Preface

Five years at NTNU have been a rollercoaster ride. Uphills at times, but also a great deal of fun. I'm grateful for the number of marvellous people I have met, the flexibility the student life brings, all the fun with the student society Spanskrøret and not to forget all the things I've learned.

A special thank you go to all the professors who withstand sharing their knownledge every day, even at time when their students seems unmotivated.

The last year I have been warmly included in Magnus Borstad Lilledahl's research group, with Andreas Finnøy, Elisabeth Inge Romijn and Rajesh Kumar. It's been educational to work with them and exciting to get an insight in how they perform their work. Anna Bofin and Monica J. Engstrøm at St. Olavs has also been very welcoming, showing me histological patterns in tissues and provided the data set to my gratefulness. Thank you all!

I would also like to thank my family, who always have been supportive for the choices I've made.

Lastly, the greatest thanks go to my life companion Yngvild, it wouldn't have been the same without you.

The cheesy quote is..

> *The future is already here, it's just not very evenly distributed.* - William Gibson

Contents

Abstract

St. Olavs hospital has supplied a dataset of 2703 tissue samples at the tumor peripheral from $\approx$ 900 patients. NTNU want to examine all tissue samples with image processing to see if second harmonic generation microscope images of tissue can help classify cancer type (I, II, III) or in other words, cancer aggresiveness. This thesis documents a method which automates the microscope imaging of these tissue microarrays (TMA) and show how images can be structured and correlated to clinical data.

Automated microscope scanning is in principle straight forward, but the implementation is dependent on many aspects of the experimental setup. In general, some of the aspects discussed in this thesis are:

- Create image analysis algorithms that are robust to experimental variations.
- Correction of systematic errors like
  - intensity variations and
  - difference in coordinate systems of scanning raster patterns and stage movement.
- Automatic stitching of regular spaced images with variable degree of signal entropy in seams.
- Adjusting z-plane for large area samples with micrometer precision.

The aspects listed above are not unique to TMA-specimens and the experimental setup, and could be useful for similar projects. But the focus of the thesis will be on TMA and the experimental setup with a Leica SP8 microscope.

The conclusions are:

- Large area scans should adjust specimen plane to be at even distance to the objective to be time effective and avoid out of focus images.
- Using heuristics/constraints improves the reliability to automatic stitching algorithms, failing gracefully on images with little entropy in overlap.
- Leica LAS X version 1.1.0.12420 have limited support for automatic microscopy, but it's possible to work around limitations to leverage fully automatic TMA-scanning.

Chapter 1

Introduction

With a population just above 5 million[1], three thousand women are diagnosed with breast cancer each year[2] in Norway. This makes breast cancer the most common kind of cancer, affecting one of every eleventh woman. Luckily breast cancer is often treatable, shown by the fatalities which was 649 in 2012[2]. NTNU and St. Olavs hospital have been cooperating on reasearch to find new ways to diagnose patients. The cooperation yielded a study[3] on 37 subjects which showed positive results on difference of collagen structure from different parts of tumor tissue. This thesis seeks to make it possible to expand the study from 37 subjects to the whole dataset available of $\approx$ 900 subjects.

The means to achieve the expanded dataset is automating the microscope imaging, with main focus on automated scanning of tissue microarrays. Tissue micro arrays are glass slides with samples arranged in a matrix pattern seen in figure 1.1. As tissue microarrays is standard procedure, not unique to breast cancer tissue, the work of this master is relevant for other studies too.

The tissue micro array shown in figure 1.1 is $\approx$ 24x15 mm in size. Using a

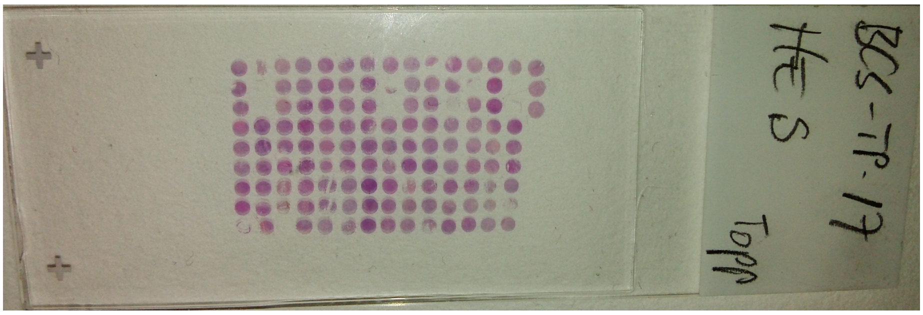


Figure 1.1: Tissue micro array of breast tissue at perifer of tumor. Three test samples are beside the array of 14x9 samples to avoid mix up of patients when rotating the slide.

moderate objective of 25x with 400 µm field of view, a single scan of the total dataset will be

$$\frac{24\text{mm}}{400\mu\text{m}} \cdot \frac{15\text{mm}}{400\mu\text{m}} = 2250 \text{ images.}$$

Depending on the precission of the microscope stage, images are not necessarry easily put together. Also, keeping microscope in focus for the whole surface can become challenging. Another approach would be not to scan the whole area in one scan, but to scan each of the $14 \cdot 9 = 126$ tissue specimens one by one. The challenge with scanning each region one by one is that the samples are often not equally spaced, and a lot of manual error prone labor is required to define the areas to scan. The method in this thesis tries to simplify the scanning process and prepare the images for further analysis.

The thesis are written with focus on two parts, namely automating the collection of images and correlating samples to clinical data. How this can be used in supervised machine learning will be briefly mentioned in the end. In total the method described should enable researchers to run experiments on large datasets of tissue microarrays in a structured and determined manner.

A reader of this text should be familiar with general physics. Matters that are specific to scanning microscopy and image processing will be described in the theory section, along with software concepts in use. The method section seeks to make the reader able to replicate the experiment on any kind of microscope, but some software and solutions will be specific to the Leica SP8 microscope. The result section will mark out leverages gained with automated scanning, and the discussion holds details on choices made when developing the method and limitations stumbled upon.

All source code in the thesis will be in the programming language Python[4]. The reader does not need to be proficient in Python programming, but acquaintance with the syntax is assumed. Code blocks will be used to clarify how problems have been solved or algorithms have been implemented. Details not essential to the problem at hand have been omitted to keep focus on the essential parts. As total amount of source code are above thousand lines it's not included in the appendix but rather available at github[???] with full history. A brief description on installation of the software is included in the appendix.

As this thesis mainly consists of work on creating automated microscope scanning, the method is also the result of the thesis. Therefore a result chapter is not included, but a brief description of the result is in the beginning of the method chapter.

Chapter 2

Theory

Tissue microarrays

A tissue microarray is a collection of specimens aranged in a matrix pattern. The specimens are typically sliced with microtome from a paraffin block containing cylinders of tissue in rows and columns. Cylinders for the paraffin block are often picked out by a pathologist who evaluate the histology of a larger tissue sample and choose appropriate locations.

The thickness of slices are in the magnitude of 1 µm, which gives efficient use of tissue samples in the sense that several hundreds of TMAs can be made from a block containing cylinders of height 1 mm[5]. In this text specimen spot will refer to a single sample in the array.

Scanning microscope

Figure 2.1 illustrate the internal workings of a Leica SP8 scanning microscope which have an epi-illumination setup. Epi-illumination is when the detectors (26) and light source (1, 3, 5, 7) are on the same side of the objective (18). But as seen, the epi-setup also allows for external detectors (19), which were the ones in use. By scanning one means that the light source is focused to a specific part of the specimen and scanned line by line in a raster pattern. While the laser are scanned over the surface, a detector measure light in samples and each measured sample will be saved to an image pixel. The scanning is done by a oscillation mirror (14). The term non descanned detector indicate that the light does not travel by the scanning mirror before reaching the detector. In SP8 (17) and (19) are non descanned detectors, where (17) measure reflected light and (19) measure transmitted light. In figure 2.1 the condensor, which gathers light for the non-descanned detector, is not illustrated, but it should be between the glass slide and the non-descanned detector (19).

The view field of a microscope is the physical area which fits inside one image. The view field depends on the magnification of the objective and the scanner zoom. Scanner zoom is when the scanner is set to oscillate with less amplitude while still sampling at the same rate. As field of view is at the magnitude of 1×10^{-4} m, specimen must be moved around to image a larger area. The device that moves the specimen is called a stage. Here stage position, or specimen position if you like, is denoted with a upper case X to distinguish it from lower case x which denote image pixel position.

The resolution of a conventional light microscope is given by the objective and/or condenser numerical apterture (NA)[7]:

$$d = \frac{1.22\lambda}{NA_{condenser} + NA_{objective}}. \tag{2.1}$$

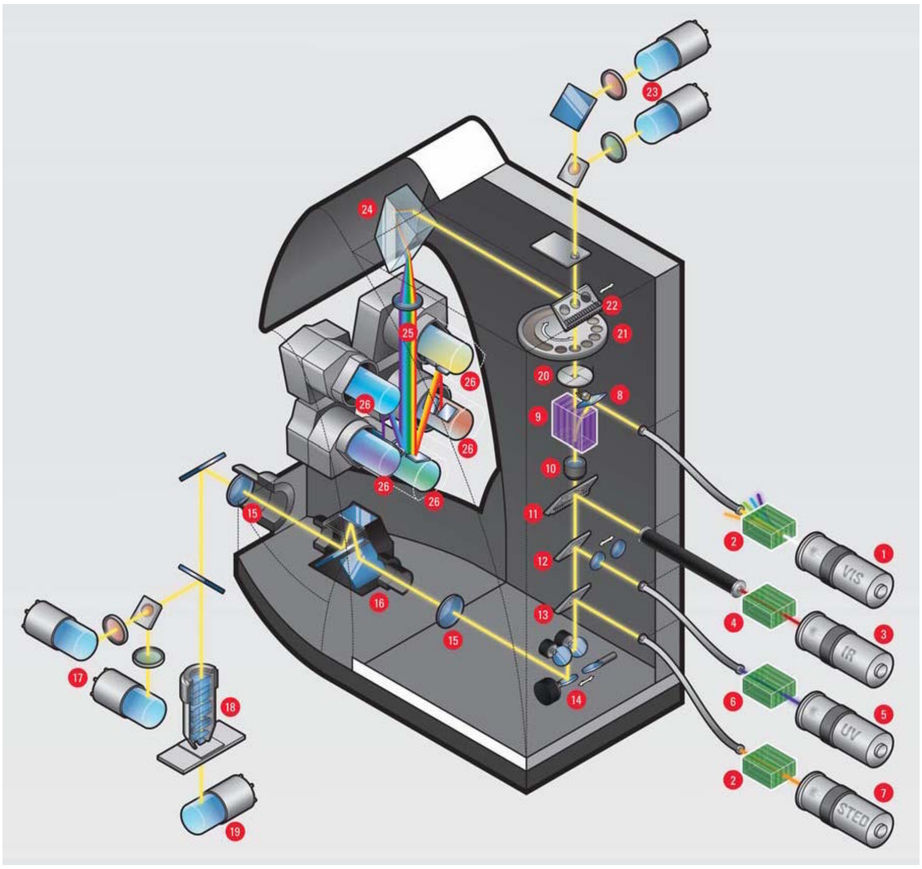


Figure 2.1: Internals of a Leica SP8 microscope. Picture from Leica SP8 brochure[6].

Here d is the minimum separable spatial distance defined by the Rayleigh criterion, λ is the wavelength of the light and NA is the numerical apterture.

A dichroic mirror, or also called a dichromatic beamsplitter, is a filter which is putted into the laser beam at 45° angle to split light of different wavelengths. The filter has a sharp transition between reflecting and transmitting light for a given wavelength, resulting in short wavelengths being mirrored 90° and high wavelengths pass through[7]. This is useful when having several detectors which should detect different wavelengths.

Second harmonic generation (SHG) is a nonlinear scattering process of two photons with the same wavelengths. The process is an interaction where the photons is transformed to a single emitted photon of half the wavelength. The process is dependent on orientation of electric dipoles in the specimen and aligned assemblies of asymetric molecules usually provides the proper conditions. Collagen tissue does hold the proper conditions for SHG-imaging[7].

As the probability for SHG is extremely low, enormouse amount if light is necessary to generate it. This fact is a benefit for scanning microscope that only the focal point is able to produce SHG, with the consequences that a sensors can be simpler, e.g. non-descanned, as light will always originate from where the laser is pointed to.

SHG necessary?

Image processing

The term image in this contex a two dimentional array of values, where each position in the array is called a pixel. Resolution is the number of pixels an image holds. E.g. a resolution of 1024x1024 is an image with 1024 pixels in both x- and y-direction, totalling 1×10^6 pixels. Each pixel represent a physical position of the specimen, where the value is the amount of light measured from the detector when scanning the specimen surface with a light source. The physical size of the pixel will depend on sampling rate. All images in this thesis are 8 bit grayscale images, meaning that each pixel can hold $2^8 = 256$ values. In an ideal experiment a pixel value of zero denote zero detected light and 255 is the maximum, but this is an simplification as noise will be measured too.

$f(x, y)$ denotes the intesity of pixel at position (x, y), where $(0, 0)$ is the top left of the image, positive x-direction going left and positive y-direction going down. $m \times n$ will denote the number of pixels in in respectively x- and y-direction. A subscript of the image name is used if several images are discussed, e.g. m_f is the number of x-pixels of image f.

Image registration

Image registration is the process of putting images into the same coordinate system. In this context the sources are images from different microscope stage coordinates. One way of finding how images are displaces is by using correlation. Correlation is defined as the following: If we have an window image w of size $m_w \times n_w$, the correlation of w and f is

$$w(x, y) \star f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t). \tag{2.2}$$

Assuming the window size is odd, $a = (m_w - 1)/2$ and $b = (n_w - 1)/2^8$. As equation 2.2 is sensitive to intensity in w and f, one usually use the normalized correlation

$$\gamma(x,y) = \frac{\sum_s \sum_t [w(s,t) - \bar{w}] \sum_s \sum_t [f(x+s,y+t) - \bar{f}(x+s,y+t)]}{\left\{ \sum_s \sum_t [w(s,t) - \bar{w}]^2 \sum_s \sum_t \left[f(x+s,y+t) - \bar{f}(x+s,y+t)\right]^2 \right\}^{\frac{1}{2}}}. \tag{2.3}$$

Kanskje ta bort dette, jeg bruker stort sett bare phase correlation, likevel om jeg har prøvd template matching.

In the normalized cross correlation, the window is often called a *template* and the process of correlation is called *template matching.* The maximum peak(s) in $\gamma(x,y)$ will be where the template has the best match, which may be in several positions if several matches are made. By padding $f(x,y)$ before correlating one can also match at the border of $f(x,y)^8$.

If f and w are large images, calculation of equation 2.3 is quite computational costly. To reduce the calculation one might use the 2-D discrete Fourier transform (DFT). The DFT $F(u,v)$ of an image $f(x,y)$ is computed by

$$F(u,v) = \mathfrak{F}\left\{f(x,y)\right\} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} f(x,y) e^{-i2\pi(ux/m + vy/n)}. \tag{2.4}$$

Here $F(u,v)$ is the frequency domain image and $\mathfrak{F}\left\{f(x,y)\right\}$ is the notation for a Fourier transform of $f(x,y)$.

Similar the inverse Fourier is defined as

$$f(x,y) = \mathfrak{F}^{-1}\left\{f(x,y)\right\} = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} F(u,v) e^{i2\pi(ux/m + vy/n)}. \tag{2.5}$$

Luckily the sums of equation 2.4 are seperatable and can be done separately in rows and columns, yielding the fast Fourier transform which reduces the calculation complexity from $O(mn)$ to $O(m \log m + n \log n)$[8].

DFT has the intresting property that a element wise multiplication in the frequency domain with one of the images complex conjugated is equivalent as a correlation in the real domain. The correlation theorem states

$$f(x, y) \star g(x, y) = \mathfrak{F}^{-1} \{F^*(u, v)G(u, v)\} . \tag{2.6}$$

Here it's assumed that images are zero padded and $F^*(u, v)$ denotes the complex conjugate of $F(u, v)$.

Sliding window filters

Sliding window filters are similar to correlation filters in the sense that they look at neighbouring pixels of a center pixel. The window will be all pixels sourrounding the center pixel defined by the size of the window. A histogram of the values in the window are updated instead of doing computation directly with the values. The histogram is efficient updated by removing the values going out of the window and adding the values comming into the window when moving to the next pixel. Sliding window filters are also called rank filters.

Might remove if it's not naturally to mention that the population bilateral filter is implemented in `leicaautomator` as compiled python (faster than existing filters in scientific packages).

Software

Leica LAS X

LAS X is the software that contols the Leica SP8 microscope. LAS X comes with an function called *Matrix Screener*, which allows the user to define structured areas to scan. The software uses the concepts fields and wells. A field is essentially an image, and a well is a collection of regular spaced images. The wells may be regular spaced, or an offset between wells can be defined in the graphical user interface. When the scan job is started Leica LAS will store images in a tree of folders in TIFF format.

CAM

In addition to controlling the microscope with the graphical user interface, a function called *Computer Assisted Microscopy* (CAM) can be turned on. CAM is a socket interface, meaning one send bytes over a network interface. This is very similar to how one can write bytes to a file, but in addition the socket interface can respond and send bytes back. The network interface runs on TCP port 8895 and one may communicate locally or over TCP/IP network. A set of 44 commands are available, but only three of them are intresting for the purpose of controlling scans; `load`, `autofocusscan` and `startscan`. More details on the interface can be read in the manual[9] or by studying the source code of the Python package 'leicacam“[10]. Code Block 1 show how one can communicate with the microscope in Python.

XML

Extensible Markup Language is a declarative language which most high level programming languages speak, which makes it suitable for computer program

Code block 1 Communicating with the Leica SP8 microscope using the Python package leicacam.

```python
from leicacam import CAM

cam = CAM()                          # connect to localhost:8895
cam.load_template('leicaautomator') # load a template named leicaautomator
cam.autofocus_scan()                 # start autofocusing
cam.start_scan()                     # start scan job
relpath = cam.wait_for('relpath') # response from microscope with filename
cam.wait_for('inf', 'scanfinished') # wait until scan is done
```

communication. A XML-file contain a single root and tree structure with parent and children nodes. Any position in the tree can be specified with an *XPath*. Code Block 2 show a typical structure of a XML-file.

Code block 2 Illustration of a typical XML-tree structure.

```xml
<?xml version="1.0"?>
<root>
    <parent>
        <child attr="val1">text</child>
        <child attr="val2">text2</child>
    </parent>
    <parent>
        <child attr="val3">text</child>
        <child attr="val4">text2</child>
    </parent>
</root>
```

The XML-file might be nested with several childen and parents, but code blocks code block 2 holds for illustration purposes. XPath for the first child in parent will be `./parent/child[@attribute="val1"]`. Here `.` is the root, `/` defines path (or nesting if you like) and `[@attribute="val"]` defines that the attribute named `attr` should be of value `val1`. This XPath will find only the first child of the first parent, but if other childs with same path also had an attribute named `attr` with the value `val1`, the XPath would have found them also. E.g.

`./parent/child` will find all children. Code Block 3 show how one would read properties in the XML-file from code block 2.

Code block 3 Accessing XML properties with the Python build-in module xml.etree.

```python
import xml.etree.ElementTree as ET

tree = ET.parse('/path/to/file.xml')            # read xml
first_child = tree.find('./parent/child')       # find one element
first_child.attrib['attr'] == "val1"            # check attribute value
all_children = tree.findall('./parent/child') # find all elements
len(all_children)                               # number of elements found
```

Scanning Template

A scanning template is a XML-file which defines which regions a scan job exists of. The structure of the file is the following:

- **./ScanningTemplate/Properties** holds experiment settings like start position, displacement between fields and wells, start position, which Z-drive to use, and so on.
- **./ScanFieldArray** holds all fields (images) and their settings as attributes in `./ScanFieldArray/ScanFieldData`.
- **./ScanWellArray** holds all wells (collection of images) and their settings as attributes in `./ScanWellArray/ScanWellData`.

OCR

Optical character recognition (OCR) is recognition of characters in an image. OCR internals are not discussed, but it basically works by looking at patterns in the image to convert it to text.

keep, remove or expand?

Chapter 3

Methods

Microscope

The images were collected with a Leica SP8 microscope using LAS X software version 1.1.0.12420 from Leica Microsystems CMS GmbH. Two lasers was in use, a pulsing Coherent laser and a continious LASOS argon laser. Full specifications of lasers are in table 3.1.

Table 3.1: Lasers

Brand	Model	Specifications
Coherent	Chameleon Vision-S	Modelocked Ti:Sapphire, wavelengths 690-1050 nm, 2500 mW, 80 MHz pulsed, $\approx$ 75 ps pulse width
LASOS	LGK 7872 ML05	Argon Continious wave, wavelengths 458, 476, 488, 496 and 514 nm, 65mW

Transmitted light was measured with non-descanned detectors. The non-descanned detectors was used with dichrioc mirror of 495 nm and band pass filters of 525/50 nm and 445/20 nm. Rotation of scanning mirror was set to 1.7° to align scanning coordinate sytem with stage coordinate system (read more in Stitching).

Images was exported as `.tif` and then compressed to lossless `_web.jpg`[11] to reduce storage space. The images was also rotated 270°, as LAS X store `.tif`-images with axes swapped in regards to the stage axes. The procedure is listed in code block **??**.

```
from leicaexperiment import Experiment
from PIL import Image

experiment = Experiment('path/to/experiment')
experiment.compress(delete_tif=True) # lossless PNG compression

for filename in experiment.images:
    img = Image(filename)
    img = img.rotate(270)           # image axes same as stage axes
    img.save(filename)
```

Overview images

Overview images was taken with an technique similar to bright-field microscopy except that the light source is a scanning laser. A 10x air objective was used, the lasers in use were the argon laser in table 3.1 with 514 nm emission line, output power set to 2.48% and intensity to 0.10. Forward light was imaged using a 0.55 NA air condenser with the non descanned detector having 525/50]) bandpass filter. Aperture and detector gain was adjusted so that the histogram of intensities was in the center of the total range without getting peaks at minimum and maximum values.

Zoom 0.75 and 512x512 pixels with 8 bit depth was chosen, which gives images of $\approx 1500\,\mu m$ and resolution of $\approx 3\,\mu m$.

SHG images

SHG images was taken with a 25x/0.95 NA water objective. The pulsed infrared laser was set to 890 nm, intensity 20%, gain 40%, offset 80% and electro-optic modulator (EOM) on. Forward light was measured with non descanned PMT sensor behind a 0.9 NA air collector. Band pass filter in front of the detector was 445/20 nm and gain of detector was adjusted so that signal spanned the whole intensity range. Aperture was set to 24 (maximum).

A resolution of 1024x1024 pixels with 8 bit image depth was used. Frequency of scanning mirror was set to 600 lines/second.

Automated scanning

The automated scanning aims to lift the burden of manually labor and prevent errors in the imaging process by finding regions with the specimen spots in an overview image. The process consists roughly of the steps:

- Take an overview image with low magnification
- Segment the overview image
- Allow user to confirm or adjust the segmentation
- Scan each region

After overview images was taken, they were equalized and stitched. The equalization step corrects uneven illumination and increases contrast for viewing purposes. To improve robustness of segmentation, a local bilateral population filter was applied to the stitched image before it is thresholded. Each separate region in the segmentation are sorted by their area size, small regions are excluded and the user can exclude or add regions if some of the specimen spots are not detected. Row and column position of the regions are calculated by sorting them by their position in the image. A more detailed description follows.

Uneven illumination

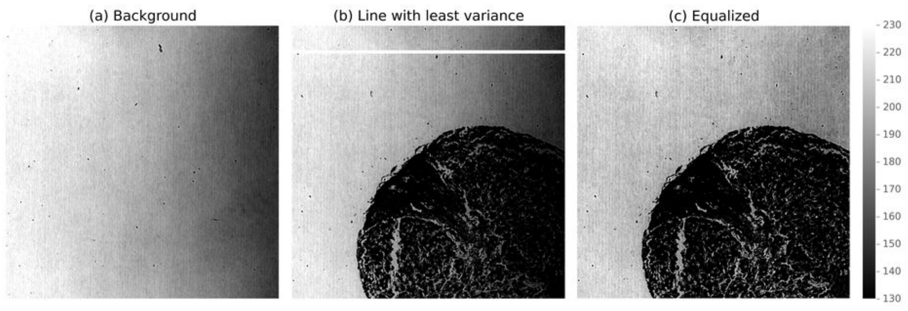


Figure 3.1: (a) Image of glass slide only and no tissue for illustrating the uneven illumination. Dots are impurities on the glass slide. **(b)** Original image with part of specimen spot. The white line is the row with least variance used for equalization. **(c)** Equalized version of (b). Note that (a), (b) and (c) are displaying values from 130 to 230 to highlight the intensity variation, colorbar is shown to the right.

The uneven illumination in the experimental setup is illustrated in figure 3.1(a). By assuming the intensity variation in all pixels are following the slope of the background, equalization was done by dividing each row in the image by the normalized intensity profile of the background.

Code block 4 Equalizing an image

```
equalized = img.astype(np.float)        # assure datatype have real division ability
equalized -= images_minimum             # normalize
equalized /= images_maximum - images_minimum
equalized /= intensity_profile          # equalize
equalized[equalized > 1] = 1            # clip values
```

As seen in code block 4 the image is first normalized. `images_minimum` and `images_maximum` is found by selecting the median of respectively minimum and maximum intensity of all images. By taking the median of all images one avoids outliers and gets the same normalization for all images. Similar technique could be used for normalizing the images after equalization, but clipping gave acceptable results. `intensity_profile` is a curve fit for one of the background rows. The background row was found by selecting the row with least variance (given that the image does have a row with background only). In figure 3.1(b) the row with least variance is indicated with a white line. The same intensity profile is used on all images, and it's fitted to a second degree polynomial to steer clear from noise as illustrated in figure 3.2(a).

The effect on pixel values can be seen in figure 3.2 (b) and (c), where each dot represents a pixel value with increasing image x-position on the x-axis. The intensity variation shown here was directed in one axis only which allowed for the simple divide all rows by the intensity profile. For more complex intensity variations, similiar approach can be done by fitting the two dimentional background to a surface, then divide the image by this intensity surface profile.

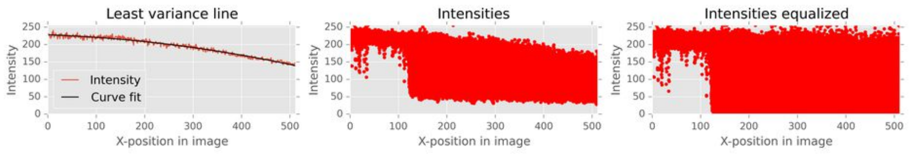
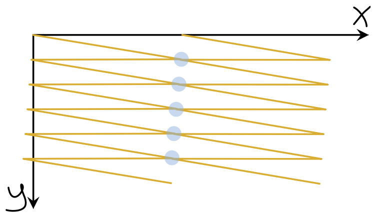


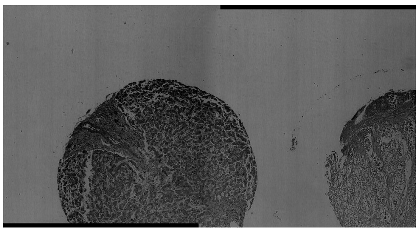
Figure 3.2: (a) Intensities for the line with least variance of figure 3.1(b). The curve is fitted to a second degree polynom to supress noise. **(b)** Intensities for image in figure 3.1(b). Each dot represents a pixel. **(c)** Intensities for the equalized image in figure 3.1(c). Each dot represents a pixel. Note that the intensities is both spread across the whole intensity range (0-255) and the skewness is fairly straightened out.

Rotation

To get good stitches the microscope scanning mirror and the stage should share the same coordinate system. It's not uncommon that it does not, giving the result of a shifted stitch seen in figure 3.3.



(a) Illustration of rotated scanning mirror coordinate system.



(b) Best stitch of two images when stage and scanning mirror does not hold the same coordinate system.

Figure 3.3: Illustrations and stitch of two images with scanning pattern rotated compared to stage movement. In (a) the first row of the first image lines up with second row in second image. The second image should therefor be one pixel above the first image. In (b) relative scanning pattern rotation is counter clockwise, giving the second image below the first image. A calculating of stage position by y-equivalent to equation 3.4 will give a systematic error in the y-position.

Relative rotation between coordinate system was measured by taking two images, finding their displacement with phase correlation and calculating the angle by

$$\theta = \tan \frac{\Delta y}{\Delta x}. \tag{3.1}$$

Here Δy and Δx is the displacement in pixels between images. To align the coordinate systems, scanning rotation was set to $-\theta$ in LAS X.

Stitching

Stitching was done by phase correlating all neighbor images, calculating the median translation and using this median translation between all images. This was done as correlation between two images with little entropy in the seam are

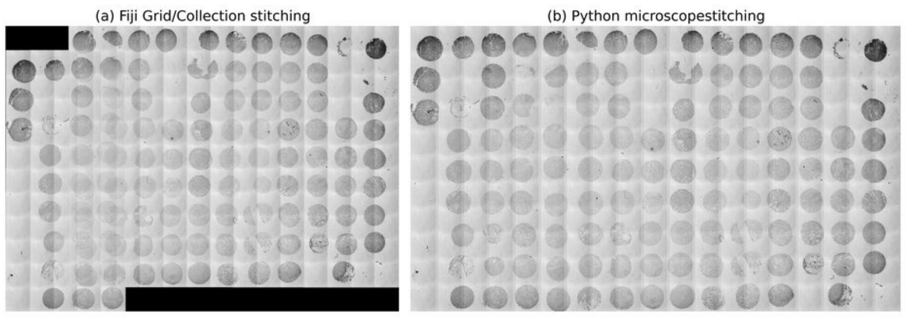

(a) Fiji Grid/Collection stitching
(b) Python microscopestitching

Figure 3.4: **(a)** Unreliable stitching with Fiji, the image translation calculated by phase correlation is chosen without adhering to displacement constraints. **(b)** Using same overlap for all images gives negliable errors, here using the Python package `microscopestitching`.

prone to fail. More details on this matter are in the discussion. Code Block **??** show the basics of the procedure on a row of images for sake of simplicity.

```python
from skimage.feature import register_translation
import numpy as np

# find all neighbor translations
translations = []
prev = row_of_imgs[0]                        # row_of_imgs: list of 2d arrays
for img in row_of_imgs[1:]:                  # exclude first image
    translation, error, phasediff = register_translation(prev, img)
    translations.append(translation)         # add translation to the list
    prev = img                               # reference to previous image
translations = np.array(translations)        # allow for slice notation
offset_y = np.median(translations[:,0])
offset_x = np.median(translations[:,1])
assert offset_x == 0, "x-offset should be zero, " + \\
                      "adjust the scanning mirror rotation"

# combine into one big image
y, x = img.shape         # assume all images are of same size
n = len(row_of_images)
total_height = n*y - offset_y*(n-1)
stitched_img = np.zeros((total_height, x))
for i, img in enumerate(row_of_images):
    y_start = i*y - i*offset_y
    stitched_img[y_start:y_start+y, :] = img
```

Segmentation

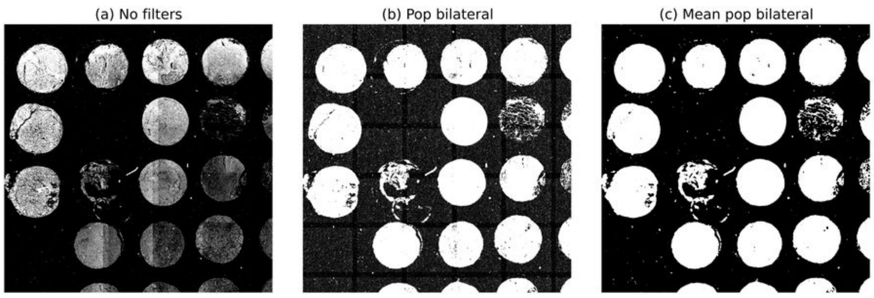


Figure 3.5: Otsu thresholding of figure 3.4(b). **(a)** Otsu thresholding applied without any filters. Picks out dark areas, but disjointed, especially for brighter sample spots in bottom left. **(b)** Thresholding after a local bilateral population filter. Quite noisy in the background. **(c)** Thresholding after local bilateral population and local mean filter. Background noise is gone and sample spots are coherent.

As seen in figure 3.4(b), the samples at the edge are darker than the samples in the center. To improve this intensity variation, the overview image is filtered with a local (reverse) bilateral population filter. Reverse in this sense means that the filter counts number of neighbouring pixels that are outside a specified range. The effect of the filter is a less computational demanding and somewhat similar to an entropy filter. Areas with low signal variation (the background) give low values and areas with high signal variation (the samples) give high values.

To reduce noise after the bilateral population filter, a mean filter was applied. The size of structure elements was 9x9 pixels for both filters. Figure 3.5(a),

(b) and (c) show how the segmentation is affected by the filters. Code for reproducing the steps are in code block 5.

Code block 5 Filter and segment an image with local bilateral population and Otsu thresholding.

```python
from skimage.morphology import square
from skimage.filters import threshold_otsu
from leicaautomator.filters import mean, pop_bilateral

selem = square(9)
filtered = pop_bilateral(image, selem)
filtered = mean(filtered, selem)

threshold = threshold_otsu(filtered)
segmented = filtered >= threshold # high values indicate signal
```

After segmentation, regions was sorted by their area size and only the largest regions are kept. Row and column was calculated by sorting regions by position, measuring the distance between them and increment row or column number when there is a peak in the distance to previous region. The code can be seen in code block 6 and figure 3.6 illustrate typical area size (a), position (b) and position derivative (c).

The whole process of segmentation was done interactive as part of the Python package *leicaautomator*, where settings can be adjusted to improve segmentation and regions can be moved, deleted or added with mouse clicks. The interface is shown in figure 3.7.

Calculate stage position from pixel position

After regions was localized, pixel-size in meters was calculated by

$$x_{resolution} = \frac{\Delta x}{\Delta X}. \tag{3.2}$$

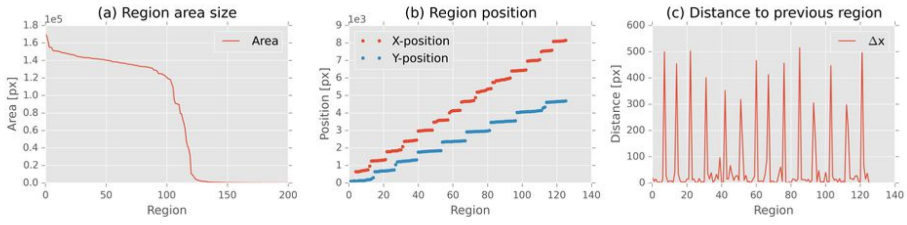
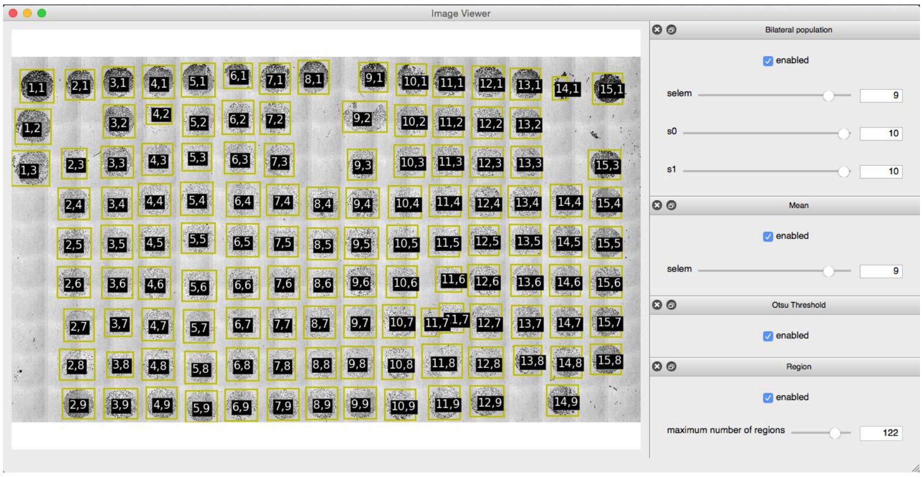


Figure 3.6: **(a)** Sorted region areas. Area size drops dramatically around region 125 according to number of samples on slide. Plot does not have corresponding x-axis with (b) and (c), as regions are sorted by size. **(b)** Regions sorted by position. The two plots do no share the same x-axis. There is a gap between the positions when row and columns are increasing. **(c)** X distance to previous region when regions are sorted by x-position. Same x-axis as in (b) for the x-position plot. 14 peaks indicate that the image contain 15 columns.



Bilateral population
☑ enabled
selem 9
s0 10
s1 10

Mean
☑ enabled
selem 9

Otsu Threshold
☑ enabled

Region
☑ enabled
maximum number of regions 122

Figure 3.7: The process of segmentation in a graphical user interface. Regions 4,2, 11,7 and 14,1 might be adjusted by the user, all other regions are detected fairly well.

Code block 6 Pick out largest regions and calculate row and column position.

```python
from skimage.measure import label, regionprops

labels = label(segmented, background=0) # background=0: exclude background
regions = regionprops(labels)           # measure region properties
regions.sort(key=lambda r: -r.area)     # sort by area size, largest first

max_regions = 126
if len(regions) > max_regions:
    regions = regions[:max_regions]     # only keep max_regions

for r in regions:
    r.y, r.x, r.y_end, r.x_end = r.bbox # for convenience

for direction in 'yx':                  # same algorithm for row and columns
    regions.sort(key=lambda r: getattr(r, direction))

    previous = regions[0]
    for region in regions:              # calc distance to previous region
        dx = getattr(region, direction) - getattr(previous, direction)
        setattr(region, 'd' + direction, dx)
        previous = region
```

Here Δx is displacement in pixels and ΔX is stage displacement in meters read from the overview scanning template in the experiment `AdditionalData/{ScanningTemplate}overview.xml` at XPath `./ScanningTemplate/Properties/ScanFieldStageDistanceX`. Left most left pixel was calculated by

$$X_{start} = X_{center} - \frac{m}{2} \cdot x_{resolution}. \tag{3.3}$$

In equation 3.3 X_{center} is the stage position and m is number of pixels in the image from the overview scan. X_{center} was read from the overview scanning template at XPath `./ScanFieldArray/ScanFieldData[@WellX="1"][@WellY="1"][@FieldX="1"][@FieldY="1`
`/FieldXCoordinate`. The stage x-coordinate for any pixel was then calculated by

$$X = X_{start} + x \cdot x_{resolution}. \tag{3.4}$$

To be able to scan regions of different shape and size, a bounding box for the region was used to calculate the scanning area. Moving the stage to the boundary position will center the boundary in the image, and therefor start position of first image is calculated by

$$X_{start} = X + \frac{\Delta X_{job}}{2}. \tag{3.5}$$

TODO: make clearer, rename X_start

Here, ΔX_{job} is stage displacement between images in the job scanning template. X_{start} will have an error of

$$\epsilon = \frac{1}{2}(\Delta X_{job} - \Delta X_{img}), \tag{3.6}$$

where ΔX_{img} is the total size of the scanned image. This was considered neglectible as $\Delta X_{job} \approx \Delta X_{img}$ and number of columns scanned was calculated by

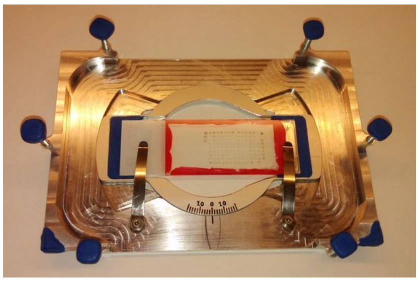
$$f_x = \lceil \frac{\Delta X}{\Delta X_{field}} \rceil. \tag{3.7}$$

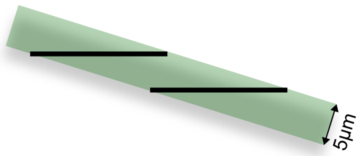
Scanning each region

To avoid unnecessary long stage movements between rows or columns, regions was looped through in a zick-zack pattern, given by their row and column position. For each region the scanning template was edited, the template was loaded and the scan was started through CAM. Single templates was used due to a Leica LAS software limitation; scanning templates with irregular spaced wells can not be loaded. Code Block 7 illustrates the process.

Code block 7 Automated scanning of regions with CAM.

```python
from leicascanningtemplate import ScanningTemplate
from leicaautomator import zick_zack_sort
from leicacam import CAM

cam = CAM() # instantiate connection to microscope

# regions sorted as [r(1,1), r(1,2), r(2,2), r(2,1), r(3,1), r(3,2), ...]
# here r(2,1) is region(col=2, row=1)
regions = zick_zack_sort(regions, ('well_x', 'well_y'))

tmpl_path = r"C:\Users\TCS-User\AppData\Roaming\Leica Microsystems\LAS X" + \
            r"\MatrixScreener\ScanningTemplates" + "\\"
tmpl_name = tmpl_path + '{ScanningTemplate}leicaautomator'
for n, region in enumerate(regions):
    # alternate between tmpl_name0/1.xml, due to a
    # bug LAS cannot load the same name twice
    tmpl = ScanningTemplate(tmpl_name + str(n%2) + '.xml')

    tmpl.move_well(1, 1, region.real_x, region.real_y)
    tmpl.write()

    cam.load_template(tmpl.filename)

    # do an autofocus
    cam.autofocus_scan()
    cam.wait_for('inf', 'scanfinished')

    # run the scan job
    cam.start_scan()
    # record output filename
    region.experiment_name = cam.wait_for('relpath')['relpath']

    # continue with next region when scan is done
    cam.wait_for('inf', 'scanfinished')
```


(a) Sample holder with adjustment of z-plane.



(b) Tilted z-plane of sample seen from the side. Black lines indicate two images and the objective focus for those.

Figure 3.8

Alignment of z-plane

The samples in figure 1.1 are 5 µm thick and keeping the sample plane at same distance from . A movement of 1.7 mm will shift columns by one, which describes the accuruancy required.

Correlating images with patient data

Slide maps, seen in figure 3.9, and patient database was given by St. Olavs. As the slide maps contained circles, slide maps were filtered to remove all but text before it was read with OCR. The OCR text output was checked for errors programatically (id should be of correct format, id should increment, patients should be registered with correct slide in database column `TP_nr`, each patient should have three samples). OCR errors was fixed manually and other errors was recorded (see section Slide map errors in the appendix).

Every pasient id from the slide map was then saved to a stata database along

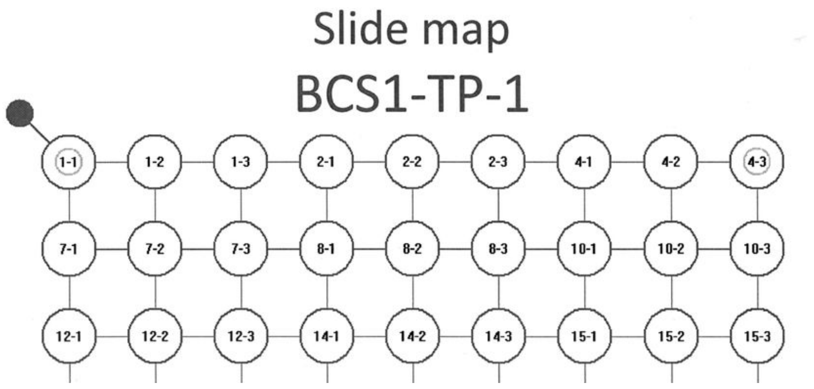


Figure 3.9: Top of slide map TP-1. Ids are not incrementing systematically and need to be registered to correlate samples to respective patients. Ids are inside circles and hard to read with OCR. First part of id is same as `ID_deltaker` in patient database, second number is sample number. There should be three samples for each patient.

with its slide number, row and column. Code Block 8 show how the clinical data was correlated with samples.

Code block 8 Get patient outcome of sample on TP-1 row 3 column 5.

```
import pandas as pd

# read databases
locations = pd.read_stata('data/ids/locations.dta')
clinical_data = pd.read_stata('data/clinic_data.dta')

# position query
condition = (locations.TP_nr == 1) & \
            (locations.TP_rad == 3) & \
            (locations.TP_kolonne == 5)

# get patient id
patient_id = locations[condition]['ID_deltaker']

# check exactly 1 patient registered at given row/col
assert len(patient_id) == 1

# clinical data query
condition = clinical_data.ID_deltaker == patient_id.iloc[0]

# get outcome
outcome = clinical_data[condition]['GRAD']
```

Chapter 4

Discussion

What to communicate: discuss results, limitations, possibilities for improvement

ML: Hvilke valg har blitt tatt, hva er viktig for neste bruker, hva er begrensninger, utviklingsmuligheter, pros/cons, hvor bra fungerer det. . . .)

Scanning

The motivation for automating the microscope scanning process is to help the user of the microscope to achieve his goals with less mental overhead, so the user can use the effort on research instead of repetitive trivial labor. TMA samples can contain up to 1000 samples for each glass slide[5], and helping with organizing the work is vital. The glass slides in discussion here holds 14 columns of specimen spots, which is 60 non-overlapping images with a 25x objective. This means that an operator of the microscope must keep track of the current stage position in the array with limited field of view.

Though the complexity can be handled by a human, the process of manually scanning TMA consist of a lot error prone work. As tissue is somewhat aranged, specimen spots have features which are easy to extract and are suitable for discimination, tools in microscope software exists for scanning large structures, creating automated microscope scanning with image processing becomes a low hanging fruit.

To illustrate the pros of using the method described in this thesis, lets compare it to the manual approach. By using LAS X matrix screener, the procedure will be fairly structured. The manual labor in the scanning would roughly consist of:

1. Count number of rows and columns.
2. Align TMA in microscope.
3. Measure average inter sample displacement.
4. Find the maximum sized specimen spot and measure it's size.
5. Define an experiment holding the correct number of rows, columns, displacement between samples and sample size.
6. Update inter sample offsets one by one.
7. Potetially disable fields on specimen spots with smaller size than the largest.
8. Potentially identify and rule out missing samples.
9. Make sure autofocus positions will hold signal (e.g. specimen spot should be in the autofocus image).
10. Scan.

The procedure was tested out and step 6 was the most labor intensive, browsing through 126 samples aligning them. An alignment of one sample took about 40 seconds, giving 1.5 hours of intensive click-and-adjust. Also, an error in some of the steps can potentially disrupt steps further down the line, making the procedure even more labor intensive. In example, inaccuriancy in average displacement between samples will lead to displacement adjustment of many wells, accidentally bumping the sample holder could impose restart of the procedure, and so on.

A simple means to avoid some of the steps in the intricate procedure above is using a single scan containing the whole matrix area. The procedure then simplifies to:

1. Align TMA in microscope.
2. Find outer boundaries.
3. Create predictive focus map or define autofocus for more or less regular spaced intervals containing a specimen spot.
4. Scan.
5. Separate specimen spots in images and assign row and column to them.

Compared to the first procedure listed, this procedure have the advantage of being less labor intensive when on the microscope, but manually browsing through $24\,\mathrm{mm} \cdot 15\,\mathrm{mm} \;/\; (400\,\mu\mathrm{m})^2 = 2250$ images may be a daunting task without a specialized tool.

The main concern with the last procedure was focus and a couple of scans confirmed the concern by having out of focus portions. The out of focus can be of several reasons, e.g. inter specimen z-displacement or temperature changes moving the specimens in z-direction. As the autofocus in LAS X runs before the scan, the only way to tackle temperature changes is by chopping up the scan in several chunks. As the goal was to reduce manual labor, doing this as a part of the procedure was not considered viable.

In other words, the most likely way to get desired result is by using the first procedure listed. So most part of the procedure was automated and the method described is therefore a combination of the two procedures above. Several of the steps remain the same but automated, so the procedure for the user of the microscope reduces to:

1. Align TMA in microscope.
2. Find outer boundaries for overview scan.
3. Verify that the algorithms have picked out the specimen spots.

- If not, the user may adjust filter settings or directly edit the detected regions.

4. Scan.

This was considered to meet the goals; reduce mental overhead when collecting images from TMA glass slides.

Images and stitching

The LAS X matrix screener has two options for storing images, in `.tif` or `.lif`-format (Leica image format). `.tif` exporting has the advantage that it's fully controllable trough the CAM interface, as LAS X will report filenames of images when scanning. `.tif` is also an ISO standard[12] which most image programs can open. In contrast, with the `.lif`-format the user have to save the images manually in the graphical user interface and the format is not a wide adopted standard. Based on the pros of openness and automatization `.tif` was chosen. None of the formats are readily putted together.

With 10x objective and 0.75 zoom, maximum field of view is equal to 1550 µm. Average specimen spot diameter was $\approx$ 1200 µm. These two facts would allow for imaging specimen spots into separate images if they were neatly arranged.

This was not found out to be true for our dataset, and it would also burden the user of the microscope to measure and define a scan with correct inter specimen displacement. A more robust way is therefor to combine all images into one.

Combining images can be done in interactive manner, where a program loads images as one “moves” around. But creating this abstraction would demand for a way other programs can “talk” to the abstract image object containing all images. Therefor a simpler approach was taken, stitching all images into one large image. This allows for any program that can open `_web.jpg` to work with the images.

First approach on stitching was to use existing stitching software, in specific the *Grid/Collection stitching*-plugin of Fiji[13]. The plugins finds displacements between images by using phase correlation, and it works fairly well except for the lack of control when phase correlation fails. The failing of the phase correlation is mainly due to little entropy in the seam between images. It can be seen in figure 3.4, where the failed row have to much overlap. The failed row is a clean cut in the sense that the overlap between the images contain background only and no specimen. A background surface is quite even and will give a flat correlation in contrast to the wanted peak which express a match is found. In other words, the overlap between the images contain too little information for correlation and the match fails.

In addition, as we want to calculate stage position from the stitched image with[???]

There is two ways to overcome failing

chosen when stitching the overview image. Using the same overlap in this context gives reliable stitching with negligible errors. The overlap is chosen by calculating all overlaps with phase correlation and taking the median. The stitching was put in a Python package and can be used as shown in code block **??**.

The ungraceful failing of the Fiji stitching plugin was a major drawback, as the

Code block 9 Stitching images with the Python package *microscopestitching.*

```python
from microscopestitching import stitch
from glob import glob

files = glob('path/to/images/*')
images = []
for i, file in enumerate(files):
    # rectangle of 4 rows and len(files)//4 columns
    row = i % 4
    column = i // 4
    images.append((file, row, column))

stitched_image = stitch(images)
```

automatic scan relies on finding specimen spots in the overview image.

LAS X comes with a function for this, which draws a line in the image and lets the user adjust the rotation while moving the stage coordinates. A reference point should then follow the line if the scanning mirror and stage holds the same coordinate system. The user himself have to find the rotation in a inductive way by counting pixels or measuring how far the reference point moves away from the line.

A more robust

It would be possible to get all specimen spots in The TMAs available for this thesis was not neatly aranged. This along with

have room for ≈ 350 µm uncertainty in specimen placement in the array. This

When taking the overview images it's hard to make every image contain a complete specimen spot

The `.tif` images are stored in a folder tree with folder for *every* field. For a complete tissue microarray that is a couple of thousand folders, which easily becomes unmanageable if browsing directly. To improve the situation, files were stitched together such manage, one can use the python package `leicaexperiment` to work with files. This means that they have to be combined in some manner,

- observed intensity decline in seam
- ome tif output
- imagej constraints not working
- beside -10, 1, above -10, +2
- inaccurate stage
- fail to register correct translation
- little signal

Segmentation

Stable stage insert

Leica LAS design

- user should be mainly in LAS
- automating on the side as a supplement
- load before CAM can be used
- does not load all settings from XML

Chapter 5

Conclusion

What to communicate: brief summary of the result and discussion, advice for further work

ML: Automatic imaging and segmentation of TMA has been demonstrated)...and....

Chapter 6

Appendix

Python software

The software in this thesis is written in Python due to Python's cross-platform support, simple syntax and vast scientific ecosystem. With Python one gets free access to a lot of scientific software libraries of high quality and top-level support through channels like github. As source code for most libraries are available, stepping into the nitty-gritty details can give insight in algorithms and be very educational.

Any Python package mentioned in the code blocks is install-able through pip. In example `leicacam` can be installed by opening a terminal and type `pip install leicacam`. The computer must have `pip`[14] and the required compilers if the package depends on compiling code. This is true for most of the software, it depends on fast algorithms implemented in compiled languages like C and Fortran.

Compiling the huge scientific libraries like numpy and scipy can take a while, so it's recommended to use a Python distribution like Anaconda???. Anaconda pre-ships with the most common scientific libraries and it also contains the package manager `conda` which have pre-compiled packages available for most operating systems.

Slide map errors

```
 TP2, row  3, col  6 - pasient id missing in db: 66
 TP6, row  1, col  9 - pasient id missing in db: 222
 TP3, row  1, col  3 - id 68, wrong TP_nr in db: 3.0 != 2.0
 TP6, row  1, col  3 - id 209, wrong TP_nr in db: 6.0 != 4.0
 TP6, row  1, col  6 - id 221, wrong TP_nr in db: 6.0 != 5.0
TP22, row  2, col  6 - id 130, wrong TP_nr in db: 22.0 != 3.0
TP22, row  2, col  9 - id 244, wrong TP_nr in db: 22.0 != 5.0
TP22, row  3, col  3 - id 281, wrong TP_nr in db: 22.0 != 6.0
TP22, row  3, col  6 - id 296, wrong TP_nr in db: 22.0 != 6.0
TP22, row  3, col  9 - id 309, wrong TP_nr in db: 22.0 != 6.0
TP22, row  4, col  3 - id 318, wrong TP_nr in db: 22.0 != 6.0
TP22, row  4, col  6 - id 376, wrong TP_nr in db: 22.0 != 7.0
TP22, row  4, col  9 - id 396, wrong TP_nr in db: 22.0 != 8.0
TP22, row  5, col  3 - id 413, wrong TP_nr in db: 22.0 != 8.0
TP22, row  5, col  6 - id 449, wrong TP_nr in db: 22.0 != 9.0
TP22, row  5, col  9 - id 453, wrong TP_nr in db: 22.0 != 9.0
TP22, row  6, col  3 - id 487, wrong TP_nr in db: 22.0 != 10.0
TP22, row  6, col  6 - id 493, wrong TP_nr in db: 22.0 != 10.0
TP22, row  6, col  9 - id 525, wrong TP_nr in db: 22.0 != 10.0
TP22, row  7, col  3 - id 728, wrong TP_nr in db: 22.0 != 15.0
 TP3, row  9, col  6 - TP_nr not registered in db for ID_deltaker 140
 TP5, row  9, col  9 - TP_nr not registered in db for ID_deltaker 251
 TP9, row 10, col  9 - there should be 3 samples: ['467a-1']
 TP9, row 11, col  3 - there should be 3 samples: ['467b-1', '467b-2'])
 TP9, row 12, col  6 - there should be 3 samples: ['471a-1', '471a-2']
 TP9, row 12, col  9 - there should be 3 samples: ['471b-1']
TP10, row  8, col  6 - there should be 3 samples: ['507-1', '507-2']
TP10, row 12, col  6 - there should be 3 samples: ['525-2', '525-3']
TP11, row 11, col  6 - there should be 3 samples: ['566-1', '566-2']
 TP3, row  1, col  3 - pasient id did not increment: ['68-1', '68-2', '68-3'] <
                                                     ['102b-1', '102b-2', '102b-3']
 TP4, row  1, col  3 - pasient id did not increment: ['162a-1', '162a-2', '162a-3']
                                                     ['163-1', '163-2', '163-3']
 TP6, row  1, col  3 - pasient id did not increment: ['209-1', '209-2', '209-3'] <
                                                     ['268-1', '268-2', '268-3']
TP11, row  6, col  3 - pasient id did not increment: ['549-1', '549-2', '549-3'] <
                                                     ['552-1', '552-2', '552-3']
TP22, row  2, col  6 - pasient id did not increment: ['130-1', '130-2', '130-3'] <
                                                     ['3067-1', '3067-2', '3067-3']
```

References

1. Statistisk sentralbyrå. *Folkemengde, 1. Januar 2015*. Statistisk sentralbyrå; 2015. http://www.ssb.no/befolkning/statistikker/folkemengde/aar/2015-02-19. Accessed May 21, 2015.

2. Statistisk sentralbyrå. *Dødsårsaker, 2012*. Statistisk sentralbyrå; 2013. http://www.ssb.no/helse/statistikker/dodsarsak/aar/2013-11-01. Accessed May 21, 2015.

3. Brabrand A, Kariuki II, Engstrøm MJ, et al. Alterations in collagen fibre patterns in breast cancer. a premise for tumour invasiveness? *APMIS*. 2015;123(1):1-8. doi:10.1111/apm.12298.

4. Python Software Foundation. The official home of the python programming language. Python.org. 2015. https://www.python.org/. Accessed May 22, 2015.

5. Kononen J, Bubendorf L, Kallionimeni A, et al. Tissue microarrays for high-throughput molecular profiling of tumor specimens. *Nat Med*. 1998;4(7):844-847. doi:10.1038/nm0798-844.

6. Leica Microsystems CMS GmbH. Leica TCS SP8. 2014. http://www.leica-microsystems.com/fileadmin/downloads/Leica%20TCS%20SP8%20STED%203X/Brochures/Leica%20TCS%20SP8-Brochure_EN.pdf. Accessed

May 22, 2015.

7. Murphy DB, Davidson MW. *Fundamentals of Light Microscopy and Electronic Imaging.* Hoboken, N.J.: Wiley-Blackwell; 2013.

8. Gonzalez RC, Woods RE. *Digital Image Processing.* Vol. 3 edition. Upper Saddle River, N.J: Prentice Hall; 2007.

9. Frank Sieckmann. CAM documentation, MatrixScreener 3 & 4. November 2013.

10. Arve Seljebu. Arve0/leicacam. GitHub. 2015. https://github.com/arve0/leicacam. Accessed May 22, 2015.

11. Duce D. Portable network graphics (PNG) specification (second edition). November 2003. http://www.w3.org/TR/PNG/. Accessed May 28, 2015.

12. ISO. Tag image file format for image technology (TIFF/IT) ISO 12639:2004. May 2004. http://www.iso.org/iso/catalogue_detail.htm?csnumber=34342. Accessed May 28, 2015.

13. Fiji is just ImageJ. May 2015. http://fiji.sc/wiki/index.php/Fiji. Accessed May 29, 2015.

14. Python Packaging Authority. User guide - pip 7.0.1 documentation. May 2015. https://pip.pypa.io/en/stable/user_guide.html. Accessed May 27, 2015.