep. 1980.

[57] K. McHenry, J. Ponce, and D. Forsyth, "Finding glass," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 2005, pp. 973–979 vol. 2.

[58] J. M. McMillan, “United states patent: 6057163 - luminescence and fluorescence quantitation system,” May 2000.

[59] M. Ohta, N. Ozawa, and Y. Yokomori, “United states patent: 5169601 - immunological agglutination detecting apparatus with separately controlled supplementary light sources,” Dec. 1992.

[60] Open Source Initiative (OSI), “The BSD license.”

[61] M. Osadchy, D. Jacobs, and R. Ramamoorthi, “Using specularities for recognition,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 1512–1519 vol.2.

[62] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 66, 62, Jan. 1979.

[63] J. Parkhurst, J. Snider, and M. Williams, “SILAS™ integration software for laboratory automation,” Beckman Coulter, Inc., Tech. Rep. T-1856A, 1998.

[64] F. Pedrotti, L. Pedrotti, W. Bausch, and H. Schmitt, *Optik für Ingenieure: Grundlagen*, 3rd ed. Springer, Apr. 2005.

[65] F. Pla, J. Sanchiz, and J. Sanchez, “An integral automation of industrial fruit and vegetable sorting by machine vision,” in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, vol. 2, 2001, pp. 541–546 vol.2.

[66] J. Pochert, “Liquid level detection in high-throughput screening applications,” *International Biotechnology Laboratory*, no. Okt. 2000, p. 44, 2000.

[67] G. Porter, “Miniaturized assays magnify pipetting issues,” *American Biotechnology Laboratory*, vol. 20, no. 2, p. 71, 2002.

[68] I. Pyarali, M. Spivak, R. Cytron, and D. C. Schmidt, “Evaluating and optimizing thread pool strategies for Real-Time CORBA,” in *Proceedings of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems*, ser. OM ’01. Snow Bird, Utah, United States: ACM, 2001, pp. 214–222, ACM ID: 384226.

[69] J. Raissi, "Performance impact of thread pooling in DSOCARE," in *Proceedings of the IEEE SoutheastCon, 2006*. IEEE, Apr. 2005, pp. 108–113.

[70] K. Ritterbusch and K. Thurow, "Optical detection of the liquid level in microplates," *Technisches Messen*, vol. 77, no. 12, pp. 647–653, 2010.

[71] T. Roddelkopf, "Flexible Automatisierung komplexer Prozesse der analytischen Messtechnik," Ph.D. dissertation, Universität Rostock, 2006.

[72] L. Sarkozi, E. Simson, and L. Ramanathan, "The effects of total laboratory automation on the management of a clinical chemistry laboratory. retrospective analysis of 36 years," *Clinica Chimica Acta*, vol. 329, no. 1-2, pp. 89–94, Mar. 2003.

[73] SBS, "Microplates footprint dimensions," SBS Society for Biomolecular Screening, Tech. Rep. ANSI/SBS 1-2004, 2004.

[74] ——, "Microplates well positions," SBS Society for Biomolecular Screening, Tech. Rep. ANSI/SBS 4-2004, 2004.

[75] R. Schapire, "The boosting approach to machine learning: An overview," *LECTURE NOTES IN STATISTICS*, pp. 149–172, 2003.

[76] J. Shen, M. Yaremko, R. Chachowski, J. Atzler, T. Dupinet, D. Kittrich, H. Kunz, K. Puchegger, and R. Rohlfs, "United states patent: 5768407 - method and system for classifying agglutination reactions," Jun. 1998.

[77] E. P. Spalding, "Computer vision as a tool to study plant development," in *Plant Systems Biology*, D. A. Belostotsky, Ed. Totowa, NJ: Humana Press, 2009, vol. 553, pp. 317–326.

[78] F. Thielecke, "Systemidentifizierung für Ingenieure," Technische Universität Berlin, 2007.

[79] K. Thurow, "High throughput screening and high content screening technologies," Universität Rostock, Apr. 2009.

[80] United States, "21 CFR 11: Electronic records; electronic signature final rule," 1997.

[81] M. Volcker and C. Fattinger, "United states patent: 6473239 - imaging system with a cylindrical lens array," Oct. 2002.

[82] C. Walsh, “Improving efficiency in high throughput screening operations with high-speed identification of problem samples,” in *Proceedings & Abstracts of European Lab Automation Conference*, Hamburg, 2011.

[83] B. Widrow, M. Hoff, and Electronics Labs, Stanford Univ. CA , “Adaptive switching circuits,” 1960.

[84] B. Wire, “ArrayScan system introduces high throughput cell-based screening.” Business Wire, Pittsburgh, May 28 1996.

[85] L. Yeh, W. Wu, R. Tang, Y. Tsai, E. Chiang, P. Chiao, C. Chu, and W. Wang, “High performance 400DPI a4 size contact image sensor (CIS) module for scanner and g4 fax applications,” in *Current Developments in Optical Design and Optical Engineering*, R. E. Fischer and W. J. Smith, Eds., vol. 1527. San Diego, CA, USA: SPIE, Dec. 1991, pp. 361–367.

[86] S. Yi, R. Haralick, and L. Shapiro, “Optimal sensor and light source positioning for machine vision,” *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 122–137, 1995.

[87] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[88] L. Zhu, H. Luo, and X. Zhang, “Uncertainty and sensitivity analysis for camera calibration,” *Industrial Robot: An International Journal*, vol. 36, pp. 238–243, 2009.

[89] A. Zographos, P. Evans, S. Godber, and M. Robinson, “A line-scan system for all-round inspection of objects,” in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3174, 1997, pp. 274–282.

Appendix A

Framework

A.1 Technical Specifications of Imaging Hardware

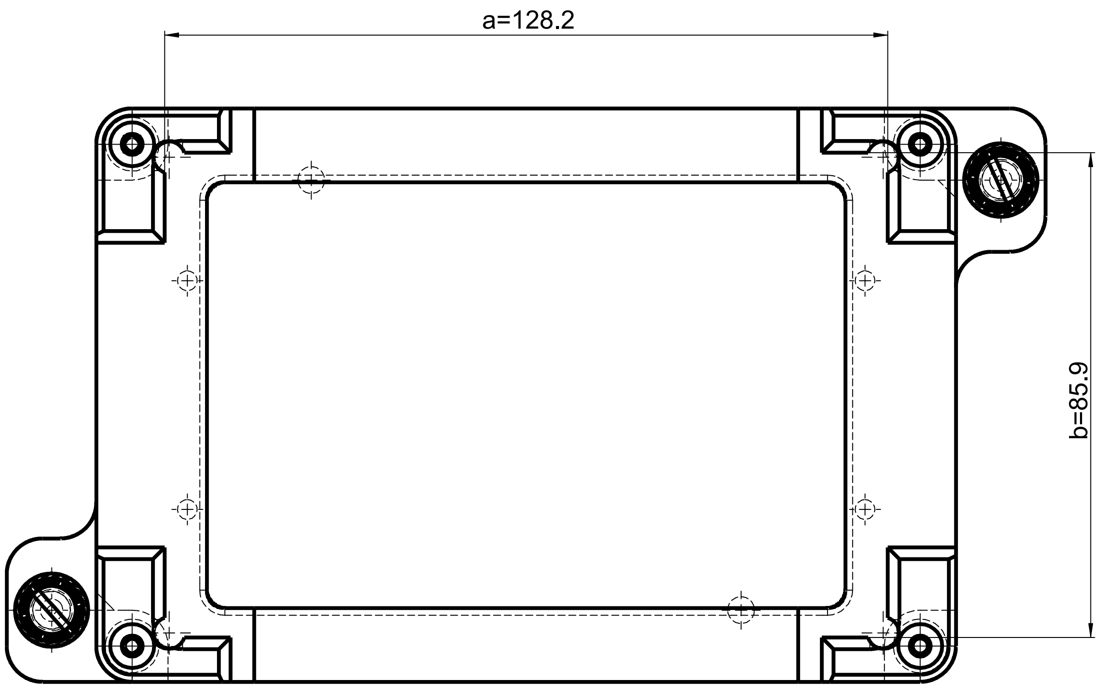


Figure A.1: Technical Drawing of a standard ALP

Table A.1: ALP measurements for figure A.1

	Standard ALP	**ABGene 96**	**FluidX CCD**	**Ziath CIS**
a [mm]	128.2	127.8	128.0	128.5
b [mm]	85.9	85.8	85.8	86.5

ABGene Smartscan 96	
Dimensions (W x D x H)	130 x 180 x 185 mm
Weight	3.7 kg
Power Supply	110-240 V, 50-60 Hz
Speed	< 1 second
Num of Sensors	2
Sensor Type	CCD
Sensor Manufacturer	Lumenera Inc.
Focal length	6.5mm
Aperture size	F 1:1.8

FluidX XTR-96MKII	
Dimensions (W x D x H)	175 x 291 x 71 mm
Power Source	AC 100 to 240 V +/- 10%, less than 8W
Speed	< 10 second @ 600dpi
Num of Sensors	1
Sensor Type	CCD
Light Source	CCFL (cold cathode flourescent light source)
Sensor Manufacturer	Avision
Max. Resolution	600dpi
Max. Image Size	4103 x 2482 px

Ziath ZTS-A6	
Dimensions (W x D x H)	145 x 234 x 41mm
Speed	< 10 second @ 600dpi
Num of Sensors	1
Sensor Type	CIS
Light Source	RGB-LED
Sensor Manufacturer	Toshiba
Max. Resolution	600dpi
Max. Image Size	3496 x 2480 px

A.2 Framework Implementation

Listing A.1: Typical recursive loop

```
recursion(int n_p, int p_act){
        BASE CASE
        if(p_act == n_p)
                return;
        else{
                RECURSION STEP
                command_to_be_executed();
                recursion(n_p, p_akt+1);
        }
}
```

Listing A.2 shows a bad implementation of a simple SIMD problem which is not threadable due to the iterating, in-loop memory allocation (“resize“) of the array that stores the result.

Listing A.2: Access conflict

```
do i = 1, 1000
  resize(a, size(a)+1)
  a(i) = b(i) + c(i)
end
```

Table A.2: SAMI Data Message

Required	**Key**	**Data**
X	Transport	The name of the transport that the data is about.
	Command	The Command message that produced this data (if any)
X	Data Type	The type of data included; one of [Numeric, Text]
X	Time Stamp	The time/date that the data was taken.
X	Count	The number of samples that data was taken on
X	Data	The data - may be a string or sub-message depending on Data Format.
X	Data Format	One of [Single, By Identifier Index] If Single, one piece of data as a string in Data; Data is a key. If By Identifier Index, each sample has a key under Data; Data is a sub-message.
	Format Text	A text description of how the data is listed. Some examples: Bitmap of the plate Wells 1-96 in row major order
	Description	A sub-message that describes the conditions under which the data were taken. For example, the temperature or wavelength of light used.

A.3 Parallel Programming

Listing A.3: Access conflict

```
do i = 1, 1000
  resize(a, size(a)+1)
  a(i) = b(i) + c(i)
end
```

Listing A.4: Badly separated loop

```
do i = 1, 1000
  c(i) = readFile(filename)  /* short timeframe, e.g.  < 1s */
  a(i) = complex_computation(b(i),c(i),d(i))
/* long timeframe, e.g. > 3s */
end
```

Listing A.5: Well separated loop

```
do i = 1, 1000
/* sequential part */
  c(i) = readFile(filename)  /* short timeframe < 1s */
end
do i = 1, 1000
/* parallel part*/
  a(i) = complex_computation(b(i),c(i),d(i))
/* long timeframe > 3s */
end
```

A.4 Parallel Performance Evaluation

Table A.3: System information

System information	
Manufacturer	Dell
	OptiPlex 990 MT
CPU	Intel Core i7-2600 (8M Cache, 3.40 GHz)
n_{cores}	4
$n_{threads}$	8
RAM	3GB (1X1GB/1X2GB) 1333 MHz
Operating System	Windows XP

A.4.1 Stack

Table A.4: 384 Wells

	time		**speedup**		**efficiency**
n_cores	**t**	σ	s	σ	
0	61.264	0.886	0.997	0.014	0.997
1	61.078	1.034	1.000	0.017	1.000
2	35.019	0.625	1.745	0.031	0.872
4	19.286	0.352	3.168	0.057	0.792
8	14.242	0.166	4.289	0.050	0.536

Table A.5: 96 Wells

	time		**speedup**		**efficiency**
n_cores	**t**	σ	s	σ	
0	78.991	0.742	0.992	0.009	0.992
1	78.374	0.718	1.000	0.009	1.000
2	47.602	0.875	1.647	0.030	0.823
4	27.067	0.604	2.897	0.064	0.724
8	19.383	0.120	4.044	0.025	0.505

A.4.2 Plate

Table A.6: 384 Wells

	time		**speedup**		**efficiency**
n_cores	**t**	σ	**s**	σ	
0	0.492	0.016	0.990	0.031	0.990
1	0.487	0.026	1.003	0.054	1.003
2	0.344	0.013	1.415	0.052	0.708
4	0.279	0.032	1.762	0.151	0.440
8	0.279	0.115	1.859	0.294	0.232

Table A.7: 96 Wells

	time		**speedup**		**efficiency**
n_cores	**t**	σ	**s**	σ	
0	0.485	0.011	0.989	0.021	0.989
1	0.480	0.018	1.001	0.038	1.001
2	0.342	0.013	1.403	0.054	0.702
4	0.269	0.013	1.788	0.081	0.447
8	0.253	0.009	1.897	0.068	0.237

A.5 CCD Distortion Measurement

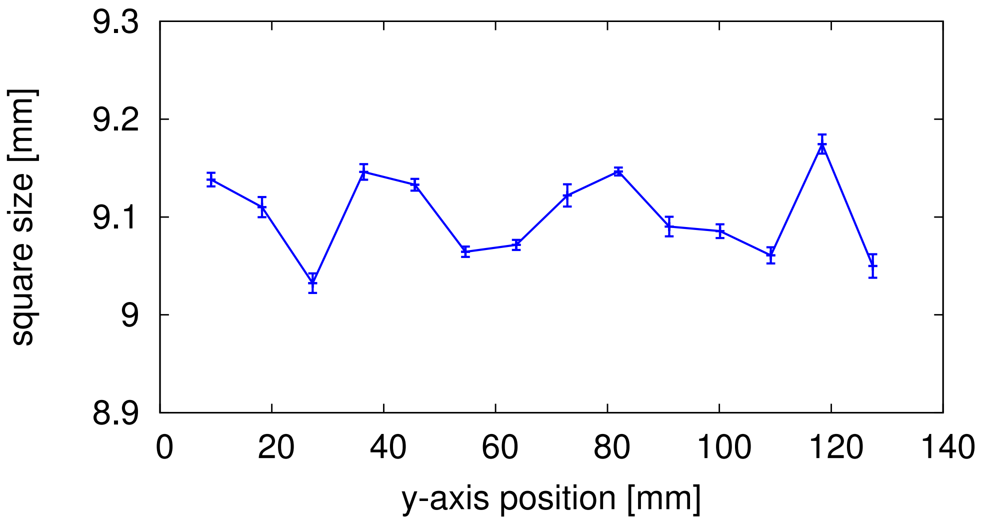


Figure A.2: CCD: square size in y-axis vs. y-axis position

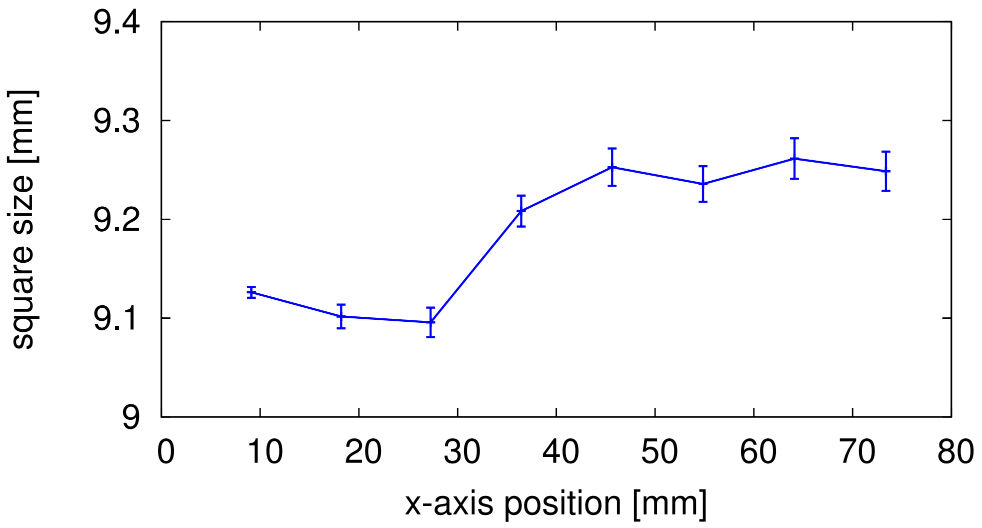
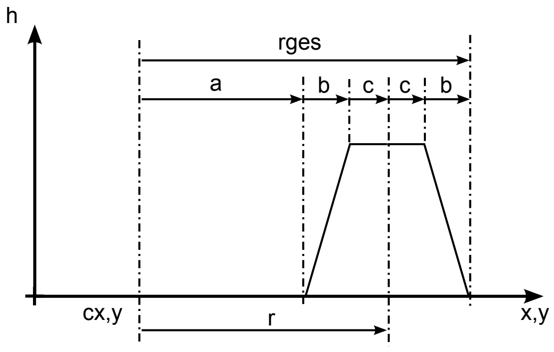
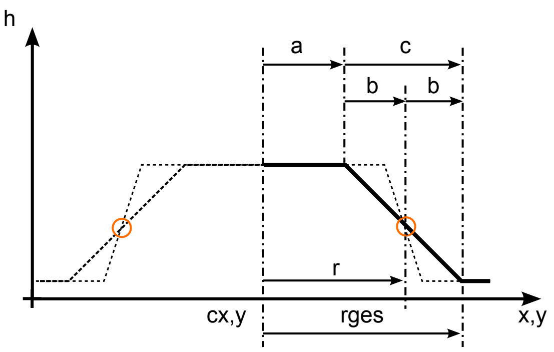


Figure A.3: CCD: square size in x-axis vs. x-axis position

A.6 Well Area Detection using 2D Cross-Correlation



(a) circle-shaped



(b) filled circle-shaped

Figure A.4: Parameterized kernel shapes

Table A.8: Overall results for plate VPP96

template	**r**	**b**	**c**	**mean error [px]**	**std [px]**
circle	100	1	10	10.8995	5.5319
circle	100	8	10	12.1864	4.3955
filled circle	100		1	11.5322	10.4568
filled circle	100		10	11.1251	10.1313

Table A.9: Outlier from image 1 (Fig. A.9) influences the result

Img	c=1 px	c=10 px
1	26.1630	25.0599
2	7.7782	7.8102
3	10.6066	10.6301
4	1.5811	1.0000

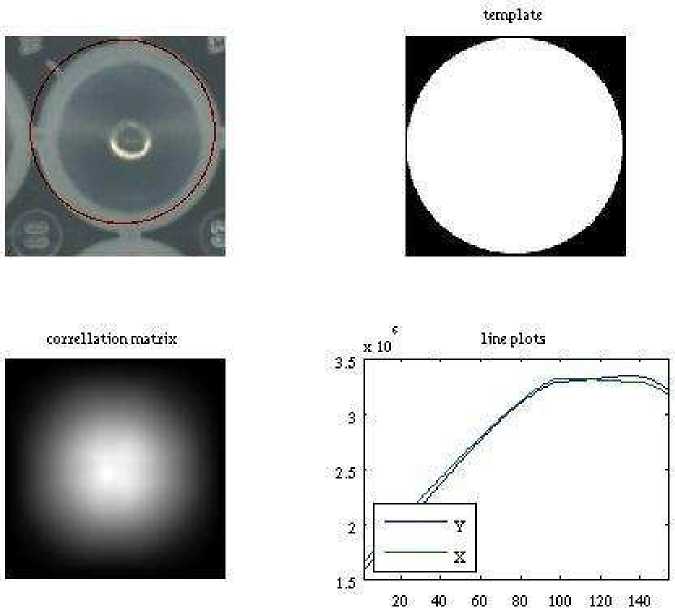


Figure A.5: Good result despite filled well, $c = 1$

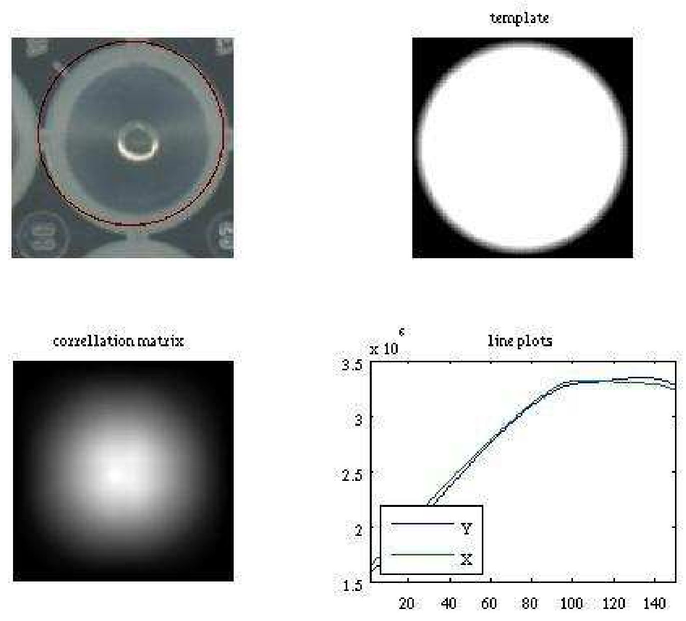


Figure A.6: Result with $c = 10$

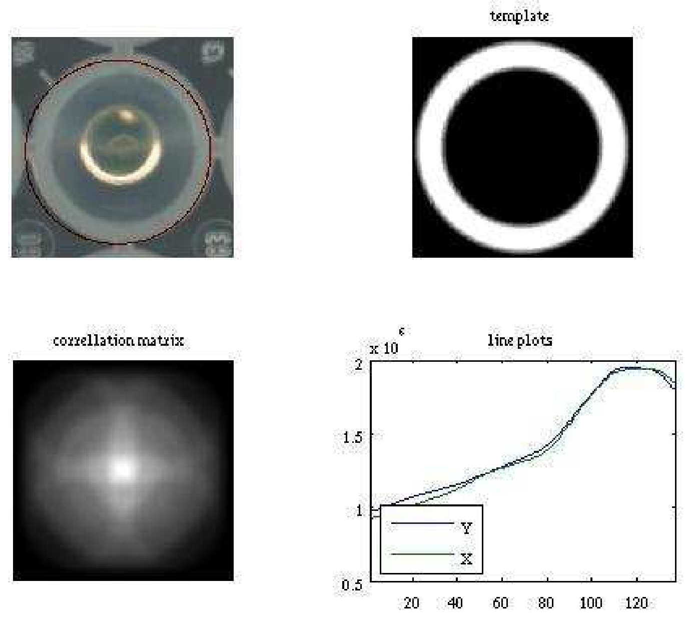


Figure A.7: Result with empty circle, $b = 8$

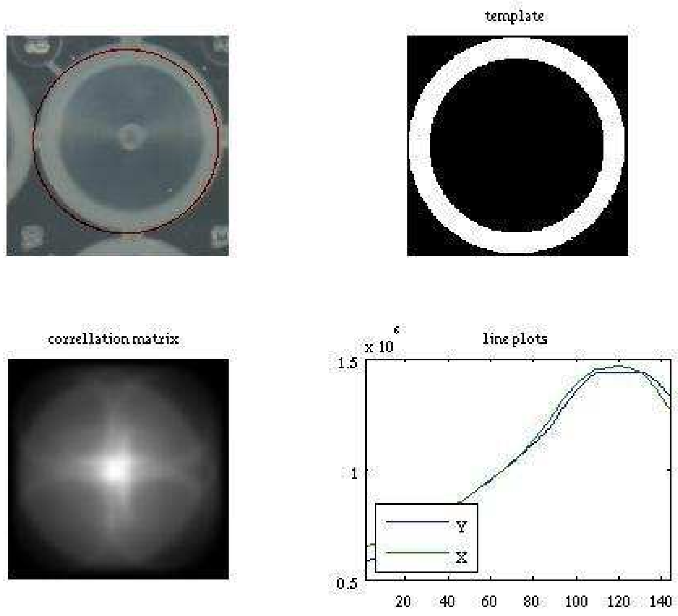


Figure A.8: Result with empty circle, $b = 1$

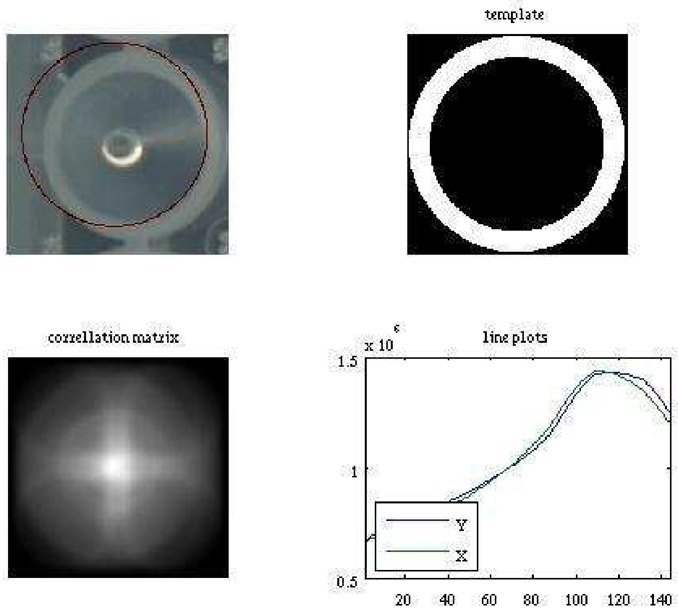


Figure A.9: Problematic behaviour at the edge of the plate

Appendix B

Pipettor Monitoring

B.1 Test Results

B.1.1 Performance versus pipetted volume

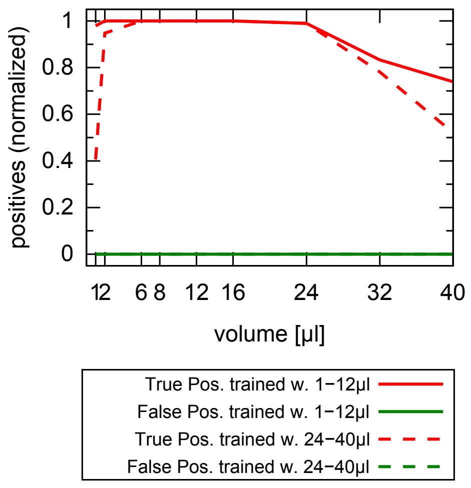


Figure B.1: VPP plate, CCD scanner

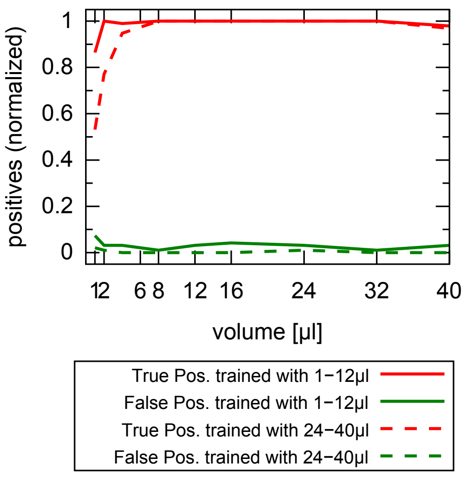


Figure B.2: VPS plate, CCD scanner

B.1.2 Influence of environmental lighting

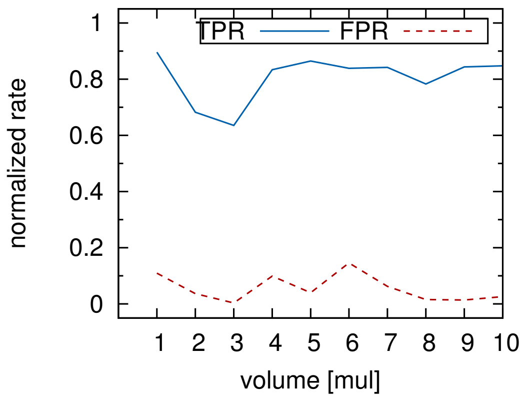


Figure B.3: VPP plate, CCD, DMSO, normal lighting

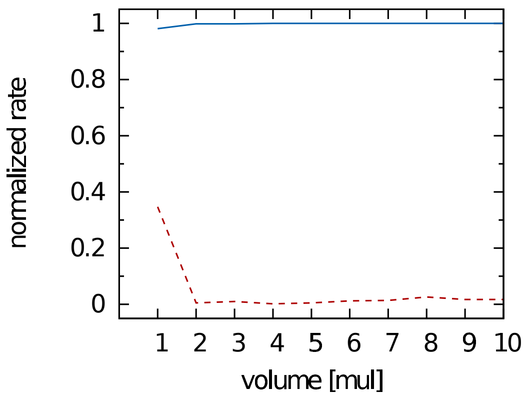


Figure B.4: VPP plate, CCD, Water, normal lab lighting

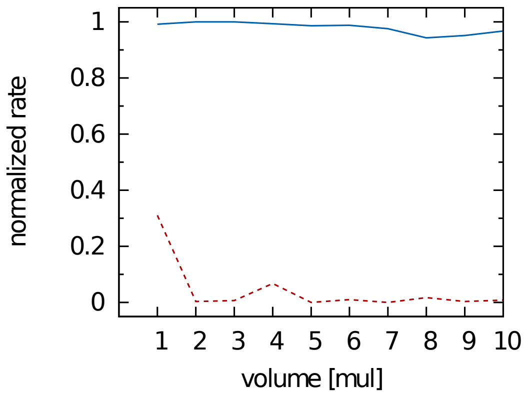


Figure B.5: VPP plate, CCD, weak lighting

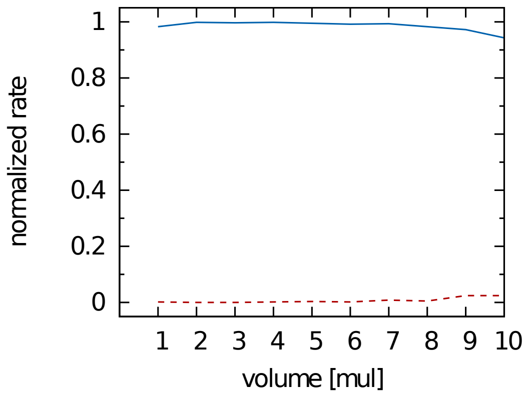


Figure B.6: VPP plate, CCD, Water, no ambient light

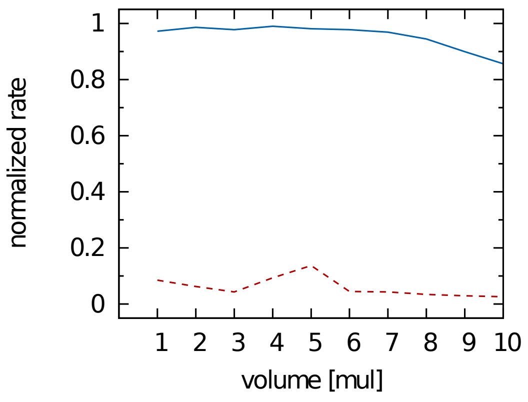


Figure B.7: VPP plate, CIS (lid), Water, normal lighting

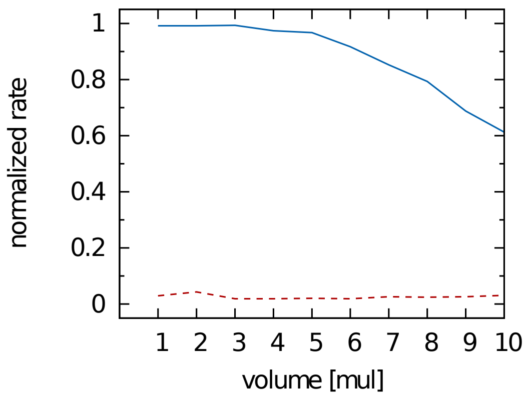


Figure B.8: VPP plate, CIS (lid), Water, weak lighting

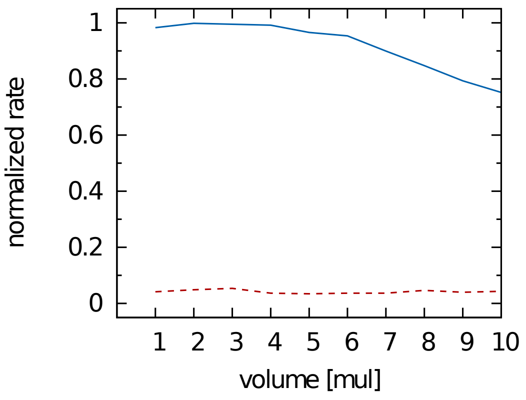


Figure B.9: VPP plate, CCD (lid), no ambient light

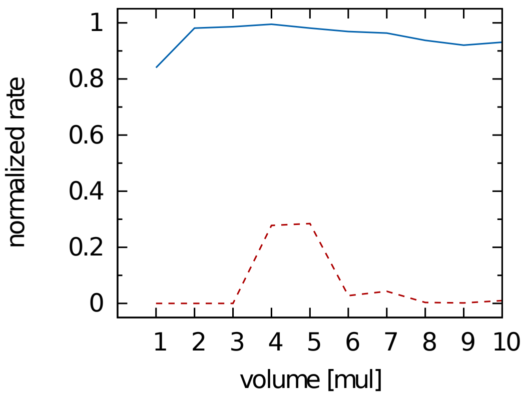


Figure B.10: VPP plate, CIS (open), Water, normal lighting

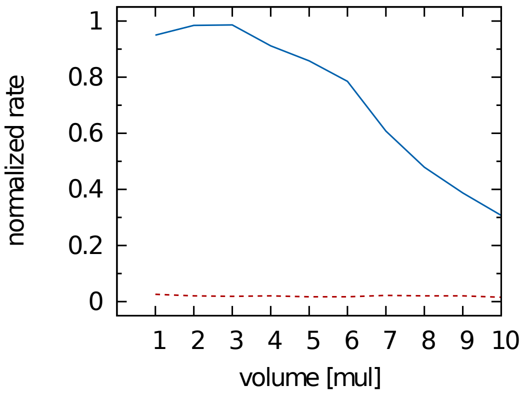


Figure B.11: VPP plate, CIS (open), Water, weak lighting

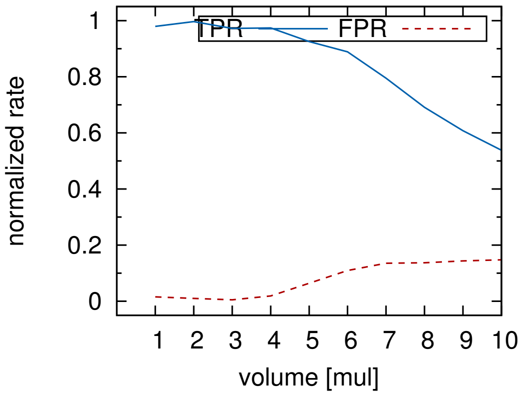


Figure B.12: VPP plate, CIS (open), Water, no ambient light

Theses

1. By integrating laboratory devices as robotic processes, the human ability to supervise processes and sense errors early in the process is lost.

2. This loss can be addressed by optical sensors that are able to detect states of the samples during the process and provide feedback to the process control software, which runs the method. Such quality checks are most effective prior to a part of the process that represents a bottleneck.

3. Optical sensors can furthermore be used to control the process, by feeding the measurements back and controlling the process based on them.

4. In order to increase efficiency and throughput, parallel sample carriers are used in laboratories. Microtiterplates are standardized labware formats that are popular and are available in multiple densities, geometries and materials.

5. To capture all samples of a microtiterplate in a one-step measurement, the plate has to be imaged from above or below the plate, using transmitted or reflective lighting setups. The relevant setup is application specific, and different devices are advantageous for different applications.

6. Systems based on flatbed scanners are suitable to image a plate with reflective lighting from below. They offer a uniform perspective of the different wells, which would require a telecentric lens when using a camera system.

7. By using multiple cameras, it is possible to achieve advantageous uniform perspectives with smaller installation height of the imaging device. Camera systems are faster compared to flatbed scanner systems.

8. To reduce development effort of optical inspection applications, a software framework can provide generic functionality that is required, independent of the actual application. Such generic functionality includes hardware communication and optical models together with their calibration.

9. Calibration includes the determination of all transformation parameters required to map real-world coordinates to the image plane. This includes intrinsic parameters

of the sensors and extrinsic parameters such as the positions of the sensors relative to each other and relative to the labware.

10. Established methods exist for camera calibration, while a suitable model is required for flatbed-scanners. A registration approach to determine the extrinsic parameters can be developed to suit the laboratory setup. It includes the placement of a reference object onto the labware position during the setup of the machine, and replaces any geometric measurements hereby.
11. A registration approach for multi camera setups is furthermore required. All hardware calibration and registration procedures are required to follow the same basic steps in order to be integrated independently.
12. Other requirements of the framework include labware compatibility, load distribution and user interfaces for system setup and operation.
13. The performance of pipettors in terms of accuracy and precision is worst on the lower volume bound of its operating range. A good example application to be implemented with the framework is a optical drop detection algorithm to monitor low-volume liquid handling processes.
14. By weighing three developed algorithms specific to a plate type, it is possible to robustly detect drops in microtiterplates. However it was required to implement a grid search to also configure each algorithm for a specific plate-device combination.
15. The first algorithm uses the image of an empty plate in order to subtract the background. Morphological operations are used to reduce noise prior to thresholding the image. The number of true pixels for every color channel is the output.
16. The second also uses background subtraction and subsequently configurable filters. Then the first three statistical moments are calculated as result.
17. The third algorithm uses a hough transformation to detect circular objects in the image. It dismisses objects with a diameter similar to the well diameter to reduce the number of false positives.
18. Subsequently, the results can be relayed to the process control system and can be used for documentation, but also for process control, i. e. by automatic repipetting or notification alerts.

Zusammenfassung

Bildverarbeitung im Sinne dieser Arbeit beschreibt die maschinelle Auswertung von Bildern zur Merkmalsextraktion und die nachfolgende Bereitstellung der gewonnen Zustandsinformation an ein übergeordnetes Prozessleitsystem zur automatischen Umsetzung von Reaktionen. Im Anwendungsbereich der Laborautomatisierung ist diese Nutzung von Bildverarbeitung (BV) für Qualitätskontrolle und Prozesskontrolle, verglichen mit anderen Bereichen wie der Fabrikautomatisierung und der Nahrungsmittelverarbeitung, wenig ausgeprägt.

Als mögliche Anwendungen für ein Bildverarbeitungssystem werden Beispiele für qualitative Prüfung, aber auch quantitative Messung und komplexe Merkmalsextraktion erarbeitet. Dabei wird auf die parallele Natur von Laborprozessen eingegangen. Die parallele Aufnahme einer großen Anzahl von Proben stellt Anforderungen an das Aufnahmesystem in Bezug auf die Perspektive und die Auflösung. Es werden kamerabasierte Systeme mit einer oder mehreren Kameras und Flachbettscanner vorgestellt.

Weitere allgemeingültige Anforderungen an BV-Systeme werden definiert. Zu den Anforderungen gehört die Kompatibilität zu anwendungsspezifischer Labware und Hardware des Aufnahmesystems sowie eine einfache Konfiguration des Systems. Weiterhin ist aufgrund des parallelen Charakters der Experimente die Menge von Eingangsdaten groß verglichen zu anderen BV-Systemen, so dass optimale Ressourcennutzung erforderlich ist. Weitere Anforderungen sind Dokumentierbarkeit der Systemkonfiguration, grafische Nutzerschnittstellen und Administrationstools sowie die Integration in Prozessleitsysteme.

Das entwickelte Integrationskonzept beschreibt eine Softwarelösung im Hinblick auf die oben genannten Anforderungen in Form eines Softwarerahmenwerks. Dabei wird vor allem auf die Hardwarekompatibilität eingegangen. Die Hardwareabstraktionsschicht enthält zwei Ebenen, eine triviale Übersetzungsschicht zur implementationsspezifischen Schnittstelle der Gerätetreiber und eine Ebene zur perspektivischen Transformation der globalen Koordinaten auf die Bildebene. Hier werden bekannte Modelle für Kamerasysteme integriert. Zusätzlich wird erstmals die systematische Integration von Multi-Kamerasystemen und ein einfaches optisches Modell für Flachbettscanner entwickelt und beschrieben. Weiterhin werden Konzepte zur optimalen, anwendungsunabhängigen Lastverteilung auf SMP-Mehrkernprozessoren und für adaptierbare Nutzerschnittstellen vorgestellt.

Im Rahmen dieser Arbeit ist eine Referenzimplementierung entstanden, die die en-

twickelten Konzepte implementiert und validiert. Sie wurde weiterhin dazu genutzt, eine Beispielapplikation für die Nutzung zur Qualitäts- und Prozesskontrolle umzusetzen. Die implementierte Anwendung ist ein Konzept für die Detektion kleinster Volumina in Mikrotiterplatten (ein standardisiertes Labwareformat). Der Algorithmus nutzt drei Detektoren und wichtet anschließend deren Ergebnisse. Durch optimale Einstellung der einzelnen Detektoren und der Wichtung ihrer Ergebnisse kann Kompatibilität zu verschiedener Hardware und Labware gewährleistet werden. Die Anwendung wurde in einer umfassenden Validierung auf ihre Robustheit gegenüber den Einflussfaktoren Umgebungslicht, Fluideigenschaften und Pipettiertechnik untersucht. In einem abschließenden Test wurde das Ergebnis bestätigt und eine erste qualitative Auswertung der Messung angestellt.

Die Verbesserung der quantitativen Ergebnisse und die Verbesserung der Robustheit gegenüber sich veränderndem Umgebungslicht, insbesondere für Kamerasysteme, sind Themen für die zukünftige Weiterentwicklung. Außerdem können andere Konfigurationen von Multi-Kamerasystemen und die Nutzung von mehreren Abbildungen einer einzigen Probe untersucht werden.

Abstract

The term computer vision (CV), as it is used in this thesis, means the automated processing of images to generate abstract knowledge about the imaged objects, to subsequently use this information by higher level systems to define appropriate reactions to a situation. Compared to quality and process control in industrial production lines, or in the food processing industry, such an application is underdeveloped in life science automation today.

This thesis presents possible applications for CV-based systems. The parallel nature of experiments conducted in high-throughput laboratories is considered, as it poses special requirements on an imaging device in terms of perspective and resolution. Applicable imaging devices will be presented that are camera-based, multi-camera-based or flatbed scanner-based.

Generic requirements of CV-based inspection (automatic optical inspection, AOI) systems are elaborated and defined. The requirements include compatibility concerning application-specific labware and hardware and straightforward methods to set up and reconfigure the system. The parallel nature furthermore leads to a large amount of input data that should be processed such that the system resources are used in an optimal manner. Further requirements are: Reproducible system settings and documentation of settings, graphical user interface and a flexible manner for the integration into process control systems.

Concepts of a software framework with respect to the above-mentioned requirements are elaborated, with special emphasis on the hardware compatibility. The introduced hardware abstraction layer consists of two sub-layers, of which the lower one acts as a trivial translator to the driver-specific hardware interface, while the upper layer is responsible for the optical modeling of the system and the mapping between real world coordinates and the image plane. Here, an available camera model is integrated, together with a simple flatbed-scanner model and a multi-camera registration method that was purposely developed for this case. Furthermore, a concept for application independent load distribution and an adaptable graphical user interface is presented.

A reference implementation has been developed that implements and validates the presented concepts. It was further used to implement a quality and process control application developed in parallel to the framework.

The implemented application aims at the detection of low-volume liquids in microtiter

plates, a labware standard. The algorithm uses three detectors and weighs their output. By optimizing each detector and the weights for a labware and an imaging device, it is possible to cover a wide range of labware on different flatbed scanner hardware. Robustness tests considering ambient light, fluid properties and pipetting techniques where conducted. In a final test, the application was validated and a first quantitative evaluation of the measurement results was considered.

Improvements of the quantitative measurements and of the robustness regarding ambient light for camera-based systems can be subject to future research. Furthermore, the use of multiple camera systems in a way that provides multiple views of a single sample can be investigated.