

# Web-Based Osteoarthritis-Analysis

## Generating Data from Native Libraries and Machine-Learning Models

BACHELORARBEIT

zur Erlangung des akademischen Grades

**Bachelor of Science**

im Rahmen des Studiums

**Medizinische Informatik**

eingereicht von

**Lukas Maximilian Masopust**

Matrikelnummer 01427382

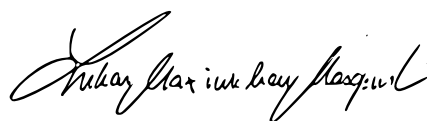
an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Mitwirkung: Zsolt Bertalan, Ph.D.

Wien, 8. Juli 2018



Lukas Maximilian Masopust

Eduard Gröller



# Web-Based Osteoarthritis-Analysis

## Generating Data from Native Libraries and Machine-Learning Models

### BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

### Bachelor of Science

in

### Medical Informatics

by

**Lukas Maximilian Masopust**

Registration Number 01427382

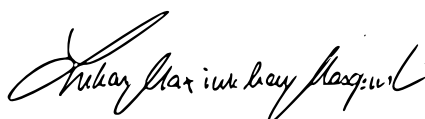
to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Assistance: Zsolt Bertalan, Ph.D.

Vienna, 8<sup>th</sup> July, 2018



Lukas Maximilian Masopust

Eduard Gröller



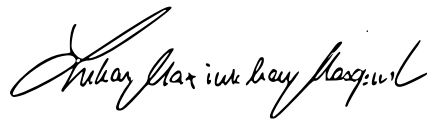


# Erklärung zur Verfassung der Arbeit

Lukas Maximilian Masopust  
Strudlhofgasse 14/5, 1090 Wien, Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. Juli 2018



---

Lukas Maximilian Masopust



# Danksagung

An dieser Stelle möchte ich meine Dankbarkeit denjenigen zum Ausdruck bringen, ohne die diese Arbeit nicht möglich gewesen wäre. Zunächst möchte ich Richard Ljuhar und den großartigen Menschen bei ImageBiopsy Lab dafür danken, dass sie mich so warmherzig in ihr Team aufgenommen und mir damit die Möglichkeit geschaffen haben, meine Arbeit in einem derart spannenden und zukunftssträchtigen Bereich der Informatik zu schreiben. Weiters gebührt Zsolt Bertalan dafür mein Dank, dass er trotz der vielen Fragen, die sich über den Lauf des Projektes ergeben haben, immer geduldig zugehört und mich bestmöglich bei Problemstellungen unterstützt hat. Weiters möchte ich mich auch bei Alexander Krumböck dafür bedanken, dass er sowohl immer ein offenes Ohr, als auch ein Auge auf das Projekt hatte.

Eine weitere Person, der ich meine Dankbarkeit zum Ausdruck bringen möchte, ist mein Betreuer Eduard Gröller für die unkomplizierte und nette Beratung was Fragestellungen zur Arbeit angeht und natürlich dafür, dass ich meine Arbeit unter seiner Aufsicht schreiben durfte. Überdies möchte ich mich auch bei meinem Kommilitonen und langjährigen Freund David Bauer für die grandiose Zusammenarbeit und die lustige Zeit bedanken. Schließlich gebührt mein Dank auch meinen Eltern Margit und Rainer Masopust, sowie meinem Bruder Paul für die langjährige Unterstützung auf zu vielen Arten, als dass man sie hier angemessen würdigen könnte.

Daher möchte ich die folgenden Worte an die hier oben gelisteten Personen, sowie an jene, die hier nicht explizit erwähnt wurden, richten: Vielen herzlichen Dank.



# Acknowledgements

I want to express my gratitude to all the people without whom this thesis would not have been possible. First I would like to thank Richard Ljuhar and the amazing people at ImageBiopsy Lab for integrating me so warmly into their team and giving me the opportunity to write my thesis in such an interesting and promising field of computer science. Further, I want to especially thank Zsolt Bertalan for patiently answering the many questions I have encountered over the course of my project and for always providing kind and helpful answers. I also want to extend my gratitude to Alexander Krumböck for always having an open ear and for overseeing the process of this project as a whole. Another person whom I want to thank is my advisor, Eduard Gröller, for the straightforward, uncomplicated and kind guidance throughout this thesis and of course for letting me write this text under his supervision.

Further, I want to thank my fellow student and close friend David Bauer for the great collaboration and for the laughs we shared during the course of this project.

Finally my sincerest gratitude also goes to my parents Margit and Rainer Masopust, as well as to my brother Paul for their continued support in so many ways that this paragraph could not do justice to honor meaningfully.

So to all the people listed above and to the many more I did not mention explicitly I want to say: Thank you so much.



# Kurzfassung

Durch die rasante Entwicklung von Künstlicher Intelligenz (KI) erweitert sich deren Anwendungsgebiet nahezu täglich. Eine der vielen Gebiete, in welchem KI ein besonderes Potential birgt, ist deren Einsatz im medizinischen Sektor. Aufgrund der weiten Verfügbarkeit von Rechenleistung können, auf neuronalen Netzen basierende Algorithmen, weit mehr Daten generieren als noch vor einer Dekade.

Traditionelle Bildverarbeitungsalgorithmen stützen sich oftmals auf klassische Computer Vision (CV)-Algorithmen wie Kantendetektion, um Strukturen in Pixeldaten zu erkennen. Während diese Methodik in der Vergangenheit respektable Ergebnisse erzielte, werden wir sehen, dass KI die Präzision derartiger Analysen signifikant erhöht.

Den Fokus dieser Arbeit stellt die Generierung von Daten auf Basis eines Röntgenbildes des Kniegelenks dar. Die von ImageBiopsy Lab (IB Lab) entwickelten Methoden benutzten vielfach CV-basierte Algorithmen um Arthrose in Kniegelenken zu erkennen. Obwohl dies bisher gute Resultate erzielte, wird diese Arbeit zeigen, dass der Einsatz von neuronalen Netzwerken nicht nur die Treffsicherheit erhöht, sondern auch weitere Informationen, wie etwa die Lateralität des zu befundenden Knies, feststellen kann.

Das Bestreben dieses Projekts war es, die Geschwindigkeit und Präzision der Arthrofindung in Kniegelenken zu erhöhen und die Verfügbarkeit durch Implementierung in eine Web-basierte Lösung zu steigern. Um die Vorteile unserer Methode zu demonstrieren, befindet sich, zum Zeitpunkt dieses Schreibens, unsere Software im Rollout in einem niederösterreichischen Krankenhaus.

Aufgrund der oben genannten Weiterentwicklungen setzt sich diese Arbeit zum Ziel, einen Überblick über die Beschreibung und den Vergleich von Methoden zur Beschaffung von Information, auf Basis von Röntgenbildern, zur Analyse der Kniegelenksarthrose zu bieten.





# Abstract

As **artificial intelligence (AI)** progresses with seemingly unstoppable speed, its wide field of applications broadens by the day. One area where **AI** advancements appear to be especially promising is their employment in the medical sector. Nowadays, due to the wider availability of processing power, algorithms based on neuronal networks can be used to generate far more data in areas where it previously seemed unthinkable.

Traditional image-processing-algorithms often utilize **computer vision (CV)**-algorithms such as edge-detection to generate data from pixel input. While this method of gaining data worked well in the past, **AI** can help to improve the precision of such an analysis. The area I focussed on in this thesis is the generation of data from x-ray images of the knee joint. **ImageBiopsy Lab (IB Lab)**'s algorithms relied heavily on **CV**-based analysis for the diagnosis of **osteoarthritis (OA)** in the knee. While this yielded good results in the past, this work will show that the use of deep neuronal networks improves accuracy in a significant way.

Further, neuronal networks can provide additional information that was a lot harder to be gained before, such as the laterality of a given image.

The aim of this project was to diagnose **OA** faster and more precisely than in the past and to embed it into a web-based solution for broader accessibility. To showcase the benefits of the described method, at the time of writing, our software is in the stage of being rolled out in a hospital in Lower Austria.

Because of the advancements mentioned above, this work will focus on the description and comparison of gaining information from x-ray images for a meaningful and efficient diagnosis of **OA** in the knee.



# Contents

<b>Kurzfassung</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Contents</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Definitions . . . . .	1
1.2 Overview . . . . .	2
1.3 Problem Statement . . . . .	2
1.4 Contribution . . . . .	3
1.5 Project Structure . . . . .	3
1.6 Structure of this Work . . . . .	4
1.7 Disclaimer . . . . .	4
<b>2 Methodology</b>	<b>5</b>
2.1 Core Technologies . . . . .	5
2.2 DICOM . . . . .	6
2.3 Native Libraries . . . . .	7
2.4 Native Library Interface . . . . .	7
2.5 Machine Learning Background . . . . .	8
<b>3 Implementation</b>	<b>9</b>
3.1 Overview . . . . .	9
3.2 Communication with Shared Objects . . . . .	10
3.3 DICOMConverter . . . . .	11
3.4 DICOMKneeAnalyzer . . . . .	13
3.5 Kellgren-Lawrence-Score and Report . . . . .	15
<b>4 Evaluation</b>	<b>19</b>
4.1 Performance . . . . .	19
4.2 Accuracy . . . . .	19
4.3 General Comparison of our Solutions . . . . .	22

<b>5 Discussion</b>	<b>23</b>
5.1 Performance . . . . .	23
5.2 Accuracy . . . . .	24
<b>6 Conclusion</b>	<b>25</b>
6.1 Summary . . . . .	25
6.2 Future Expansions . . . . .	25
<b>List of Figures</b>	<b>27</b>
<b>List of Tables</b>	<b>29</b>
<b>List of Algorithms</b>	<b>31</b>
<b>Bibliography</b>	<b>37</b>

# Introduction

## 1.1 Definitions

In order to better understand the goal of our project, we provide the following definitions related to the core of our software.

**Osteoarthritis (OA):** OA is the most common joint disorder worldwide and is found in the majority of patients that are older than 65 years of age. OA is rated as the number three most common reason for reduction of quality of life that is disease-related [GG] [WP03].

Generally speaking, OA is the degradation of a joint caused by mechanical and genetic factors. Its symptoms include typical signs of inflammation like pain, stiffness, and loss of mobility, which are factors for significant functional impairments of an OA-afflicted knee joint [GG] [Neo12].

**Kellgren-Lawrence-Score (KL-Score):** The KL-Score, is a way of measuring the severity of OA. With 0 being the lowest and 4 being the highest grade, this score represents the sum of four sub-scores, that evaluate the following parameters:

- **Joint Space Width (JSW):** The distance between the femoral and tibial bone of a knee joint.
- **Osteophyte formation:** An osteophyte is the formation of new bone material, which in OA usually occurs on the edges of the femur or the tibia. It is thought that osteophytes are the body's response to the change in load through, e.g., the deformation of the joint. Therefore, the presence of osteophytes is considered a sign of OA [HIK<sup>+</sup>17].
- **Subchondral sclerosis:** Sclerosis of a bone is the hardening of bone tissue due to strain and load, which is why sclerotic bones are another sign of OA [LYG<sup>+</sup>13].

## 1. INTRODUCTION

- Deformation: As the name indicates, deformation is simply the structural misalignment of, in this case, femur, tibia, and fibula that build up the knee joint [WKW14].

### 1.2 Overview

Since we have seen that the prevalence of OA in society is very high, IB Lab's aim was to assist doctors in quickly assessing the disease before symptoms even occur. Therefore, a stand-alone desktop application, the IB Lab Analyzer, was developed, presenting a novel approach for knee assessment. As can be seen in Figure 1.1, an x-ray image is loaded, analyzed and presented in a GUI. The generated overlay shows where the medial and lateral landmarks were found and calculates the JSW based on the found contour-points. From this and further analysis, the KL-Score and its sub-scores are calculated.

For its analysis, the IB Lab Analyzer relied heavily on typical CV-algorithms to find said landmarks and on the use of edge-detection in order to find the outlines of the respective bones. We will see, that our new approach, utilizing deep neuronal networks (DNNs) yields significantly better results.

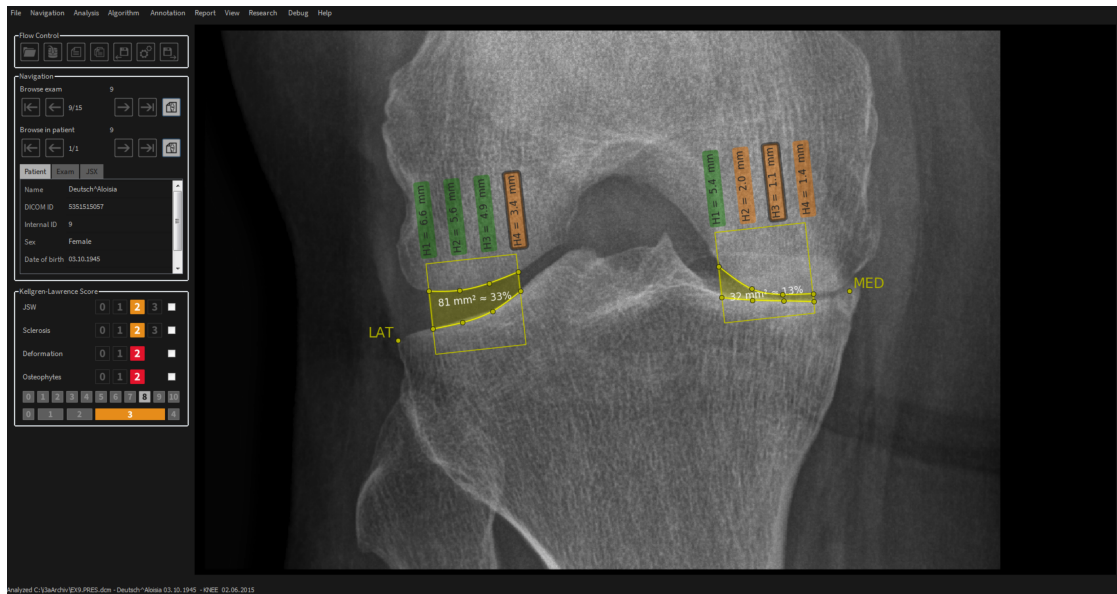


Figure 1.1: Stand-alone IB Lab Analyzer [IBL]

### 1.3 Problem Statement

The stand-alone application discussed before had its limitations, for example, that it had to be distributed to customers via local installations and therefore needed great effort to maintain. In order to not only improve our distribution and update infrastructure, but also to broaden IB Lab's potential customer base, we came up with the solution of

bringing [OA](#)-analysis-techniques to the web. To achieve this goal we used a Java-servlet based solution that could easily be integrated into existing viewers that are used in clinics today.

## 1.4 Contribution

Typical assessment of [OA](#) is done by evaluating x-ray images by a medical doctor. Since this procedure can be time consuming and therefore expensive, the aim of our project was to bring computer-supported [OA](#)-analysis to the web. Through this approach, our aim is to enable a large group of people interested in our solution to assess [OA](#) in a faster and more efficient manner.

A showcase of our project existed beforehand, utilizing a file-interface between a Java-Servlet and Python code that communicated with native libraries and wrote its calculations back into files. As one might expect, this method, because of its many interfaces, had considerably low response times and hence quite a poor frame-rate, which is why the aim of this project was to rewrite our so-called [Image Analysis Server \(IAS\)](#) from scratch. Additionally, our new implementation uses new native libraries that rely on [DNNs](#) for their output. As we will see in the results section, this yielded much better results in detecting knees, its landmarks, and contour-points.

## 1.5 Project Structure

The implementation and development of the [IAS](#) was done as a project with all steps of project management in mind. Our project team included my colleague David Bauer and me as software engineers and Alexander Krumböck as our supervisor. The deployment was done in cooperation with Armin Kanitsar from Visuapps [\[VIS\]](#), who developed a certified medical viewer that our customer, the Landeskliniken-Holding [\[LKN\]](#), uses.

At first meetings were arranged where the specifications and requirements for this project were assessed and agreed upon. Then the engineering team received an introduction to the mentioned showcase-project, whereupon development started, utilizing SCRUM as software development framework. Sprints were done on a weekly basis and the development as a whole was completed in a month. Deployment and testing was then done in accordance with Mr. Kanitsar and under the continued supervision of Mr. Krumböck.

My task and contribution to this project was the development and testing of the generation of data. This ranged, as we will discuss in the following chapters, from interfacing with native libraries to the calculation of the [KL-Score](#) itself. Further, my task was the implementation of DICOM-decompression, as well as correspondance with Mr. Kanitsar.

### 1.6 Structure of this Work

In this paper we will further address methodology, technical implementation, evaluation and results, discussion, conclusion, and the future outlook of our project.

### 1.7 Disclaimer

Because this thesis was written in cooperation with a company, we are not able to disclose every inner working to the fullest extend, but will give sufficient examples and explanations where needed.

All x-ray images used in this paper were gathered from the Multicenter-Osteoarthritis-Study (MOST) [SNG<sup>+</sup>13]. The MOST is a long-term study that was conducted from 2003 to 2010 and contains 3026 images of knee joints and their associated diseases.



# CHAPTER 2

## Methodology

In this section we will describe the fundamental methods used in order to develop [IAS](#). This includes the used technologies, as well as the image-format, interfacing with our native libraries, and a short background on our [machine learning models \(ML-models\)](#).

### 2.1 Core Technologies

The selection of core technologies was mostly determined by what was validated at the time of this project. Because [IB Lab](#) went through the process of being certified as a medical devices maker by ISO 13485 [\[ISO\]](#), most technologies were chosen because those have already been validated. Therefore the core programming language used is Java 1.8. Further, the IDE for this Project was Netbeans 8.2.

The deployment-infrastructure was an Ubuntu 16.04 Server, which is why our native libraries were compiled as [shared objects \(SOs\)](#) with the respective g++-version in mind.

## 2.2 DICOM

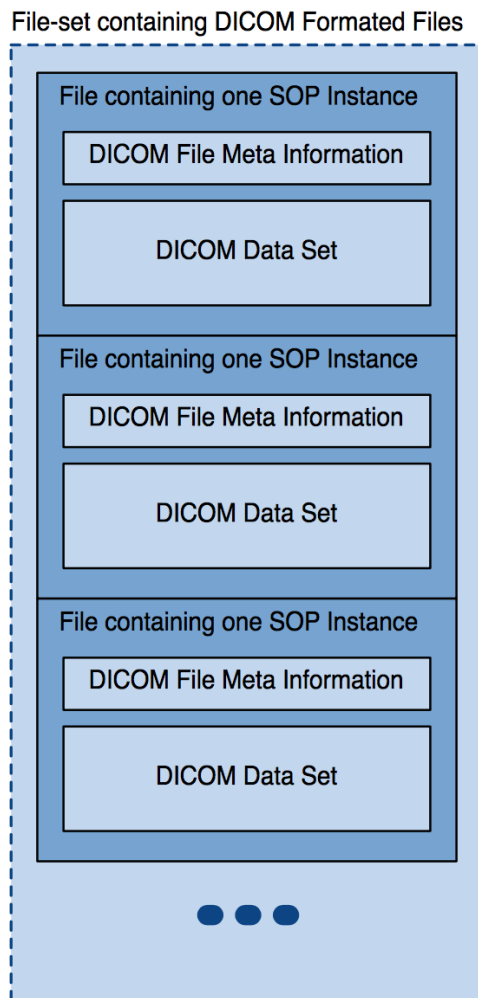


Figure 2.1: DICOM file diagram [DCM]

Since medical applications rely heavily on the standard for medical images, [Digital Imaging and Communications in Medicine \(DICOM\)](#), we will give a short introduction to the format and its properties here. [DICOM](#) is a container format for medical images and patient information developed by the DICOM Standard Committee. [DICOMs](#) therefore contain single or multiple images in a number of formats, including widely used ones such as JPEGs or PNGs. Further, [DICOMs](#) contain data objects that consist of meta-information, which incorporate a multitude of either clinically or technically relevant information. This meta-information contains anything from patient-specific data, treatment-specific data, or - especially relevant for integration into our [IAS](#) - image-specific data, such as compression, photometric interpretation, or the pixel-size of an image. For accessing the [DICOM](#) infrastructure within Java, the open-source library [dcm4che 3.3.7](#) was used.

## 2.3 Native Libraries

Native libraries are code libraries that can be called from within languages such as Java and are written in another language such as C or C++. It is shown that native code is generally faster than Java code, since native code does not need the [Java Virtual Machine \(JVM\)](#) to operate [\[Hun11\]](#). Apart from performance, the interchangeability of native libraries played a major part in considering them as a code basis for [IB Lab](#)'s solutions. This ensures, that if there were changes in one of the calculations itself, only a single module would need to be exchanged on all software solutions, making maintenance much easier.

## 2.4 Native Library Interface

Because of the requirement to use native libraries, one obstacle was to interface them correctly in our [IAS](#), which we will discuss in this section. Java itself provides a programming framework called [Java Native Interface \(JNI\)](#) that enables calling methods written in native code from within a Java-application (Figure [2.2](#)).

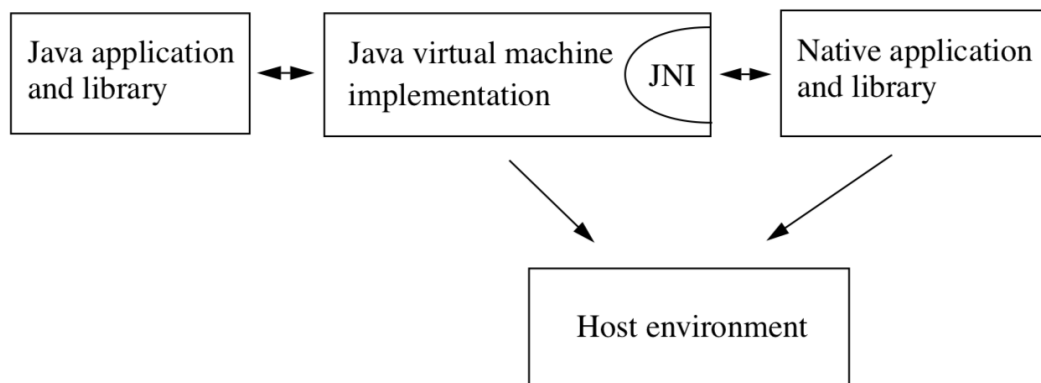


Figure 2.2: Role of JNI [\[Lia11\]](#)

One of the challenges [JNI](#) presents is that [JNI](#) requires native libraries to be compiled in a special way, which means that methods would have to be re-written in order to be accessible from Java code. Since one of the aims of our native library interface was its easy interchangeability and the possibility to use it in different software products, re-writing the C++ code in order to work with Java did not present itself as a viable solution.

Therefore, [IAS](#) utilizes [Java Native Access \(JNA\)](#), which is a library under the Apache Software License, that allows for implementing native libraries without writing additional native code, thereby enabling easier access to the functionality of our native libraries. While this approach provides better accessibility, one has to bear in mind that [JNA](#) has

more per-call overhead, which decreases performance [GRS<sup>+</sup>13]. Still, as we will see in the results chapter, utilizing JNA for native library method-calls yielded performant results, even when data is represented in a GUI with user interaction.

### 2.5 Machine Learning Background

IB Lab utilizes a number of different ML-models for its OA-analysis. Currently in our IAS, models for the following purposes are used:

- Knee-Detection
- Landmark-Placement
- Contour-Point-Placement

At this date, further ML-models are under active development and incorporated in other products. We will discuss those that are close to deployment in Chapter 6 (Future Expansions).

# Implementation

This section contains relevant information about the implementation process. Code snippets will be shown where necessary and the technical approach will be discussed.

## 3.1 Overview

For our first deployment environment in a clinic in Lower Austria, a website that is connected to a PACS-system is used as the entry point to [IAS](#). From there, a web-viewer is loaded, containing our analysis-functionality. As can be seen in [Figure 3.1](#), our software automatically finds the knee joint, predicts where the medial and lateral landmarks are, and displays an overlay with the relevant [JSW](#)- and [Joint Space Area \(JSA\)](#)-calculation-mask. All calculated points can be moved, if any correction should be needed, which triggers a live re-calculation of all points. Additionally, as can be seen in [Figure 3.2](#), the [KL-Subscores](#), as well as the [KL-Score](#) itself are displayed for the analyzed image. As the values are suggestions based on algorithms, the user is still able to overrule every score by manually clicking on the respective number.

Since the main focus of my work was the generation of data from a given x-ray image, the next section will not focus primarily on the visualization in a GUI, but more on the generation of the underlying data itself.

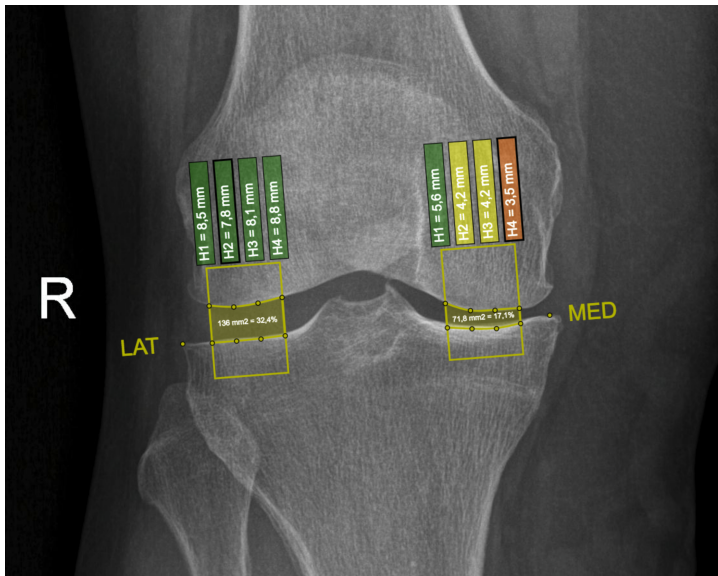


Figure 3.1: Analyzed joint with mask-overlay

Kellgren-Lawrence-Score				
JSW				
0	1	2	3	<input type="checkbox"/>
Sclerosis				
0	1	2	3	<input checked="" type="checkbox"/>
Deformation				
0	1	2		<input checked="" type="checkbox"/>
Osteophytes				
0	1	2		<input checked="" type="checkbox"/>
Sum				
0	1	2	3	4
5	6	7	8	9
10				
Grade				
0	1	2	3	4
<button>Download Report</button>				

Figure 3.2: KL-Score

## 3.2 Communication with Shared Objects

As discussed in Chapter 2, `JNA` was used to interface with `SOs`. In order to call our native methods, we created an interface that contains a static instance for each loaded `SO`. Method stubs are added to the interface, which represent the functions found within the native library. `JNA` detects these stubs and injects callable native code upon runtime. From there, all native methods can be accessed via native types. Types like `ints`, `floats`, or `booleans` can be used as native Java-types, but for reference types, such as `Arrays`, `JNA` provides `Pointers`. An example of such an interface is provided below (Listing 3.1).

```
public interface ExampleNativeInterface extends Library {

    public static final ExampleNativeInterface INSTANCE = (
        ExampleNativeInterface) Native.loadLibrary("example.so",
            ExampleNativeInterface.class);

    public Pointer calculate(Pointer p, int i);

}
```

Listing 3.1: Example interfacing of a native library

Because the creation of `Pointers` is oftentimes essential when working with native code, we will demonstrate the usage of an example `JNA-Pointer` with the `calculate`-method created above, in Listing 3.2 below. As we can see from both snippets, `calculate` also returns a `Pointer`. In order to correctly interpret this `Pointer`, the used native-type has to be known, which is why the in- and out-types of all methods need to be agreed upon beforehand. For demonstration purposes, we will assume that in this case the returned `Pointer` points to an `Array` that holds `float`-values. The equivalent Java-type would be a `float[]`. So in order to read the contents of our `JNA-Pointer`, `JNA` provides methods for all Java-types that can be mapped to this `Pointer`. In this case, we therefore use the `getFloatArray`-method, that, since we work with `Pointers`, needs an offset and the size of the returned array as parameters. Thus, the actual size of the returned `Array` is another important convention to be made, since invalid lengths could lead to crashes due to invalid memory accesses.

```
Pointer example_pointer = ExampleNativeInterface.INSTANCE.  
    calculate(p, 1);  
Pointer return_pointer = example_pointer.getPointer;  
  
int offset = 0;  
int arraySize = 10;  
Float[] result = return_pointer.getFloatArray(offset, arraySize  
    );
```

Listing 3.2: Example usage of a `JNA-Pointer`

## 3.3 DICOMConverter

In order to use the raw pixel-data that a `DICOM`-image holds, oftentimes a few alterations need to be made to further use the image.

### 3.3.1 Photometric Interpretation

The photometric interpretation of a `DICOM` states the intended interpretation of each pixel in an image `[PHO]`.

#### **MONOCHROME1 and MONOCHROME2**

`MONOCHROME1` and `MONOCHROME2` both are photometric interpretations representing pixel-data as a single monochrome plane. The difference being, that the lowest sample value in `MONOCHROME1` is displayed as white, whereas it would represent black in `MONOCHROME2`.

Since this interpretation is essential for further calculations based on the pixel-data, all

MONOCHROME1 images need to be converted. This conversion is done by a very simple algorithm, as seen in Algorithm 3.1. In a foreach-loop every int-value is NOTted in order to reverse the pixel interpretation.

---

**Algorithm 3.1:** Monochrome conversion

---

```
1 foreach pixel p in the image do  
2   | int[] return = ~p.getInts(Tag.PixelData);  
3 end
```

---

### 3.3.2 Transfer Syntax

The transfer syntax of a **DICOM** holds information on the encoding of the incorporated pixel-data. Since some of those transfer syntaxes indicate compressed image formats, they need to be decompressed before further usage.

Decompression in dcm4che 3 is fairly straightforward, since the library provides a decompressor, taking in a dataset and the respective **transfer-syntax-uid (TSUID)** (Listing 3.3). A complete list of every **TSUID** is also available online **[TSU]**.

Since medical data needs to be as accurate as possible and any potential lossy compression could lead to the misinterpretation of important information, mostly uncompressed formats are used.

```
//InputStream is provided upon initialization of our IAS  
BufferedInputStream bIn = new BufferedInputStream(is);  
DicomInputStream dIn = new DicomInputStream(bIn);  
dIn.setIncludeBulkData(DicomInputStream.IncludeBulkData.URI);  
dicom_dataset = dIn.readDataset(-1, -1);  
Object pixeldata = dicom_dataset.getValue(Tag.PixelData);  
  
if (pixeldata != null) {  
    if (pixeldata instanceof Fragments) {  
        Decompressor.decompress(dataset, metadata.  
            getString(Tag.TransferSyntaxUID));  
    }  
}
```

Listing 3.3: Decompression in dcm4chee 3



### 3.4 DICOMKneeAnalyzer

The `DICOMKneeAnalyzer` is the class used to gather all relevant information for later calculation of the `KL-Score` and therefore handles all native library interaction. Hence, after the potential conversion of our pixel-data via the `DICOMConverter`, all necessary calculations from each native library are invoked. Those calculations include the following:

- Medial and Lateral Landmarks
- Contour-points
- `JSA`
- `JSW`

To give a better understanding of what those computations actually represent, Figure 3.3 and Figure 3.4 give a visual overview of the necessary parameters used for the calculation of the `KL-Score`.

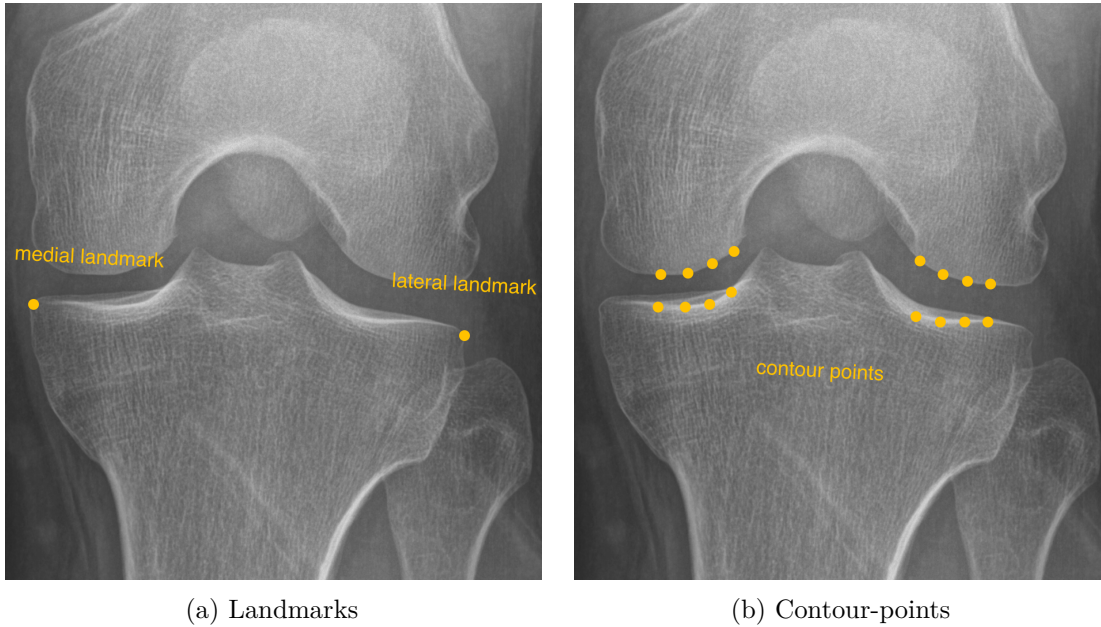


Figure 3.3: Focus points of a knee joint

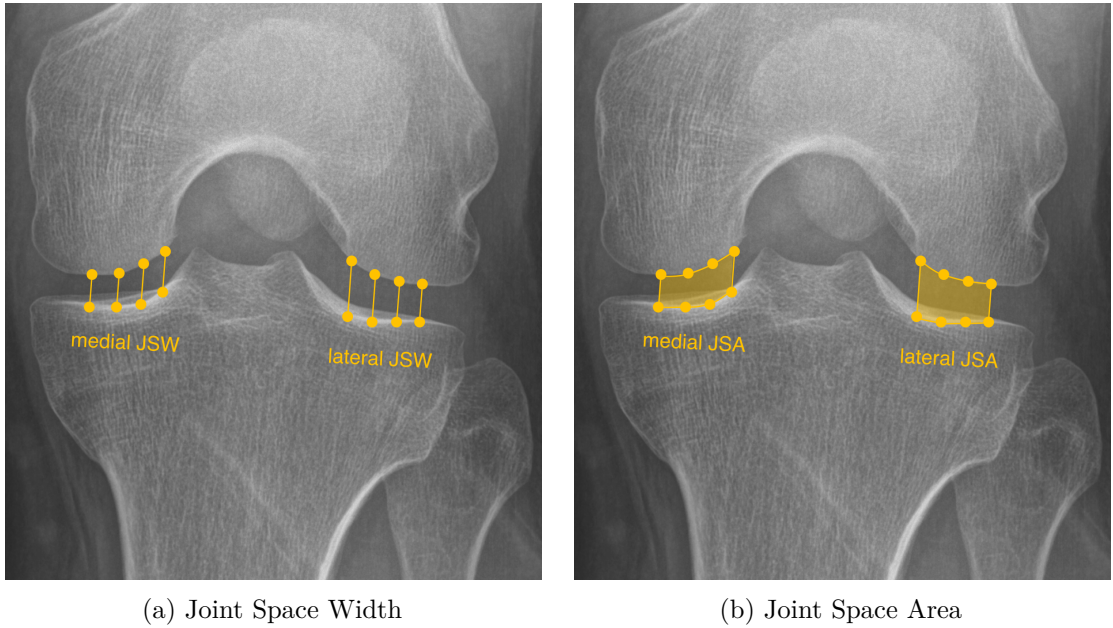


Figure 3.4: Knee joint space assessments

For the calculation of the KL-Subscores that form the actual **KL-Score**, all calculations are done by invoking native library methods. Since the native libraries used are not thread-safe, all computations are done in `synchronized`-blocks and are therefore executed in an atomic fashion. This ensures that multiple sessions cannot influence each other when calculating at the same time.

#### 3.4.1 Retrieving Medial and Lateral Landmarks

Landmarks, also known as knee focus points represent the outermost medial and lateral points of a knee joint (Figure 3.3a). Since all further visualizations and calculations depend on these points, their calculation is crucial to the assessment as a whole. For their placement, our native library utilizes a **DNN** and therefore needs the width, the height, the pixel-spacing, and the image itself as parameters. If landmarks were found, the pointer to an `Array` containing x- and y-values of both landmarks is returned. Else, we calculate default-values that are based on the height and width of the image.

#### 3.4.2 Retrieving Joint Space Focus Points

Joint space focus points (JSFP) are, by **IB Lab**'s internal definition the lower outline of the femur, as well as the upper outline of the tibia and are hence called 'contour-points' (Figure 3.3b). Because the upper outline of the tibia can already be sclerosed, the native library aims at placing it below the so-called sclerosis-line, in order to calculate contour-points more precisely. The invoking of the native method is fairly similar to the one used to detect landmarks before, with the exception, that the returned `Array` can

contain any number of points, depending on the 'contour-resolution'. This resolution gives indication of how many focus points are to be calculated. This value is a balance between the precision of the contouring and performance, which is why we currently calculate 16 points altogether.

### 3.4.3 Calculation of the Joint Space Width

The joint space is the area between the lower femur and the upper tibial bone. The Joint Space Width (JSW) therefore represents the distance between the JSFPs that were calculated (Figure 3.4a). Lower values are interpreted as bad concerning the overall state of a knee joint. Thus, if the distance (or width) decreases over time, less articular cartilage is present to absorb mechanical strain, which generally speaking, leads to pain and the formation of osteophytes. Further, if left untreated, the articular cartilage can degenerate completely, leading to a bone-on-bone contact, which can cause the complete functional impairment of a knee joint. For its calculation, our native method is measuring the distance between the upper and lower JSFPs with their respective pixel-spacing.

### 3.4.4 Calculation of the Joint Space Area

The Joint Space Area (JSA) is the area that is enclosed by the JSW-box in  $mm^2$  (Figure 3.4b). If comparing both medial and lateral compartments, assertions on the deformation of a knee can be made. Therefore, our native library returns the area defined by the given contour-points, enabling later calculation of a joint's deformation.

## 3.5 Kellgren-Lawrence-Score and Report

### 3.5.1 KL-Score

To visualize the calculated KL-Score and for users to make potential alterations, we implemented a stand-alone servlet that communicates with our main servlet and calculates the score itself, as well as its subscores (Figure 3.2). Because our software currently is not an official diagnostic-tool, it may only make suggestions, which a physician can overrule. By clicking on the respective score a check-box is marked, symbolizing that the user overruled our suggestion. Further, by clicking on the 'Download Report'-Button, a comprehensive overview of our analysis is created, which we will detail in the next subsection.

### 3.5.2 Report

Our report gives a summarized overview of the state of a knee joint, thus offering patients a comprehensible look on the state of OA in their knee. In Figure 3.5 we can see an example report for a knee with a KL-Score of 2, meaning that according to the original paper published by Kellgren and Lawrence the degree of OA is considered minimal [KL57]. The subscores for sclerosis, JSW, deformation, and osteophytes can be seen on the left

### 3. IMPLEMENTATION

---

side next to the main **KL-Score**. The boxes containing the values of each subscore are only colored, if the user confirmed them by clicking on the score in our web-interface. Below the actual values of the measured distances between femur and tibia are shown in mm. Further, the **JSA** is displayed numerically and on the right side a graph showing the relation between medial and lateral compartments can be found.

## Softwareunterstützte Arthrosebewertung nach Kellgren-Lawrence\*

PATIENTEN DATEN		UNTERSUCHUNGS DATEN	
Patienten Name	Lukas Maximilian Masopust	Untersuchungs ID	
Patienten ID	01427382	Aufnahmedatum	20140606
Geburtsdatum	25051995	Körperseite	L
Geschlecht	M	Institut	Knie Studie
Größe [cm]	180	Analyst	
Gewicht [kg]	66	Gerät	Fluorospot Compact FD

KNIE ARTHROSE BEWERTUNG		KELLGREN & LAWRENCE
<b>SKLEROSIERUNG</b> <div>0</div> 0 = Keine 1 = Leicht 2 = Leicht mit Zystenbildung 3 = Stark mit Zystenbildung	<b>Gelenksspalt</b> <div>2</div> 0 = Nicht/fraglich verschmälert 1 = Deutlich verschmälert 2 = Stark verschmälert 3 = Aufgehoben	<b>SCHWEREGRAD</b> <div>2</div> Geringe Gelenkspaltverschmälerung und beginnende Osteophytenbildung, angedeutete Unregelmäßigkeiten der Gelenkfläche
<b>DEFORMATION</b> <div>1</div> 0 = Keine 1 = Leicht 2 = Deutlich	<b>OSTEOPHYTEN</b> <div>1</div> 0 = Keine/fraglich 1 = Eindeutig 2 = Groß	
GESAMTWERT ALLER PARAMETER:		

GEMESSENER Gelenksspalt		
	MEDIAL	LATERAL
H1 [mm]	8.9	5.5
H2 [mm]	7.7	4.3
H3 [mm]	8.4	4.3
H4 [mm]	8.7	3.0

GEMESSENE Gelenkspaltfläche		
	MEDIAL	LATERAL
Fläche [mm²]	141.6	73
Schiefelage [%]**	66%	34%
rel. Fläche [%]**	32%	17%

Later/Medial Verhältnis in Proze

MEDIAL LATERAL

ANMERKUNGEN

Figure 3.5: Example report



# Evaluation

In this chapter we will discuss the performance and accuracy of our `machine learning (ML)`-based native libraries and those that used `CV`-algorithms. Further, we will provide a general comparison of all of our current solutions.

## 4.1 Performance

In their paper Grimmer et al. `GRS+13` show that `JNA` offers severely less performant native method calling when compared to native mapping via `JNI`. Since Java's `JNI` utilizes direct mapping as seen in Python's native interface, the conclusion can be drawn that native calls via `JNA` are less performant than those from within Python, hence slowing down all calculations in each native method.

## 4.2 Accuracy

As we will see, utilizing `DNNs` for detecting the knee joint and finding both the medial and the lateral landmark yields significantly better results when compared to the traditional method of using `CV`-algorithms.

### 4.2.1 Setup

Since the hardware is irrelevant to the precision of our measurements, we will discuss the software used to gather the data for our statistical evaluation. For testing the `ML`-approach and the legacy `CV`-based method, we used the respective latest versions of both native libraries.

A set of 29 different `DICOM` images was used as our test sample. Those images were gathered through the MOST-study, a research study conducted by the US-American

National Institute of Health [SNG<sup>+</sup>13].

Each image was sent to the server via Postman, a free tool for sending and receiving network requests [POS]. We initialized a new session, thereby sending the image as a DICOM-stream, and obtained the calculated SVG via the getSVG-command.

Thereafter we extracted each landmark's x- and y-coordinates for every analyzed knee and measured the distance between the resulting points from our native library and compared them to where they were placed in our ground truth.

We generally found that the ML-model offered landmarks that were significantly closer to our ground truth than the CV-algorithms, as illustrated in the example below (Figure 4.1).

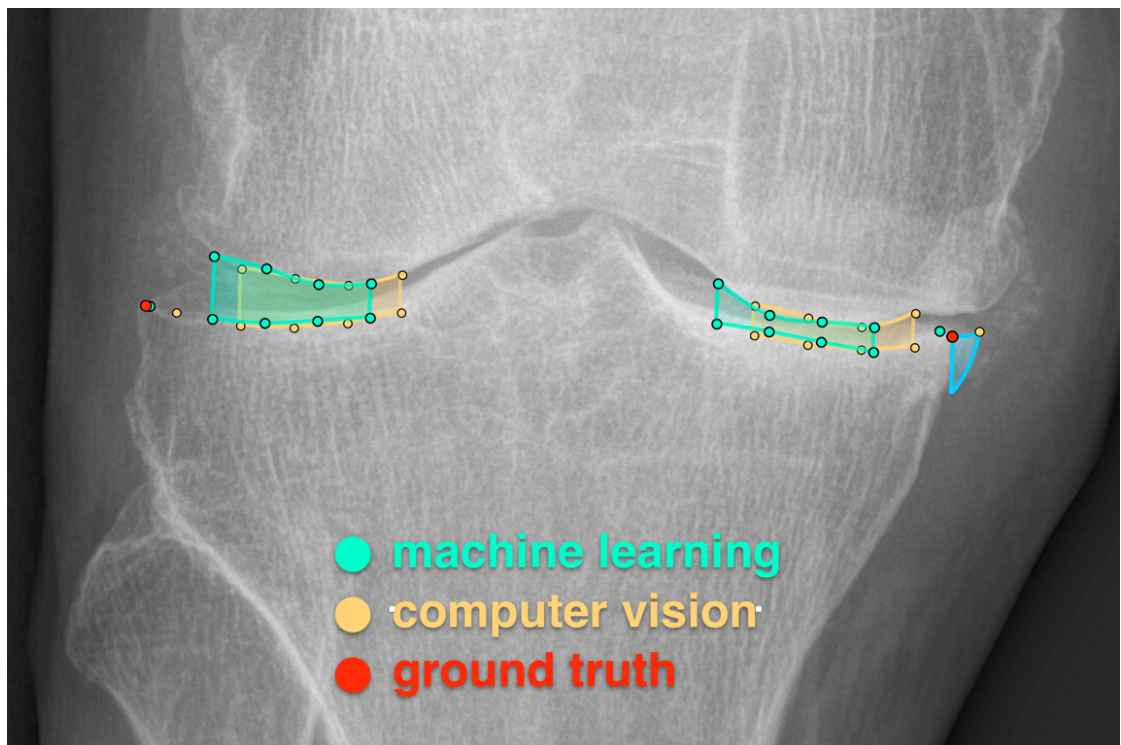


Figure 4.1: Graphical comparison of ML-, CV-data and the ground truth

We can discern from the Figure 4.1 that the placing of landmarks has severe consequences on the respective contour-points and therefore alters both the JSW and the JSA.

The difference in precision is due to the fact that the ML-model detects structural alterations such as osteophytes (marked with the blue shape) and therefore places the landmarks accordingly.



To give a visual representation of the deviation of both **ML**- and **CV**-algorithms from our ground truth, in Figure 4.2 we provide a histogram showing where most deviation-values of all 29 sample images can be found. Since our sample had resolutions ranging from 1564 x 1248 to 1806 x 3527, we used the absolute distance in millimeters for this measurement. A greater deviation indicates that the landmarks were placed with a bigger offset to our ground truth assessment.

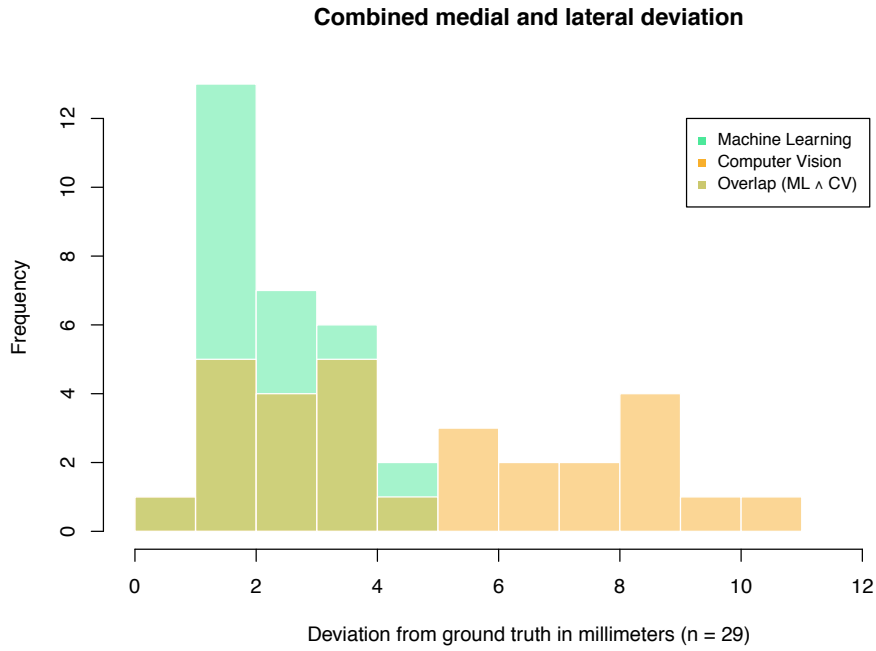


Figure 4.2: Histogram of the deviation from ground truth of both **ML**- and **CV**-data

As we can discern from this evaluation, coordinates gathered from **ML**-algorithms deviate around 1 *mm* from ground truth, whereas those calculated by **CV**-algorithms spread rather equally and can deviate up to 11 *mm*. This illustrates, as discussed in Chapter 5, that **ML-models** generate data that is far more precise than legacy solutions.

To test the significance (we assume 5 % as significant), we ran a T-test. The results of the test can be seen in Table 4.1 below. The **ML**-analysis is significantly more accurate than its **CV**-counterpart.

	Mean Deviation from Ground Truth in <i>mm</i>	p-value
<b>ML</b> -Data	2.34	0.0001954
<b>CV</b> -Data	4.74	

Table 4.1: Mean deviation and p-value from **ML**-, and **CV**-data

### 4.3 General Comparison of our Solutions

As later explained in Chapter 6 under Future Expansions, one of our goals, in order to gain complete platform-independence, is running IAS inside a Docker-container [DOC]. Thus, we will compare our IAS-solution to our legacy stand-alone desktop client, the IB Lab Analyzer and to our Docker-container-solution in Table 4.2.

	IB Lab Analyzer	IAS (Application Server)	IAS (Docker)
OS (Client)	Windows 7, 10	any	any
OS (Server)	-	UNIX-based Systems	any
Deployment	local installation	local viewer integration	web-access
Maintenance	local updates	exchanging .war-file	pushing Docker-image
Decompression	yes	yes	yes
Printable Reports	yes	yes	yes
Overlay in GUI	yes	yes	no
Points modifiable	yes	yes	no

Table 4.2: Comparison of all current solutions



# Discussion

We could see that the modular design of our solution is beneficial for later maintainability. While calculating scores in native libraries was slower due interfacing with them using [JNA](#), the ability to exchange them later, if upgrades are to be done, makes up for the loss in performance.

Further, as discussed below, performance increased in a significant way, when compared to our legacy [CV](#)-approach. Since faster assessments go along with faster diagnosis, a doctor can focus on more patients in less time, thereby maximizing contact with each individual patient while at the same time minimizing costs.

It also seems clear, that an online solution is favorable for deploying our software, since the [IAS](#) can easily be distributed to any medical facility that utilizes a viewer. This further increases maintainability, because potential updates can be deployed easily over the web. Also, as we will see in the conclusion of this thesis, the modular design further increases later expandability in order to, for example, support different file formats, input-streams, or calculation modules.

## 5.1 Performance

As we could infer from our evaluation in Chapter [4](#), our current version of [IAS](#) was slower generating data from native libraries. Still, as can be seen in the work of my colleague David Bauer [\[Bau18\]](#), whose aim was to generate the overlay for visualizing our data in a user interface, response times and therefore frame-rates were greatly improved. As mentioned in the introduction (Chapter [1](#)), the early showcase of our project utilized files for interfacing Java, Python, and native code, which severely degraded performance.

### 5.2 Accuracy

On the other hand, we saw that the precision of landmark-detection significantly increased, when utilizing [DNNs](#). Therefore, the [JSW](#), [JSA](#) and hence the respective subscores are calculated based on more accurate data, leading to a generally more reliable calculation of the [KL-Score](#).

# Conclusion

## 6.1 Summary

In this thesis we have seen the relevance of `OA` and its assessment via the `KL-Score`. We have discussed how our `IAS` contributes to making `OA`-analysis more accessible and easier deployable. We investigated the benefits and challenges of using native code for our calculations and described our way of interfacing with those methods. By showing our implementation, we saw how we communicate with `SOs`, our way of interpreting pixel-data and the way we calculate the `KL-Score`. We evaluated our method and could see that our approach, while being slower at calculating data, yielded much improved accuracy when compared to traditional `CV`-algorithms.

To conclude this thesis we will give an outlook of what the future holds for our project. We will discuss both future technologies, as well as potential deployment infrastructures.

## 6.2 Future Expansions

We have seen `IAS` in its current working state. Since the reception of our software was overwhelmingly positive, there are already a lot of ideas for future editions.

### 6.2.1 Version 2

The future goal of `IB Lab` is not only to offer knee-analyzation capabilities. There is already research being done in different areas utilizing `DNNs`. Hence, the next big version of `IAS` currently exists in the form of design documents. To incorporate different models for all kinds of image analysis, our aim is to engineer software that is more modular and dynamic. For example, our application should return masks and calculations for images, regardless of the calculations being done in native libraries. A whole range of functionality could be added in the form of native code, without alterations to the server

itself.

Currently, there are new [ML](#)-based native libraries in testing, that are able to assess sclerosis, deformation, and osteophytes, which will be added to this revamped version of [IAS](#).

### 6.2.2 Docker

The current form of [IAS](#) is designed to work under Ubuntu with version 4 of the Linux Kernel, because our native libraries take advantage of some system-calls that are specific to the kernel-version. For us to gain complete platform-independence, we are currently working on running [IAS](#) inside a Docker environment. At the point of writing, a working prototype of our software is already running inside an EnvoyAI machine.

### 6.2.3 EnvoyAI

EnvoyAI is a platform that offers AI capabilities for radiologists [ENV](#). They provide physicians with a gateway to access all kinds of different machines for analyzing radiographic imagery, thereby enabling a completely different infrastructure for radiodiagnosics. Through this approach, radiologists will have software and algorithms for all sorts of analysis available at their fingertips, without depending on local installations. While EnvoyAI clearly seems like a platform that could revolutionize medical analysis, their infrastructure is not only beneficial to the end user: companies like ours are offered a very simple way of distributing their code by uploading the Docker-container to the web. Thus, updates of our software no longer need to be installed manually for a specific workstation, but are pushed to all customers with a few clicks.

### 6.2.4 Ubuntu 15.04 Compatibility

The upgrade of the Linux Kernel from Version 3 to 4 brought changes to the libraries that our [SOS](#) utilize. Because there are servers in clinics that are still running Ubuntu 15.04, a re-compiled version of our native libraries is currently in development.

# List of Figures

1.1	Stand-alone IB Lab Analyzer [IBL]	2
2.1	DICOM file diagram [DCM]	6
2.2	Role of JNI [Lia11]	7
3.1	Analyzed joint with mask-overlay	10
3.2	KL-Score	10
3.3	Focus points of a knee joint	13
3.4	Knee joint space assessments	14
3.5	Example report	17
4.1	Graphical comparison of [ML], CV-data and the ground truth	20
4.2	Histogram of the deviation from ground truth of both [ML] and CV-data	21





# List of Tables

4.1	Mean deviation and p-value from ML-, and CV-data . . . . .	21
4.2	Comparison of all current solutions . . . . .	22



# List of Algorithms

3.1 Monochrome conversion . . . . .	12
-------------------------------------	----



# Listings

3.1	Example interfacing of a native library	10
3.2	Example usage of a JNA-Pointer	11
3.3	Decompression in dcm4chee 3	12



# Acronyms

**AI** artificial intelligence. [xiii](#)

**CV** computer vision. [xiii](#), [19–21](#), [23](#), [25](#), [27](#), [29](#)

**DICOM** Digital Imaging and Communications in Medicine. [6](#), [11](#), [12](#), [19](#), [20](#)

**DNN** deep neuronal network. [2](#), [3](#), [14](#), [19](#), [24](#), [25](#)

**IAS** Image Analysis Server. [3](#), [5–9](#), [22](#), [23](#), [25](#), [26](#)

**IB Lab** ImageBiopsy Lab. [xiii](#), [5](#), [7](#), [8](#), [14](#), [25](#)

**JNA** Java Native Access. [7](#), [8](#), [10](#), [11](#), [19](#), [23](#)

**JNI** Java Native Interface. [7](#), [19](#)

**JSA** Joint Space Area. [9](#), [13](#), [16](#), [20](#), [24](#)

**JSW** Joint Space Width. [1](#), [9](#), [13](#), [15](#), [20](#), [24](#)

**JVM** Java Virtual Machine. [7](#)

**KL-Score** Kellgren-Lawrence-Score. [1](#), [3](#), [9](#), [13–16](#), [24](#), [25](#)

**ML** machine learning. [19–21](#), [26](#), [27](#), [29](#)

**ML-model** machine learning model. [5](#), [8](#), [20](#), [21](#)

**OA** osteoarthritis. [xiii](#), [1–3](#), [8](#), [15](#), [25](#)

**SO** shared object. [5](#), [10](#), [25](#), [26](#)

**TSUID** transfer-syntax-uid. [12](#)





# Bibliography

- [Bau18] David Bauer. Automated Visual Assessment of Osteoarthritis. Bachelor thesis submitted to the Institute of Visual Computing and Human-Centered Technology, TU Wien. 2018.
- [DCM] DICOM File Format, [http://dicom.nema.org/DICOM/2013/output/chtml/part10/chapter\\_7.html](http://dicom.nema.org/DICOM/2013/output/chtml/part10/chapter_7.html), accessed on 2018-06-02.
- [DOC] Docker, <https://www.docker.com>, accessed on 2018-08-29.
- [ENV] EnvoyAI, <https://www.envoyai.com>, accessed on 2018-05-25.
- [GG] Steven R. Goldring and Mary B. Goldring. Clinical aspects, pathology and pathophysiology of osteoarthritis. *Journal of musculoskeletal & neuronal interactions*, 6(4):376–8.
- [GRS<sup>+</sup>13] Matthias Grimmer, Manuel Rigger, Lukas Stadler, Roland Schatz, and Hanspeter Mössenböck. An efficient native function interface for Java. In *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform Virtual Machines, Languages, and Tools - PPPJ '13*, page 35, New York, New York, USA, 2013. ACM Press.
- [HIK<sup>+</sup>17] Shinnosuke Hada, Muneaki Ishijima, Haruka Kaneko, Mayuko Kinoshita, Lizu Liu, Ryo Sadatsuki, Ippei Futami, Anwajan Yusup, Tomohiro Takamura, Hitoshi Arita, Jun Shiozawa, Takako Aoki, Yuji Takazawa, Hiroshi Ikeda, Shigeki Aoki, Hisashi Kurosawa, Yasunori Okada, and Kazuo Kaneko. Association of medial meniscal extrusion with medial tibial osteophyte distance detected by T2 mapping MRI in patients with early-stage knee osteoarthritis. *Arthritis research & therapy*, 19(1):201, Sep 2017.
- [Hun11] Robert Hundt. Loop Recognition in C++/Java/Go/Scala. 2011.
- [IBL] Image Biopsy Lab, <https://imagebiopsylab.com/products/jsx>, accessed on 2018-07-08.
- [ISO] ISO 13485, <https://www.iso.org/iso-13485-medical-devices.html>, accessed on 2018-05-22.

- [KL57] Jonas H. Kellgren and John S. Lawrence. Radiological assessment of osteoarthritis. *Annals of the rheumatic diseases*, 16(4):494–502, Dec 1957.
- [Lia11] Sheng Liang. *The Java Native Interface*, volume 56. 2011.
- [LKN] Landeskliniken-Holding, <https://www.lknoe.at>, accessed on 2018-07-08.
- [LYG<sup>+</sup>13] Guangyi Li, Jimin Yin, Junjie Gao, Tak S Cheng, Nathan J Pavlos, Changqing Zhang, and Ming H Zheng. Subchondral bone in osteoarthritis: insight into risk factors and microstructural changes. *Arthritis research & therapy*, 15(6):223, 2013.
- [Neo12] Tuhina Neogi. Clinical significance of bone changes in osteoarthritis. *Therapeutic advances in musculoskeletal disease*, 4(4):259–67, Aug 2012.
- [PHO] Image Pixel Module, [http://dicom.nema.org/medical/dicom/2014c/output/chtml/part03/sect\\_C.7.6.3.html#sect\\_C.7.6.3.1.2](http://dicom.nema.org/medical/dicom/2014c/output/chtml/part03/sect_C.7.6.3.html#sect_C.7.6.3.1.2), accessed on 2018-05-28.
- [POS] Postman, <https://www.getpostman.com>, accessed on 2018-05-26.
- [SNG<sup>+</sup>13] Neil A. Segal, Michael C. Nevitt, K. Douglas Gross, Keith D. Gross, Jean Hietpas, Natalie A. Glass, Cora E. Lewis, and James C. Torner. The Multi-center Osteoarthritis Study: opportunities for rehabilitation research. *PM & R : the journal of injury, function, and rehabilitation*, 5(8):647–54, Aug 2013.
- [TSU] Transfer Syntaxes, <https://www.dicomlibrary.com/dicom/transfer-syntax/>, accessed on 2018-05-27.
- [VIS] Visuapps, <http://www.visuapps.com>, accessed on 2018-07-08.
- [WKW14] Marius C Wick, Martin Kastlunger, and Rüdiger J Weiss. Clinical imaging assessments of knee osteoarthritis in the elderly: a mini-review. *Gerontology*, 60(5):386–94, 2014.
- [WP03] Anthony D Woolf and Bruce Pfleger. Burden of major musculoskeletal conditions. *Bulletin of the World Health Organization*, 81(9):646–56, 2003.