AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

KATEDRA ODDZIAŁYWAŃ I DETEKCJI CZĄSTEK

Praca dyplomowa

Application to analyze Winston-Lutz tests using artificial intelligence

Aplikacja do analizy testów Winstona-Lutza z wykorzystaniem sztucznej inteligencji

Autor: Konrad Jakub Walas
Kierunek studiów: Informatyka Stosowana
Opiekun pracy: dr inż. Bartłomiej Rachwał

Kraków, 2024

Contents

Chapter 1

Introduction

Radiation therapy is an important field of medical science. Due to its highly specialized nature, there are very few free, ready-made software tools for working with this topic. In addition, the materials that can be found online are usually specialized articles, often requiring additional fees to gain access. For this reason, any applications connected to this area, especially open-source ones, are a significant help for students and scientists associated with this field.

One of the common devices used in radiation therapy is the linear accelerator. Like any sophisticated medical equipment, these devices require regular calibration to maintain their precision and reliability. Calibration is essential, not only to ensure the accuracy of the radiation dose delivered to patients but also to meet rigorous safety standards and regulatory compliance.

This thesis addresses this problem by proposing open software tools to assist in the maintenance of linear accelerators by developing a web application with functionality related to the calibration of these devices.

The collaboration with Damian Kubat, the head of the Department of Medical Physics at the National Institute of Oncology in Kraków, has led to the identification of the most significant challenges encountered during routine calibration of linacs. Consequently, two modules have been developed with the potential to address these challenges.

1.1 Purpose

This work aims to extend an existing web application with two modules:

- Winston-Lutz test
- MLC leaves alignment analysis for compliance with the designed plan

Both modules provide tooling for the calibration of linear accelerators with multileaf collimators. Ready solutions for carrying out the Winston-Lutz test are already available on the web, for example, the Pylinac library for Python language [22], and they were used to develop the corresponding module. However, as for the analysis of the collimator leaves alignment and compliance with the plan, no ready-made solutions were found during the research phase, therefore the presented solution was designed from scratch as part of the thesis.

1.2 Structure of the thesis

The thesis consists of five main parts, that span the next chapters:

Part I - Background : Chapters 2, 3, 4

Concepts and theory that are related to the scope of the thesis.

Part II - Module Definition : Chapters 5, 6

Logical description of created modules and used algorithms.

Part III - Implementation : Chapters 7, 8

Technical description of the application and used tools.

Part IV - Analysis : Chapter 9

Performed evaluations and their results.

Part V - Summary : Chapter 10

Conclusions and fields of applications.

Chapter 2

Glossary

API - Application Programming Interface

Backend - Server side of the application in a client-server architecture

BB - Ball bearing

Phantom used in W-L test.

CAX - Center of the radiation field

As stated in Pylinac documentation (v3.25) this term is a misnomer, but it is used due to backwards-compatibility reasons. To ensure consistency, this term is also used in this thesis. [24]

CSS - Cascading Style Sheets

The styling language used to describe the appearance of HTML elements.

CT - Computed Tomography

CV - Computer Vision

DICOM - Digital Imaging and Communications in Medicine

Technical standards and file formats commonly used to store medical images.

DRF - Django Rest Framework

EPID - Electronic Portal Imaging Device

Frontend - Client side of the application in a client-server architecture

HTML - HyperText Markup Language

The markup language used to define the structure of a web page.

HTTP - Hypertext Transfer Protocol

JS - JavaScript

The scripting language used to define custom logic on a web page.

LA analysis - Leaves alignment analysis

Linac - Linear particle accelerator

MLC - Multileaf Collimator

Part of Linac that controls beam geometry.

ORM - Object Relational Mapping

Layer that connects object-oriented programming languages with databases.

QC - Quality control

Process of verifying the characteristics of a given product and relating the results to the specified requirements.

RT - Radiation therapy

SE - Structuring element

A shape used in morphological operations.

SID - Source-Image Distance

The distance between the radiation beam source and the image receptor.

UI - User Interface

Part of the application exposed to its users, that provides a medium of communication between application and users.

W-L test - Winston Lutz test

Chapter 3

Theoretical background

3.1 Radiation therapy

Radiation therapy is a treatment that utilizes beams of ionising radiation to control the growth or kill cancer cells. This type of treatment can be effective in cases where the cancerous cells are concentrated in a particular area of the body and have not spread to other regions. This treatment requires specialised, high-precision equipment - *linear particle accelerators*. [12]

3.1.1 Linear particle accelerator

A linear particle accelerator, often referred to as a *linac*, is a device that accelerates charged particles (electrons) to velocities near the speed of light by using oscillating electric fields to push the electrons through a series of accelerating cavities. Subsequently, these electrons collide with a heavy metal target, resulting in the generation of high-energy X-rays. These are then shaped as required upon exiting the machine. [15]

Linac rotation and isocenter

Linear accelerators used in radiotherapy rotate around a single point called the *isocenter*. At the isocenter, the axes of rotation of the linear accelerator intersect, thereby defining the point of maximum radiation concentration. Patients are placed on a movable table to align the treatment area and the isocenter. The three rotating components of the accelerator are:

1. **Gantry**, designated as G on Figure 3.1

   A gantry is a device that holds a radiation source, radiation detectors, collimator and all the apparatus needed to send out a beam of particles.

2. **Coach / Table**, designated as T on Figure 3.1

   The coach is utilised to facilitate the positioning of the patient within the device.

3. **Collimator**, designated as C on Figure 3.1

   A collimator is a device which narrows a beam of particles. The narrowing can mean either limiting the spacial cross-section of a beam or aligning the beam's direction with a specific direction. [12]

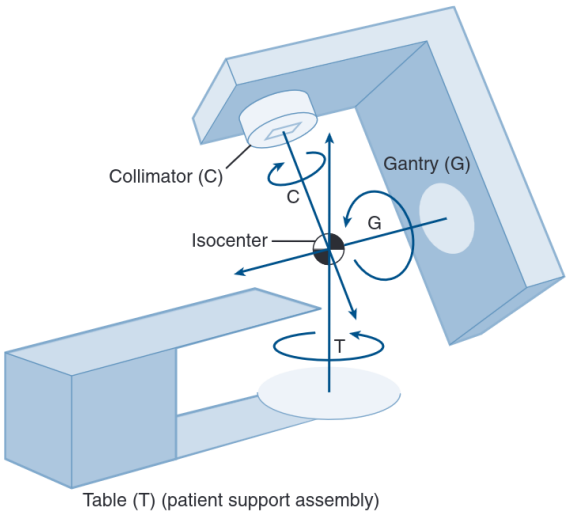


Figure 3.1: Layout of gantry-based linac [12]

Multileaf collimator (MLC)

A *multileaf collimator* (MLC) is a collimator composed of individual leaves of a high atomic number material, typically tungsten. These leaves can move independently in and out of the path of a beam to shape and vary its intensity, to enable generating beams of particles with high precision, covering the exact geometry of the targeted body area. [12]

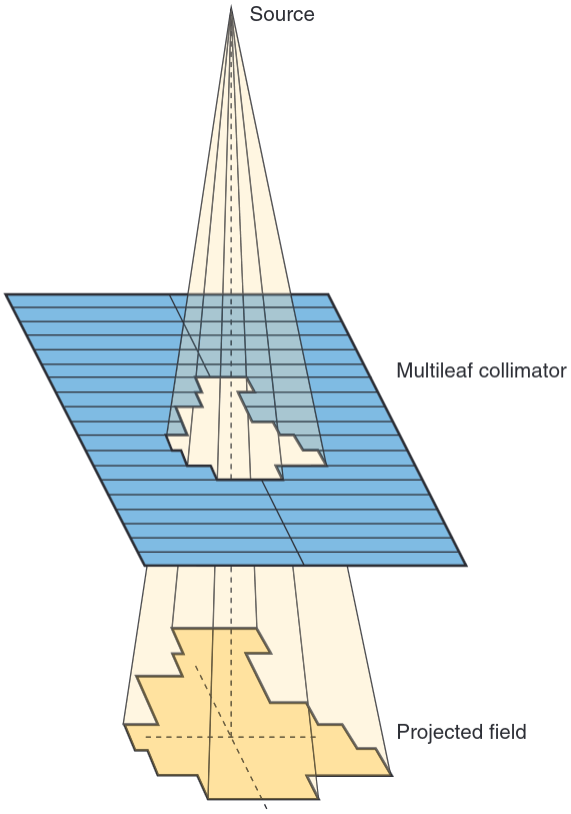


Figure 3.2: Narrowing beam with multileaf collimator [12]

3.1.2 DICOM

Digital Imaging and Communications in Medicine (DICOM) is a technical standard for the digital storage, management and transmission of medical images and related information. It is important to notice that this standard specifies not only file format but also information like the set of protocols used for communication and syntax of commands and associated information that can be exchanged using these protocols. [7]

DICOM file format

A DICOM data consists of various attributes and is grouped into data sets. This ensures that all relevant data about the image (like patient ID) cannot be separated and accidentally lost. Besides attributes like patient name, age etc. file contains one special attribute - image pixel data. Pixel data can be converted to other formats like JPEG or PNG and the easily displayed on any modern device. [6]

3.2 Artificial Intelligence

Artificial intelligence (AI) is a very broad term that generally describes any form of intelligent behaviour displayed by a computer system. AI is a field of research in computer science that explores approaches that enable machines to observe reality, learn and take intelligent actions to achieve defined goals.

There are numerous subfields of AI, concentrated around particular goals, like reasoning, natural language processing or support for robotics. Current research focuses on the achievement of artificial general intelligence, for example, the ability to perform any task that a human can perform, at least at an equal level, but such a state has not yet been achieved. Therefore, any current tool or model focuses on solving a specific task.

Simulating intelligence boils down to separating problems into several subproblems that can be addressed individually using different tools and models. The following system characteristics are not exhaustive examples of possible research fields:

- Reasoning and problem-solving
- Knowledge representation
- Planning and decision-making
- Learning
- Natural language processing
- Perception
- Social intelligence

This thesis will focus on one of them - perception. It describes the ability to process input from various sensors (like sonar, wireless signals, cameras, microphones, radar or lidar) to deduce aspects of the world. This field includes topics like object recognition and tracking, facial recognition, speech recognition and image classification. The branch that deals with research in this area is called *computer vision*. [31]

3.2.1 Computer Vision

Computer vision tasks involve processing, analyzing and extracting data from the real world in the form of visual input to produce symbolic or numerical information, often in the form of decisions. [18]

For a CV algorithm to analyze an image, it must first be processed into a form that is suitable for the algorithm. This frequently involves the extraction of specific features from the given image, such as edges, whose analysis will lead to meaningful conclusions.

3.3 Tools of image processing and analysis

Digital image is image composed of picture elements, called *pixels*. In this case, each pixel represents a two-dimensional function with spatial coordinates given by x and y as input and a finite, discrete numerical representation of its intensity or grey level as output. [13]

Color representation and greyscale

A *greyscale* image consists of pixels that have only one value representing the amount of light, and therefore only contains intensity information. [17] On the other hand in *coloured* images one pixel can store more values called *channels*, each representing one of the colour components that combined produce the desired colour. A grayscale image has just one channel.

In the code, greyscale images can therefore be represented as a two-dimensional array of $n \times m$ numbers and colour images as a three-dimensional array of $n \times m \times c$ numbers, where n and m specify the dimension of the image and c represents the number of channels.

RGB

The *RGB colour model* is an additive model which enables the representation of a wide range of colours by adding together *red*, *green* and *blue* - primary colours of light. The name of the model comes from the first letters of its component colours. [14]

3.3.1 Thresholding (binarisation)

Image *thresholding* is a method of segmenting images. It involves modifying the image so that each pixel has one of two values often interpreted as the presence or absence of an object. For this reason, thresholding is also called *binarisation*. Simple binarisation methods consist of replacing each pixel in the image with a black pixel if the pixel intensity is less than a fixed threshold T, or with a white pixel if the pixel intensity is greater than T.[33]

Binarisation can only be performed on a monochannel image. Multichannel images must first be converted to monochannel ones. Another approach would be to apply binarisation to each channel separately and then combine the image. One of the methods is to leave a white pixel if its value in each channel is above the threshold.

Binarisation helps to significantly reduce the dimensionality of the problem and is the basis for many further image analysis methods.

Adaptive thresholding

Simple methods use global value as a threshold. However, this is not always advantageous; for example, if an image contains uneven lighting in different areas, it can cause dimmed areas to turn all black, leaving the desired features undetected.

The solution is *adaptive thresholding*, which determines the threshold value for a pixel based only on the region around it. As a result different thresholds for different regions of the image are used, giving better results for images with varying illumination.

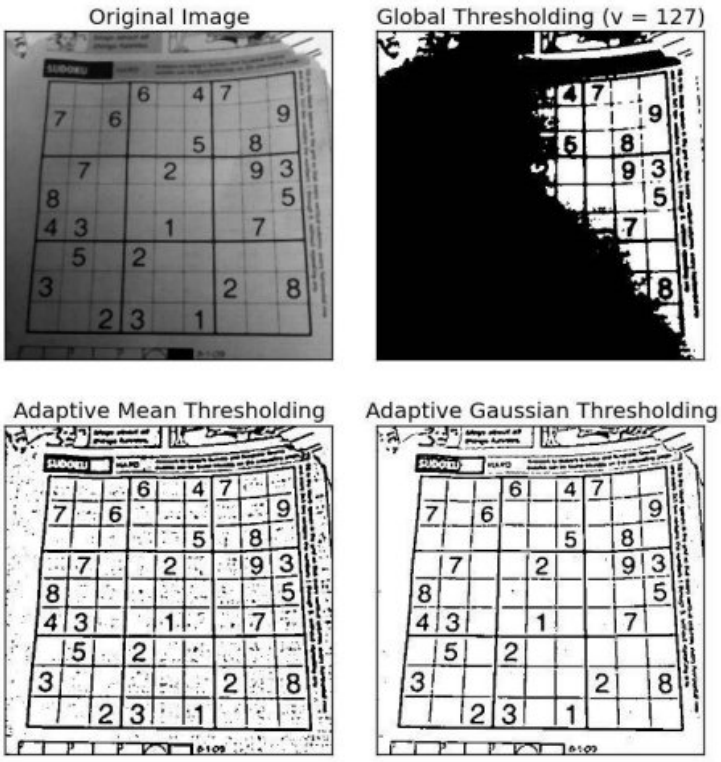


Figure 3.3: Comparison of global and adaptive thresholding for image with varying illumination [48]

3.3.2 Inversion

Binary image *inversion* is the process of transforming an image in such a way that the pixels that were originally black are transformed into white, and vice versa. For RGB images inverted image is a *negative* of the original image. [20] It is achieved by subtracting the value of the pixel from its maximum possible value for each channel:

$$I'(x, y, c) = \mathbf{M} - I(x, y, c) \tag{3.1}$$

where

- I' is the inverted image,
- $\mathbf{M}$ is max value that pixel can have
- $I(x, y, c)$ is value of original image for pixel's channel c and coordinates x and y

Figure 3.4: Original and inverted image [45]

3.3.3 Filtering

Filtering is a process of transforming the image into a slightly modified version by sequentially applying a specifically designed image, called *the kernel*, to all pixels of the source image. Depending on the choice of kernel, a different effect can be obtained. Filtering can be performed by convolution between the kernel and the image. [13]

Example filtering operations

- Edge detection
- Sharpening
- Blurring
- Identity
- Fourier lowpass/highpass (frequency filtering)

Convolution

Convolution is fundamental *neighbourhood image transformation*, meaning its output value at a given pixel is a function of the values of the pixels present in the neighbouring region centred on the considered pixel.

Convolution is defined as an operation on two images f and g, whose origin is in the centre of its pixel frame $\mathcal{D}_g$ (middle one pixel). The output of the convolution at a given pixel x of f is defined by the weighted sum of the pixels of f falling within $\mathcal{D}_g$ when the origin of $\mathcal{D}_g$ coincides with $\mathbf{x}$, the weights being defined by the values of g [33]:

$$[f * g](\mathbf{x}) = \sum_{\mathbf{b} \in \mathcal{D}_g} [f(\mathbf{x} - \mathbf{b}) g(\mathbf{b})] \tag{3.2}$$

Intuitively, convolution is the process of adding each element of the image to its local neighbours, weighted by the kernel.

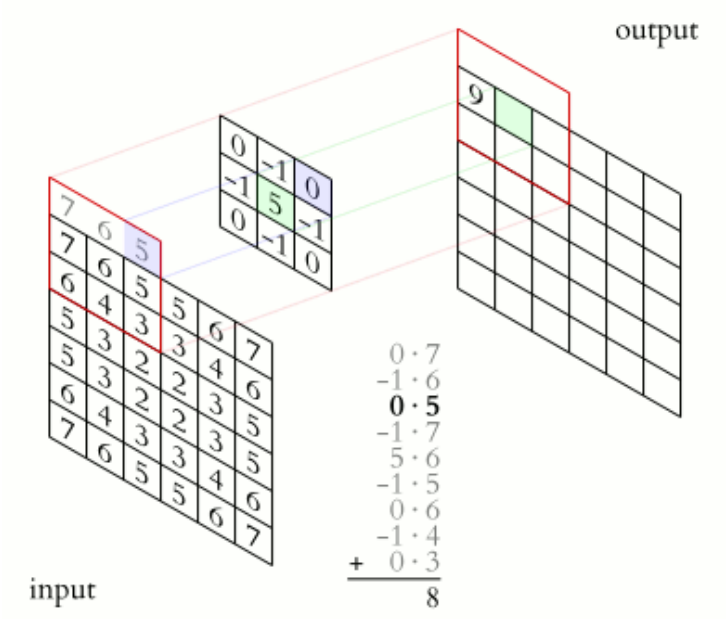


Figure 3.5: Visualisation of 2D discrete convolution [50]

3.3.4 Edge detection

Edge detection is a set of mathematical techniques that aim to identify *edges*, defined as curves in a digital image where the brightness of the image changes sharply, or where there are discontinuities. Edge detection is an essential tool in image processing and computer vision, especially in area and feature extraction. The algorithms differ in their choice of kernel, called an *operator*. [38]

Sobel operator

The operator uses two 3×3 kernels which are applied to the original image to calculate approximations of the derivatives – one for vertical changes, and one for horizontal. The resulting images are defined as follows: [32]

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \tag{3.3}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \tag{3.4}$$

where

- $\mathbf{A}$ is the source image;
- $\mathbf{G}_x$ image, where each point contains the horizontal derivative approximation
- $\mathbf{G}_y$ image, where each point contains the vertical derivative approximation

The x-coordinate increases in the 'right' direction and the y-coordinate increases in the 'down' direction. Resulting gradient approximations can be combined at any point in the image to produce a gradient magnitude:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \tag{3.5}$$

Example images

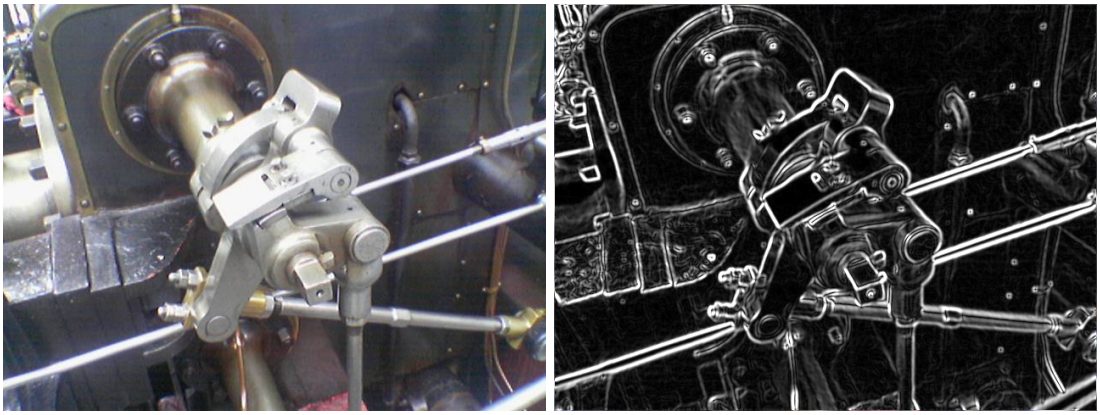


Figure 3.6: Original image [53] and resulting image with combined gradient approximations computed with Sobel operator [54]

3.3.5 Blob detection

Blob is a group of pixels in an image that are connected and share common properties. *Blob detection* is the process of detecting regions in an image that differ in properties, such as shape, colour, or brightness from its surroundings. The most common blob detection methods use convolution. The procedure typically involves three steps:

1. thresholding, which divides the image into segments,
2. assembling linked pixels into clusters,
3. examining clusters to find details such as their dimensions, centres and shapes. [2]

3.3.6 Morphological operations

Morphology can be defined as a theory for the analysis of spatial structures. It aims at analysing the shape and form of objects. It can be used to solve such tasks as:

- Noise filtering
- Correction of uneven illumination
- Edge enhancement
- Image segmentation

There are two basic operations: *erosion* and *dilation*, and their combinations *opening* and *closing*.

Structuring Element (SE)

The basic idea of morphology is to use a pre-defined shape to probe an image and then draw conclusions on how well or poorly this shape fits the other shapes in the image. The "probe" is called the *structuring element* and it is a separate image (kernel) of smaller size and well-defined shape. In practice, in image analysis, the most common kernel will be the square matrix $n \times n$.

Erosion

The erosion of a set X by a structuring element B is denoted by $\varepsilon_B(X)$ and is defined as the locus of points $\mathbf{x}$ such that B is included in X when its origin is placed at $\mathbf{x}$:

$$\varepsilon_B(X) = \{\mathbf{x} | B_{\mathbf{x}} \subseteq X\}. \tag{3.6}$$

Intuitively, it can be understood as reducing objects' edge by the kernel going around it.

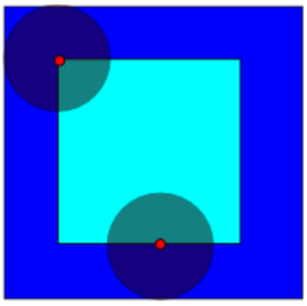


Figure 3.7: Erosion of dark-blue square by a circle structuring element [51]

Dilation

The dilation of a set X by a structuring element B is denoted by $\delta_B(X)$ and is defined as the locus of points $\mathbf{x}$ such that B hits X when its origin coincides with $\mathbf{x}$:

$$\delta_B(X) = \{\mathbf{x} | B_\mathbf{x} \cap X \neq \emptyset\}. \tag{3.7}$$

Intuitively, it can be understood as the thickening of an object by an edge caused by the kernel going around it.

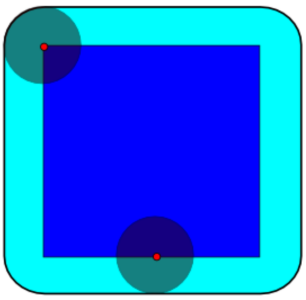


Figure 3.8: Dilation of dark-blue square by a circle structuring element [52]

Opening

The opening γ of an image f by a structuring element B is denoted by $\gamma_B(f)$ and is defined as the erosion of f by B followed by the dilation with the reflected SE $\check{B}$:

$$\gamma_B(f) = \delta_{\check{B}}[\varepsilon_B(f)], \tag{3.8}$$

which can also be written as:

$$\gamma_B(X) = \bigcup_\mathbf{x} \{B_\mathbf{x} | B_\mathbf{x} \subseteq X\} \tag{3.9}$$

Closing

The closing of an image f by a structuring element B is denoted by $\phi_B(f)$ and is defined as the dilation of f with a structuring element B followed by the erosion with the reflected SE $\check{B}$:

$$\phi_B(f) = \varepsilon_{\check{B}}[\delta_B(f)], \tag{3.10}$$

which can also be written as:

$$\phi_B(X) = \bigcap_\mathbf{x} \{B^c_\mathbf{x} | X \subseteq B^c_\mathbf{x}\} \tag{3.11}$$

Morphology description and definitions based on [33].

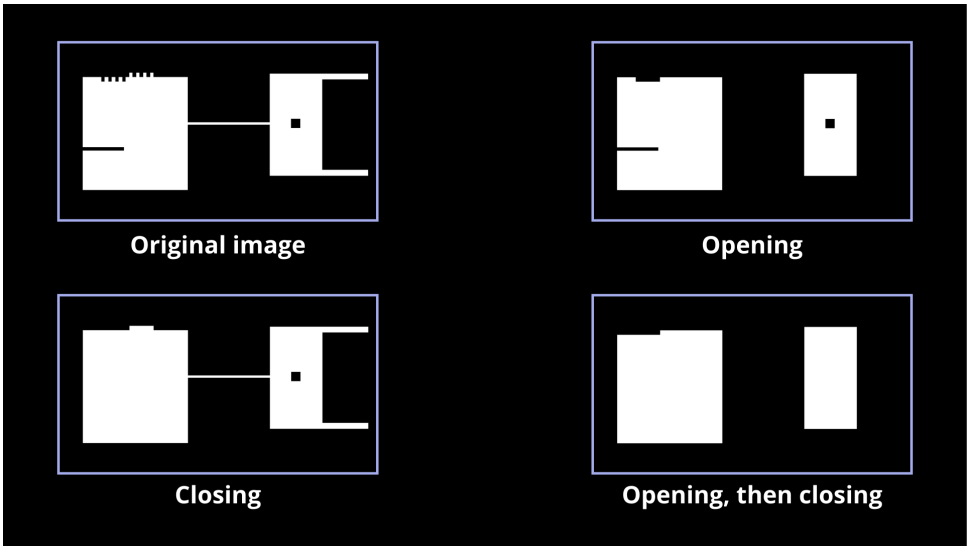


Figure 3.9: Example of closing and opening operations

3.4 Quality Control

Linacs, like any advanced medical device, need constant maintenance and calibration. The whole process of taking care of the device's performance is called *quality control* (QC).

One of the QC procedures for linacs is the *Winston-Lutz test* (W-L test). The test aims to control the mutual alignment of the mechanical isocenter of linac (the point where the gantry axis and couch axis intersect) designated using treatment room laser system and actual radiation beam isocenter.

To conduct the test, a series of images must be taken in a specifically prepared environment. Then, images are analysed to conclude the position of the radiation isocenter. [19] [30]

Another QC procedure that can be performed is the *leaves alignment analysis* (LA analysis) for MLCs. This procedure involves comparing the image taken, with the planned layout of leaves. If the detected area differs from the planned one, this means that the leaves are not aligning correctly, leading to erroneous generation of the radiation field by the linacs.

The environment setup and the process of capturing images are further explained in chapter 4. Image analysis procedure and description of algorithms can be found in chapter 5 - for W-L test, and in chapter 6 - for LA analysis.

3.5 Web applications

An application that runs in a browser is a *web application*. It is a very popular type of software considering its ease of use and portability for the average user and will continue to grow in future. [40]

3.5.1 Benefits of web apps

Web applications have several benefits that make them a good choice for modern software development. The following section outlines some of the most significant ones.

Accessibility

Web apps can be accessed from any device that has access to a modern browser. People from different locations can access shared documents, files and other data, work on them synchronously and see changes made by others in real-time, without the need to exchange any additional data.

Efficient development

The process of developing an app is relatively simple. There is no need to manage the distribution of the application - if a new version is deployed, every user can use it instantly. Additionally, since any version can work on any modern browser, there is no need to maintain different iterations of the product significantly reducing the resources needed for its maintenance.

Simplicity for users

Users are not required to download apps and no additional software is required to be installed on the user's machine, which makes them easy to access and eliminates the need for user maintenance. Web apps are always up to date since they automatically receive security and software updates.

Scalability

Adding new capacity, like users or available space is easy, especially if data is stored in the cloud. There is no need to install or change any existing infrastructure. [42]

3.5.2 Architecture of web applications

Web apps have a *client-server architecture*. The codebase is divided into two components - *client-side* and *server-side*.

Client-side architecture

Client-side scripts and data are loaded when a user visits the address of a web page in his browser. This is part of the application that displays data and provides interactive interface functionality (like buttons and forms) for the user. For this reason, it is often referred to as application *frontend*.

When a user clicks on a button or link in the interface, the web browser loads and launches a script that reacts to this action rendering some page element, called *component*, or communicating with the server-side of the application.

Server-side architecture

Server-side code manages data processing and persistence. Users cannot directly interact with server-side functionality, only through the frontend interface. Therefore this part of the application is often called *backend*. The backend often uses databases to store data and communicate with other servers to process complex data. [42]

3.5.3 User Interface (UI)

User interface is the point of contact for interaction and communication between humans and electronic devices. This can include display screens, keyboards, mice, and the appearance of a desktop. It also refers to the way a user interacts with buttons, animations, sounds, typefaces, and other visual and audio components on a website or application. [41]

3.5.4 Communication between frontend and backend

There are many different ways and protocols in which the frontend and backend can communicate, but one of the most popular is the *RESTful API*.

Application Programming Interface (API) is a mechanism that enables two different software components to communicate with each other using a set of protocols and definitions.

REST is built on top of Hypertext Transfer Protocol (HTTP) and enables programmers to define a set of operations that can be performed on resources.

The resource is any data or content, that exists on the server. Resources are uniquely identified by a predefined address. This address of resource is called *endpoint*.

The most common operations are:

1. POST

   Used to create previously non-existing resources.

2. GET

   Used to read data of resource.

3. PUT

   Used to update existing resources.

4. DELETE

   Used to delete existing resources.

Communication is based on the request-response cycle. The client sends a request to the server, server processes this request and sends back the response. All calls are stateless, which means that RESTful service cannot retain anything between calls. Data persistence is achieved with the usage of external databases. [29]

3.5.5 Open source software

In contrast to closed source, commercial software, *open source software* is released under a license that gives users the right to use, change and distribute the software and its source code to anyone for any purpose. The important thing is that users get insight into the project's source code, which they can change and redistribute in other projects. [36]

The development of this kind of application in a scientific environment is very important. Rather than providing a closed system that can process data on a one-time basis (often for a fee), developers are offering a comprehensive tool that can be reused in a range of other applications.

Chapter 4

Environment configuration

This chapter describes the environment configuration needed to conduct the W-L test and the LA analysis, including an example of the linear accelerator setup from the National Institute of Oncology in Kraków. Images generated by the described linac will be used in later chapters.

4.1 Configuration of environment for W-L test

1. A small BB phantom (ϕ around 3 mm), made of material with a high atomic number (typically steel, titanium or tungsten) is placed on the top of the couch and positioned at the mechanical isocenter determined by the treatment room laser system.
2. The radiation beam is collimated to a perpendicular or cylindrical shape by the MLC, thereby ensuring that its centre coincides with the collimator axis and thus also passes through the mechanical isocenter.
3. EPID is positioned on the opposite side of the BB so that the beam from the collimator falls on it.
4. Number of images are generated for different combinations of gantry, collimator and couch angles. All angles need to be saved as metadata as it is needed by the algorithm.
5. The resulting images can then be analyzed to identify any deviations from the norm, specifically when the shadow cast by the phantom is not located in the centre of the radiation field. [10]

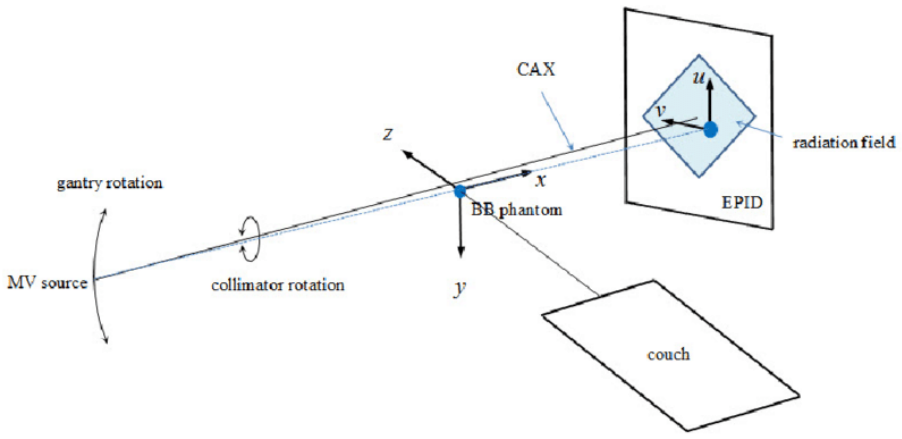


Figure 4.1: Visualisation of W-L test environment configuration
[49]

4.2 Configuration of environment for LA analysis

The EPID is positioned in such a way that the radiation field is fully contained within it. In the LA analysis, the angles of the gantry, couch or collimator do not matter, however for the algorithm to work correctly, the photo must be arranged so that the leaves move horizontally, along the lower and upper edges of the resulting image. Nevertheless, the photo can be pre-processed and arranged correctly using digital techniques. The metadata that needs to be recorded are the planned leaf positions and the dimensions of the resulting image.

Images taken in the W-L test configuration may be used for the LA test if they meet the above requirements.

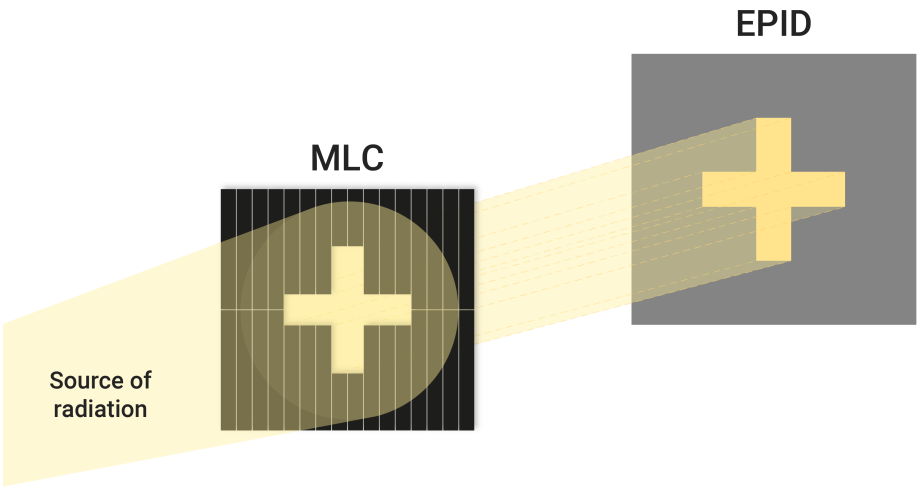
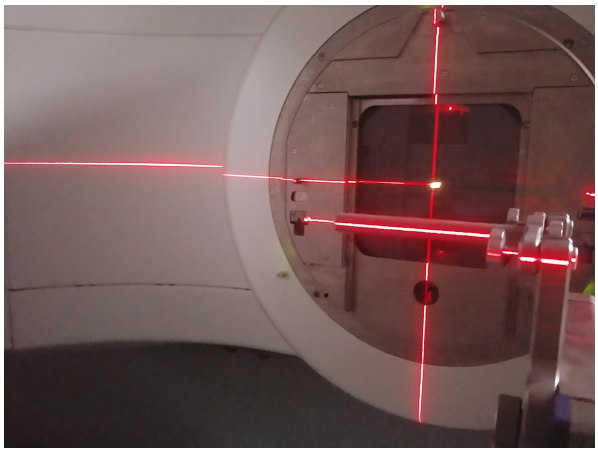


Figure 4.2: Visualisation of LA analysis environment configuration

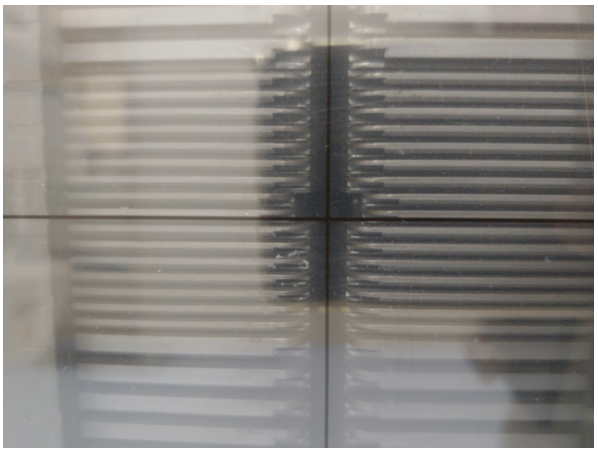
4.3 Example of linac configuration from the National Institute of Oncology in Kraków

The accelerator used is Varian TrueBeam [39]. BB with a diameter of 2mm was placed in fixed on top of the coach and positioned using room laser system - Figure 4.4a. EPID was positioned on the opposite side of the BB from the collimator - Figure 4.3. The leaves in the MLC have been formed so that the radiation field created by the radiation beam is a straight line with a square with a side of 10mm in the middle - Figure 4.4b.

Figure 4.3: Setup of Varian TrueBeam linac



(a) Positioning of BB using laser system



(b) Layout of leaves in MLC

Chapter 5

Winston-Lutz test module

This chapter describes the structure, functionality and algorithms used in the W-L test module. For technical description and implementation in application see section 8.2.

5.1 Pylinac and coordinate space

The module uses the Pylinac library (Winston-Lutz module) as its underlying framework. [25]. The coordinate system used in Pylinac conforms to IEC 61217 coordinate space. [47]

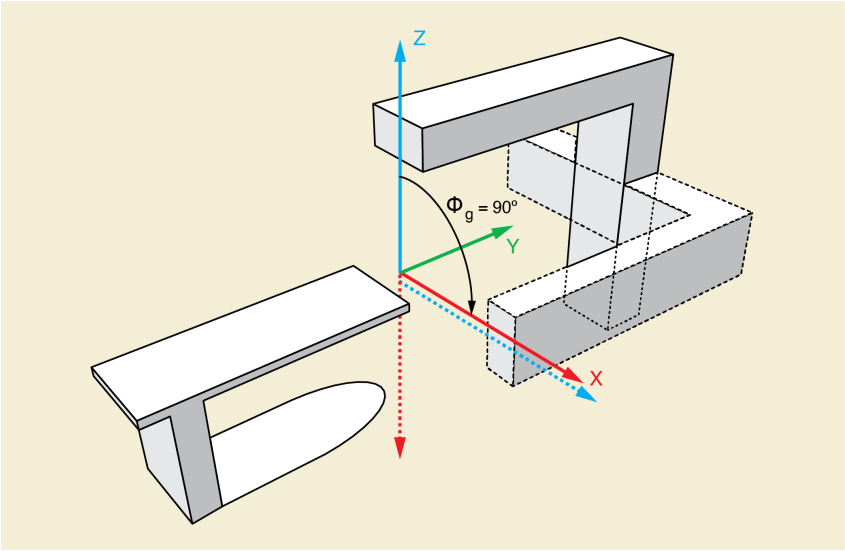


Figure 5.1: Coordinate space used in Winston-Lutz pylinac module [47]

The structure and algorithms described below are encapsulated in the Pylinac library, which provides an API to perform the test that uses them. The module described in the thesis performs the W-L test, using Pylinac's methods. However, the library returns raw data and plots, which are then processed and organised in the application to be presented concisely in the application frontend.

5.2 Features

- **Couch shift instructions**

  Module provides feedback how to shift the couch to relocate the mechanical isocenter to match the radiation isocenter.

- **Automatic field and BB positioning**

  CAX and BB are found automatically, along with the vector and scalar distance between them.

- **Isocenter size determination**

  The 3D gantry isocenter size and position can be determined using back projections of the EPID images, independently of the BB position.

- **Axis deviation plots**

  Variations of the gantry, collimator, couch, and EPID are plotted together with their root mean square variation.

- **Isocenter visualization**

  The 3D plot of the isocenter and BB is generated.

- **PDF report generation**

  PDF report with all the above data summed up can be generated.

5.3 Interface structure

The module loads and processes EPID images and outputs resulting data in the form of numerical values, plots and images.

5.3.1 Inputs

- Valid W-L test EPID images (defined in subsection 5.4.2)
- Size of BB

5.3.2 Ouptuts

Numerical values

- Couch shift instructions
- Number of total images
- Number of gantry images
- Number of couch images
- Number of collimator images
- Maximum 2D CAX to BB distance
- Median 2D CAX to BB distance
- Mean 2D CAX to BB distance
- Maximum 2D CAX to EPID distance
- Median 2D CAX to EPID distance
- Mean 2D CAX to EPID distance
- Gantry 3D isocenter diameter
- Maximum Gantry RMS deviation
- Maximum Collimator RMS deviation
- Maximum Couch RMS deviation
- Maximum EPID RMS deviation
- Gantry + Collimator 3D isocenter diameter
- Collimator 2D isocenter diameter
- Couch 2D isocenter diameter

Plots

- In-plane Gantery displacement
- In-plane Epid displacement
- In-plane Collimator displacement
- In-plane Couch displacement
- Isocenter Visualisation

Images

- Gantry wobble
- Collimator wobble
- Couch wobble
- Gantry images
- Collimator images
- Couch images
- GB Combo images
- GBP Combo images

5.4 Algorithm

The algorithms presented in this section are not the author's work; they are adapted from the pylinac documentation [25]. However, these materials facilitate comprehension of the test methodology, the generation of results, and, consequently, the functioning of the application. For this reason, they are presented here.

5.4.1 Algorithm allowances

- Images from any EPID can be used, with any SID.
- Even though BB distance from real isocenter affects 2D image analysis, it does not need to be near it to determine isocenter sizes.

5.4.2 Algorithm restrictions

- The BB must be within 2.0cm of the real isocenter.
- The images must be obtained with the EPID.
- The BB must be completely within the field of view.
- The linac coordinate system must be IEC 61217.

5.4.3 Algorithm definition

1. Find the field CAX

   Pixel span ($max - min$) is divided by 2. Any pixels above the threshold are considered open field. Pixels are binarized and CAX is assumed to be in the centre of mass of the pixels.

2. Find the BB

   The algorithm is described in subsection 5.4.4.

3. Calculate couch shift vector

   Calculations as defined in subsection 5.4.5

4. Backproject the CAX for gantry images

   Based on the gantry angle and ξ (Equation 5.2), a 3D line projection of the CAX is designated. The origin is considered at BB. For this step, only images with the coach at angle 0 are considered.

5. Determine gantry isocenter size

   Using the above, the maximum distance to any line is minimized. This distance is the gantry isocenter radius.

6. Determine collimator isocenter size

   Maximum distance between field CAX locations for any two images is taken as collimator isocenter size.

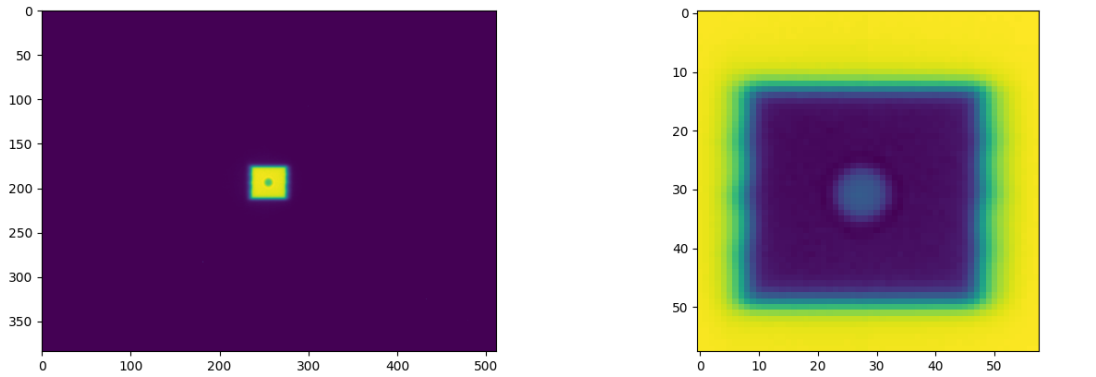
7. Determine couch isocenter size

   The maximum distance between BB for any two images is taken as isocenter size. CAX location of an image where the gantry, collimator and couch angles are equal to 0 is taken as a reference.

5.4.4 Algorithm for finding the BB

1. Images are cropped if needed. The cropped section is later referred to as the *sample*.

2. Image is inverted if needed. Because pylinac uses the weighted centroid method to find the BB, it needs to "weigh" more at the centre than at the edges of the BB.



(a) Original image

(b) Cropped and inverted sample image

Figure 5.2: Cropping and invertion [46]

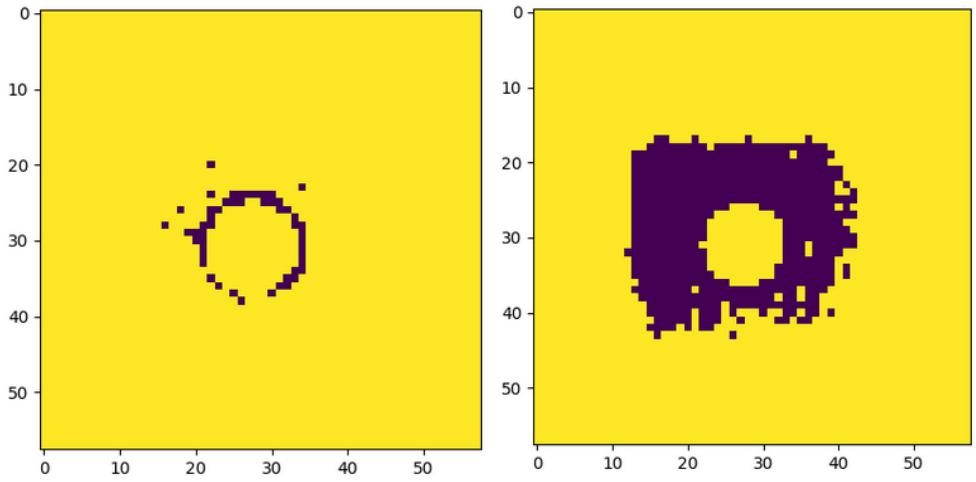
3. The sample is normalized to a range of 0-1.

4. The sample is converted to binary using thresholding. The threshold starts at a value of 0.02 and any pixel with a value below this is set to 0, otherwise, it is set to 1. Then each blob is checked if it passes all detection conditions. *Detection conditions* are defined depending on the use case, and for the BB finding algorithm it is defined as

   (a) Blob is round,

   (b) blob has an area within the expected range given by the BB size,

   (c) blob has a circumference within the expected range given the BB size.



(a) Image at the start of the thresholding process

(b) Image after several thresholding process iterations

Figure 5.3: Thresholding [46]

If no blob passes all detection conditions, the threshold is increased by 0.02, and the detection process is repeated. If BB is detected, the thresholding algorithm will stop. If BB is not detected, the algorithm will continue until the threshold reaches 1.0.

5. Center and boundary of detected BB is returned. If no BB is detected by then, error is raised.

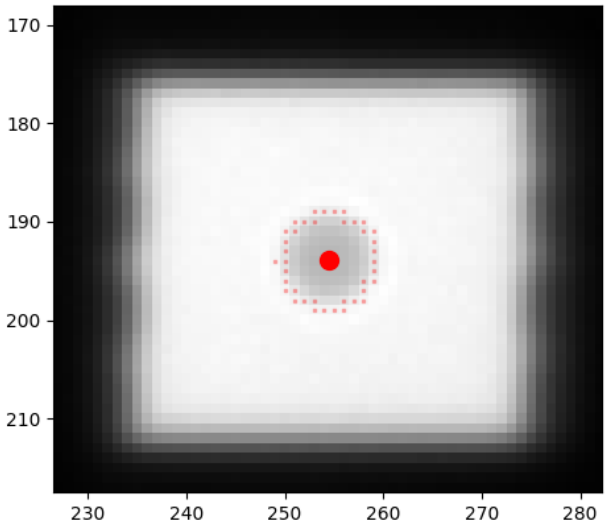


Figure 5.4: Image with found BB. [46]

5.4.5 Calculating couch shift

Geometric transformations of 2D planar images can be used to calculate shifts that could be applied to BB to match the radiation isocenter. [5]. For each image $\mathbf{A}$ is needed to be determined:

$$\mathbf{A}(\phi,\theta) = \begin{pmatrix} -\cos(\phi) & -\sin(\phi) & 0 \\ -\cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) & -\sin(\theta) \end{pmatrix} \tag{5.1}$$

where

- θ is the gantry angle
- ϕ is the couch angle

Then, the ξ matrix is calculated:

$$\xi = (y_1, -x_1, ..., y_i, -x_i, ..., y_n, -x_n)^T \tag{5.2}$$

where x_i and y_i are the scalar shifts from the field CAX to the BB for n images.
Having the above $\mathbf{B}(\phi_1, \theta_1, ..., \phi_n, \theta_n)$ can be calculated:

$$\mathbf{B} = \begin{pmatrix} \mathbf{A}(\phi_1,\theta_1) \\ \vdots \\ \mathbf{A}(\phi_i,\theta_i) \\ \vdots \\ \mathbf{A}(\phi_n,\theta_n) \end{pmatrix} \tag{5.3}$$

Shift vector $\mathbf{\Delta}$ can be solved using $\mathbf{B}$ and ξ:

$$\mathbf{\Delta} = \begin{pmatrix} \Delta Y \\ \Delta X \\ \Delta Z \end{pmatrix} \tag{5.4}$$

Couch shift calculation and algorithm descriptions based on [25], which itself is based on [44], [11] and [5].

Chapter 6

Leaves alignment analysis module

This chapter describes the structure, functionality and algorithms used in the LA analysis module. For technical description and implementation in application see section 8.2.

6.1 Features

- **Automatic detection of discrepancies in leaves alignment**

  Leaves that do not align with the given specification are automatically detected on a given valid EPID image. Each leaf is investigated separately. Leaves planned position are specified as JSON file.

- **Determination the divergence distance**

  Once the dimensions of the input image have been defined, the algorithm returns the distance of each divergent leaf from its planned position.

- **Visually highlighting the discrepancies on images**

  The locations where the leaf deviates from the specified tolerance or is not detected are visually marked on the image.

6.2 Interface structure

The module loads one EPID file along with metadata and returns processed data about not-aligned leaves in the form of a list with errored leaves, and images of where the deviations are graphically marked. In the current state of application development, data such as image size and leaf definition are entered manually. However, in the future, these values could be extracted from the DICOM RT Plan file.

6.2.1 Inputs

Numerical values

- Horizontal real size of image (x) [mm]
- Vertical real size of image (y) [mm]
- Tolerance x [N pixels]

  Distance of the detected leaf edge from the planned edge, above which a divergence is recognised.

- Tolerance y [N pixels]

  Distance of the horizontal, upper or lower edge of the leaf within which divergences are ignored. This is a method of mitigating the potential imperfections of the algorithms.

- Permitted errors per leaf

  The number of rows in which discrepancies are detected in a single leaf, beyond which it is considered faulty.

- Binarisation threshold

  Threshold value of the binarisation used in step 1 of Algorithm for determining edges of leaves (subsection 6.3.4)

- SE size

  Size of the side of the square used as a structuring element in step 3 and 9 of Algorithm for determining edges of leaves (subsection 6.3.4).

- Sobel kernel size

  Size of kernel used in Sobel filter in step 4 and 6 of Algorithm for determining edges of leaves (subsection 6.3.4).

Files

- Valid EPID image (defined in subsection 6.3.2)
- JSON file containing leaves specification

  The file should contain a list of objects with keys:

  - `id`: id of leaf
  - `y_t`: y coordinate of upper edge of leaf
  - `y_b`: y coordinate of bottom edge of leaf
  - `x`: x coordinate of vertical edge of leaf
  - `side`: side from which the leaf is located

6.2.2 Outputs

Numerical values

- List of divergent leaves with a mean distance between detected and planned edge

Images

- Original image
- Preprocessed image with detected edges of leaves
- Image where left leaves alignment is visualised with colour
- Image where right leaves alignment is visualised with colour
- Above images with outlined leaves

6.3 Algorithm

The algorithms presented in this module have been developed by the author for this module and the thesis.

6.3.1 Algorithm allowances

- Images from any EPID can be used, with any SID.
- W-L test-type images can be used as long as they conform to restrictions.

6.3.2 Algorithm restrictions

- All leaves's edges must be visible.
- Image is arranged so that the leaves move horizontally, along the lower and upper edges of the resulting image.
- The image, once loaded, must be convertable to greyscale and the area where radiation fell must be darker than the non-radiated areas.
- No gap exists between the upper and lower edges of leaves.

6.3.3 Algorithm definition

1. Determine the edges with algorithm defined in subsection 6.3.4.
2. Detect upper and lower boundaries of the radiation field.
3. For each horizontal row of pixels in boundaries, check if the detected edge is within the specified tolerance, and mark the leaves and detected edges accordingly. Perform this step separately for images with left and right edges.
   (a) Mark the points in the current row which lie on the detected edge in green if they are in specified tolerance from the planned edge.
   (b) Mark the points in the current row which lie on the detected edge in red if they are further away from the planned edge than the specified tolerance.
   (c) Mark the points in the current row which lie on the detected edge in yellow if they are further away from the planned edge than the specified tolerance, but are close to the upper or lower edge of the leaf.
   (d) Mark the whole row in the corresponding colour if the edge is not detected.
4. For each leaf, count the number of rows where problems highlighted in red are found, and if the number is greater than the specified parameter, mark the leaf as faulty.

6.3.4 Algorithm for determining edges of leaves

1. Binarise image.
2. Invert image.
3. Apply morphological closing operation to ensure continuity of radiation field.
4. Apply Sobel filter for direction x on the image from step 3.

   The result of this operation will be an image in which the pixel values are equal to the calculated gradient of the surrounding pixels in the horizontal direction. Pixels located on the "left" edge will have different values than those located on the "right" edge. More precisely, one set of values will be positive, while the other will be negative.
5. Perform two separate binarizations of the image from step 4 with two different thresholds, separating the image into an image with left edges, and an image with right edges.
6. Apply Sobel filter for direction y on the image from step 3.
7. Convert image from step 6 by calculating gradient magnitude for each pixel.(Equation 3.5)
8. Change the value of all pixels to no edge in the image from step 5 if in the image from step 6 corresponding pixel contains an edge. Perform this step separately for images with left and right edges.
9. Apply morphological closing separately on both images from step 8 to ensure continuity of edges.

Chapter 7

Used technologies and tools

7.1 Frontend

7.1.1 HTML, CSS and JS

HyperText Markup Language (HTML) is used to define the structure of a web page. While HTML code alone is sufficient for displaying a basic static page, it is always used in combination with the other two technologies that give the page its appearance and interactivity. [16]

Cascading Style Sheets (CSS) is a list of rules that are used to enhance the visuals of a web page. This language determines the actual appearance of elements defined with HTML. [3]

JavaScript is a scripting language used to add interactivity to web pages and define custom logic of HTML elements. JS is a weakly typed language, which means that different values can be used where another type is expected, and the language will perform necessary conversions to make that possible. Although this approach is convenient for quick prototyping, it can be problematic in the long term, particularly in situations, where a value of the wrong type is used by mistake. The language will permit that usage during the coding process, and this could result in an error during the execution of a program (in this case often on user's web page).

To prevent such problems, a variant of JS called *TypeScript* (TS) was developed. TS adds support for static types, and it is *transpiled* to JS, which means, that TS is used only during development, when tooling helps programmers to write good code, and after development codebase is translated to JS which is run by browsers. [37]

7.1.2 React

React is a library for JS language that improves the process of building interfaces. The interface is composed of individual pieces called components, which are parts of code that encapsulates the interface element with its logic and are written in a functional and declarative way. React is the most popular frontend library and has a rich ecosystem as well as numerous extensions.[28]

7.1.3 React Router

React Router is module that enables seamless page switching called *client side routing*. In contrast to traditional websites, where each new page is requested and loaded from the server, along with CSS and JS assets, client-side routing enables an app to update the page without making another request. Instead, the app can immediately switch parts of the UI to different components. This approach allows a faster and more dynamic user experience. [27]

7.1.4 Axios and fetching data from backend

Traditionally, every time a page needed to be reloaded, for example, after a user interaction, the browser sent a request to the server, which provided new data and necessary files. However, this process makes the user experience worse, since upon loading new pages, all the data has to be acquired all over again. A modern approach to designing web apps is different - the user interface is rendered by browsers on the client side, and the server is only queried for new data when it is needed in a process called *data fetching*. This approach greatly reduces the amount of data sent over the network and therefore speeds up the UI.

Axios is a library that integrates with the JS and React ecosystem and gives you the ability to fetch the data it needs. [1]

7.1.5 Tanstack Query

Tanstack Query is a module that enables declarative and automatic data fetching. It simplifies the fetching process by encapsulating the imperative logic of working with requests and allows programmers to specify only the source of data and how fresh it needs to be able to fetch and use it. It also provides mechanisms like caching and background updates out of the box. [35]

7.1.6 Tailwindcss and daisyUI

Tailwindcss is a framework for CSS that simplifies writing styles by providing pre-written classes that can be used directly in components. It provides a straightforward way to declare concise components that are responsive and performant, and out-of-the-box support for sizing, colours, typography, shadows, theming and hover and focus states. [34]

DaisyUI is a component library for TailwindCSS that provides ready-to-use components and styles that can be further modified with tailwindCSS. [4]

7.2 Backend

7.2.1 Python

Python is an interpreted, scripting language with great support for data structures and an extensive ecosystem of data analysis tools, which makes it a great tool to work with data-oriented tasks. [43]

7.2.2 Django and DRF

Django is a web framework for Python which provides the necessary tools for setting up a production-ready server that can be utilised in building a web application. [9]

Django REST framework (DRF) is a toolkit for building web APIs with built-in support for web browsable API, authentication, data serialization, and ORMs. [8]

7.2.3 Pylinac

Pylinac is an open-source library for Python that provides QC tools in the field of therapy and diagnostic medical physics. It contains modules for automatic analysis of images and data obtained from linear accelerators, CT simulators, and other radiation oncology equipment. The library also provides lower-level modules and tools for image analysis algorithms. [22]

7.2.4 OpenCV

OpenCV is an open-source computer vision library for Python and many other languages. OpenCV provides a comprehensive set of tools for all types of image analysis. [21]

Chapter 8

Application Technical Description

The objective of this thesis is to extend an existing open-source application, named *dose3d*, which provides a comprehensive set of ready-to-use tools that are beneficial in the field of radiation therapy. The development comprises the creation of two modules: one enabling the W-L test and the other enabling leaves alignment analysis. A theoretical description of the modules can be found in chapter 5 and chapter 6, respectively. This chapter describes the application, its design, and its implementation, divided into frontend and backend.

8.1 Frontend

8.1.1 UI

Structure

The tabs containing links to both modules can be found on the side panel in the modules section - Figure 8.1. Each module consists of two pages - a setup page and a results page. The setup page contains a form, where the user can specify input data and two buttons, one for submitting the form and the other for aborting operation. All fields need to be filled in to proceed, otherwise the submit button will be disabled. When a form is submitted, the algorithm is executed and the results are displayed on the results page.

W-L test setup page

The W-L test setup page consists of two inputs, one numerical for specifying BB size, and the other one for files. Only the zip file can be selected. Once the file has been uploaded correctly, the field will be highlighted in green. W-L setup page is shown on Figure 8.2.

W-L test results page

The W-L test results page contains two sections - one with numerical data and one with graphical data in the form of plots. The page displays immediately, but the algorithm needs time to process the data. During loading, a spinner is shown in the relevant section.

The numerical data is presented in a grid format, with a variable number of columns depending on the width of the page. Hovering the cursor over a `i` label next to the corresponding section name, will reveal a pop-up window, providing a detailed description of the section.

In the top right corner, there is a button to download a PDF report containing a summary of the test. W-L results page is shown on Figure 8.3.

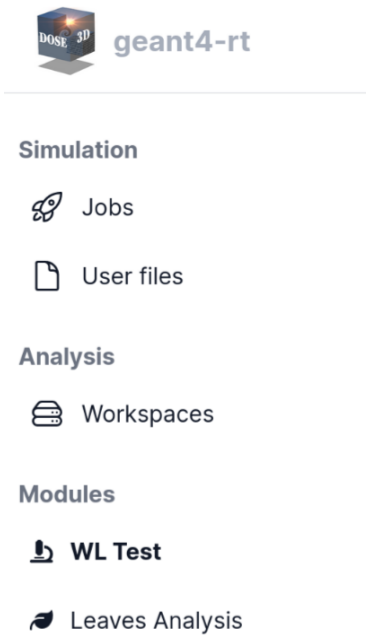


Figure 8.1: Side panel with links to application modules

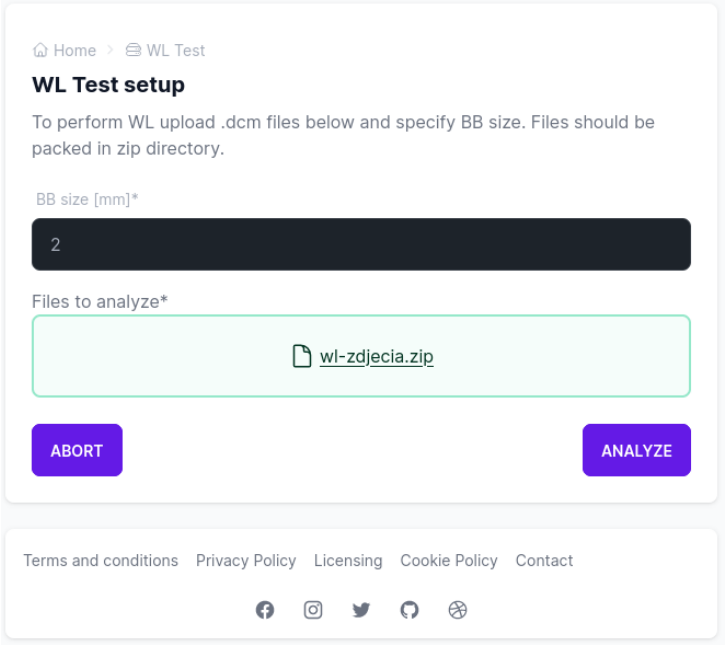


Figure 8.2: W-L test setup page

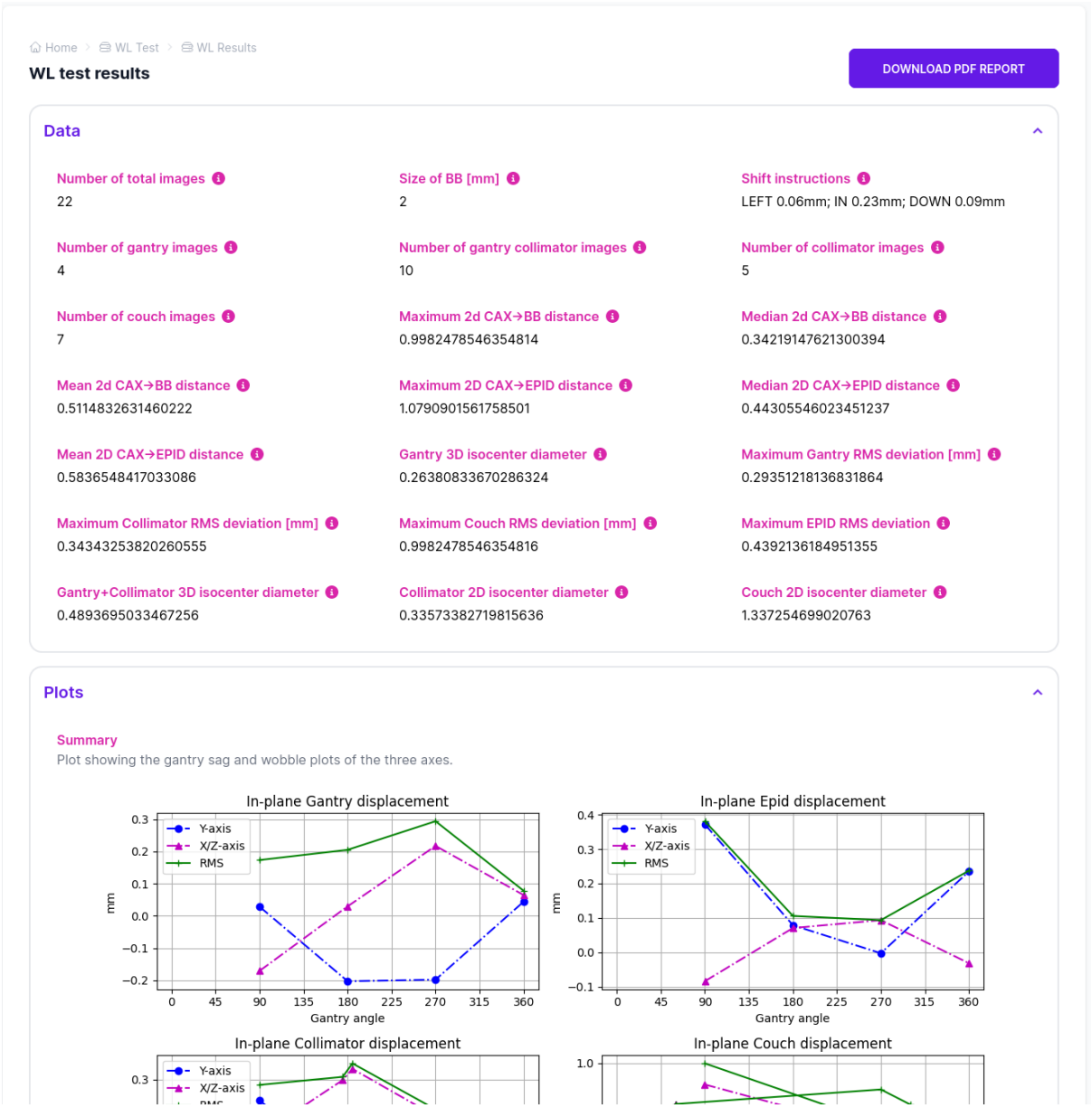


Figure 8.3: W-L test results page

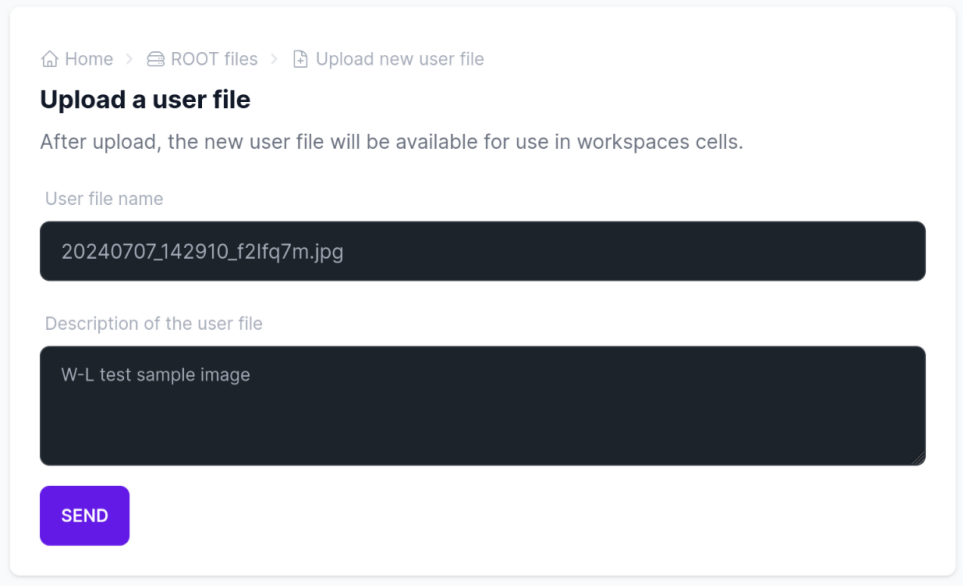


Figure 8.4: Common interface for uploading files

LA analysis setup page

LA analysis setup page consists of eight numerical inputs and two file selectors. Files for this module are uploaded via the existing common file upload interface for the entire application. Interface is shown in Figure 8.4.

Once all the fields have been filled in and the files selected, a preview of the preprocessed will be available. This allows the parameters of the algorithm to be adjusted so that the edges of the radiation field are correctly detected. LA setup page is shown on Figure 8.5.

LA analysis results page

The results page shows the original image, the preprocessed image, the resulting image for the left leaves and the resulting image for the right leaves.

A list of faulty leaves is listed below on the left. On the right-hand side, a legend for the colours can be found.

In the top left-hand corner of the page, there is a toggle switch which, when clicked on, brings up the visualisation of the leaves on all 4 images. LA analysis results page can be found on Figure 8.6.

8.1.2 Implementation

W-L Test

The setup component contains forms with one numerical-only field and a second file field that uploads the file to the server. The file is hashed to save space and data and cache the data for results. If there is the same file already present on the server it is overwritten. BB size and filename are sent as a parameter to the results page.

The result page component has 3 separate React Query hooks that separately fetch numeric data, plots and pdfs, to speed up loading. Once loaded, each element is displayed independently of the others. The data is cached and the key consists of filename and BB size, with a stale time of 5 minutes. Therefore, if the user executes a query with the same parameters within 5 minutes, the data is loaded from the cache and is displayed immediately without the need to make another request to the backend.

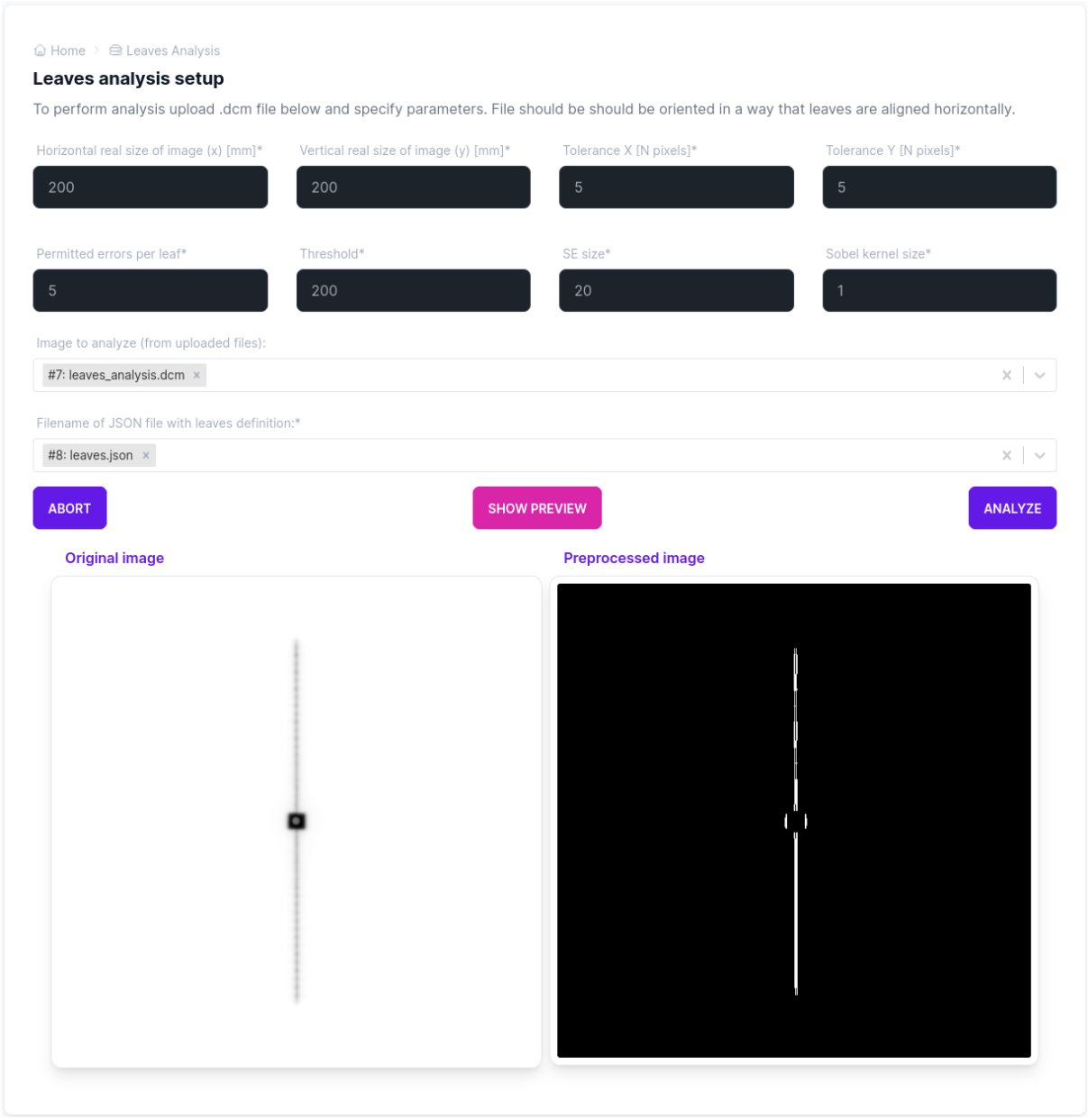


Figure 8.5: LA analysis setup page

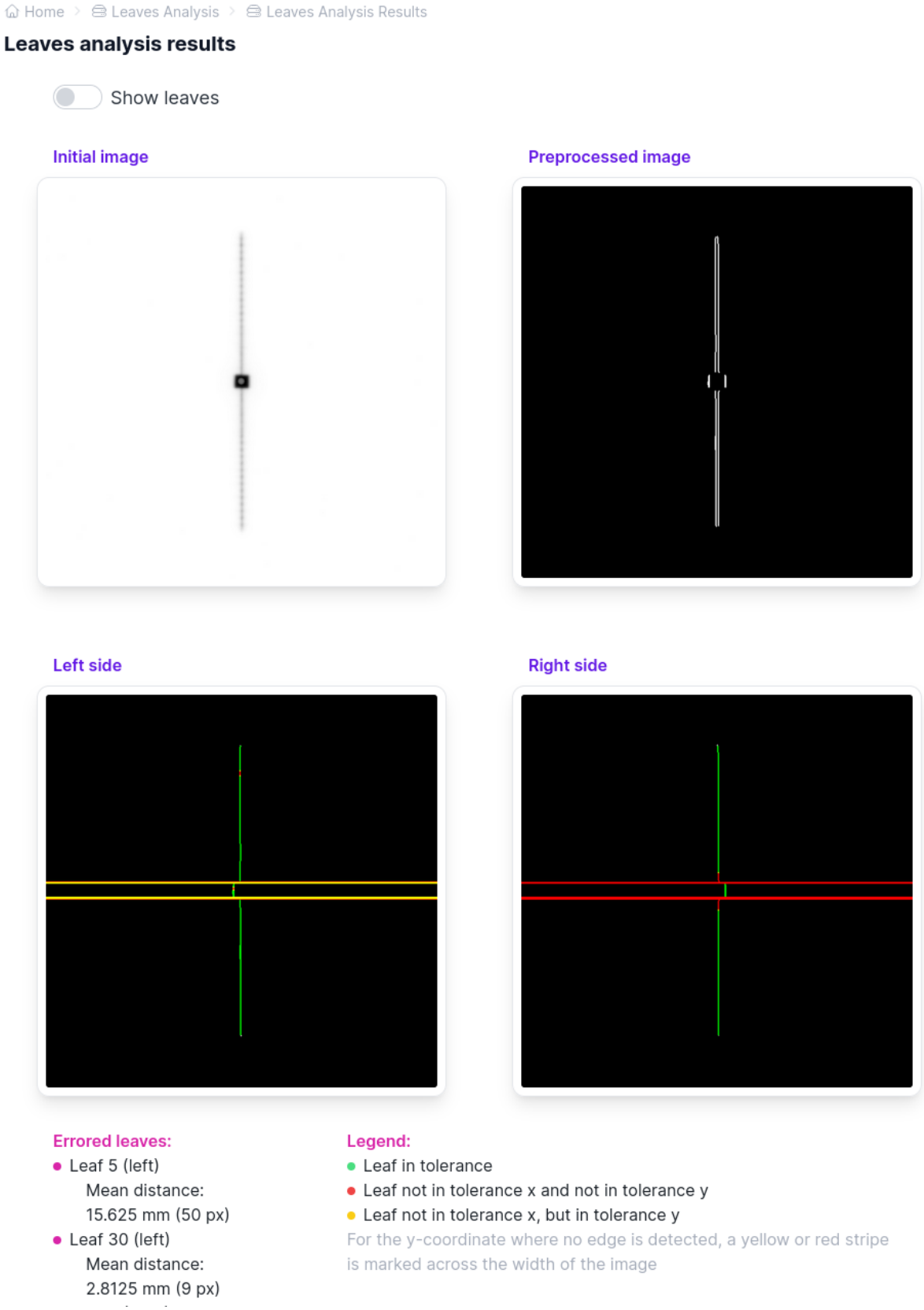


Figure 8.6: LA analysis results page

LA analysis

On the setup page, there are eight fields in which only numbers can be entered. Furthermore, there are two fields for selecting files that have previously been uploaded via the common interface. After selecting a file, the server is queried and returns the names of the uploaded files, which are then displayed as options to the user.

When the `show preview` button is clicked, a request to the backend is made, and the server returns a pre-processed image. As with the W-L test, it is cached, with the test parameters being the keys. This means that if a user switches back to the previously used parameters, the preview images will be loaded immediately from the cache. In contrast to the `show preview` button, which does not switch pages, clicking on the `analyze` button will take the user to the results page.

As LA analysis does not take long, the results page contains only one React Query hook. The server then returns the data, along with two variants of the images - with and without leaves drawn. When the toggle is clicked, the images are replaced on the frontend without additional requests to the server.

8.2 Backend

8.2.1 Structure

The modules have been added to the application in a separate folder `contrib`. The LA analysis module is more straightforward and consists of a `views.py` file with two views that manage the endpoints defined below and a second file with algorithm definitions.

W-L module also includes analogous `views.py` file and algorithm declarations files, as well as `models.py`, `serializer.py` and `utils.py` files. These files are responsible for handling uploads that overwrite files with the same name to optimize storage.

Additionally, both modules contain a `apps.js` file, which is a required component in the Django framework for applications located in separate folders.

8.2.2 API

W-L Test

Four endpoints are available in this module. 2 of them return data about the W-L test, and they have been split to enable independent data loading on frontend - text data, that loads faster, can be displayed earlier, and plots can be loaded in parallel.

Endpoints

- `/api/wl/text/`

  Endpoint for fetching text data results of W-L test.

  Type: GET

  Request payload: Filename and BB size.

  Response payload: All numerical data of W-L test as defined in section 5.3.2

- `/api/wl/plots/`

  Endpoint for fetching plots of W-L test.

  Type: GET

  Request payload: Filename and BB size.

  Response payload: Zip file with plot of isocenter visualisation and summary plot.

- `/api/wl/pdf/`

  Endpoint for downloading PDF report.

  Type: GET

  Request payload: Filename and BB size.

  Response payload: PDF file with results of W-L test.

- `/api/wl/upload/`

  Endpoint for uploading images that will be used in the W-L test.

  Type: POST

  Request payload: Zip file with W-L EPID images.

  Response payload: Status of operation.

LA Analysis

Two endpoints are available in this module, one for the preprocessing of images and a second for full LA analysis.

Endpoints

- `/api/la/preprocess/`

  Endpoint for fetching preprocessed images.

  Type: GET

  Request payload: Filename of image to analyze, binarization threshold, structuring element size, Sobel kernel size.

  Response payload: Zip file with original image and image after preprocessing (subsection 6.3.4)

- `/api/la/analyze/`

  Endpoint for fetching result images.

  Type: GET

  Request payload: All inputs defined in subsection 6.2.1.

  Response payload: All outputs defined in subsection 6.2.2. Images are zipped into the zip file.

Chapter 9

Analysis evaluation

This chapter consists of five example analyses - three for the W-L test and two for the LA analysis. In each case, the inputs and results of the test are defined, and the entire process is concluded with the author's commentary on the analysis.

9.1 WL test

9.1.1 Pylinac Demo

Pylinac has a built-in demo which can be started by calling the appropriate function. Sample images are then loaded and the W-L test is performed in the standard way.

Inputs

- 17 test images
- BB size of 5 mm

Numerical results

- Couch shift instructions: LEFT 0.05mm; OUT 0.26mm; DOWN 0.20mm
- Maximum 2D CAX to BB distance: 1.24mm
- Median 2D CAX to BB distance: 0.69mm
- Mean 2D CAX to BB distance: 0.62mm
- Maximum 2D CAX to EPID distance: 2.36mm
- Median 2D CAX to EPID distance: 1.31mm
- Mean 2D CAX to EPID distance: 1.26mm
- Gantry 3D isocenter diameter: 1.03mm
- Maximum Gantry RMS deviation: 1.01mm
- Maximum Collimator RMS deviation: 0.78mm
- Maximum Couch RMS deviation: 1.24mm
- Maximum EPID RMS deviation: 1.32mm
- Gantry + Collimator 3D isocenter diameter: 1.10mm
- Collimator 2D isocenter diameter: 1.08mm
- Couch 2D isocenter diameter: 2.35mm

Plots

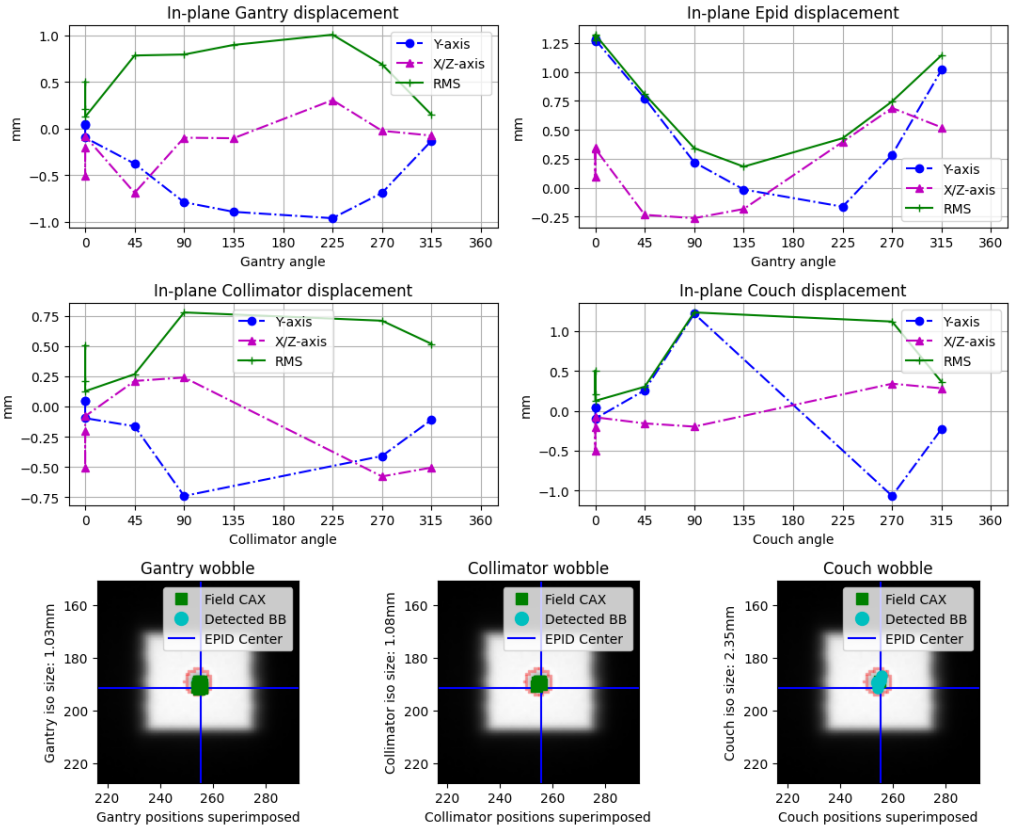


Figure 9.1: Summary plot of demo test

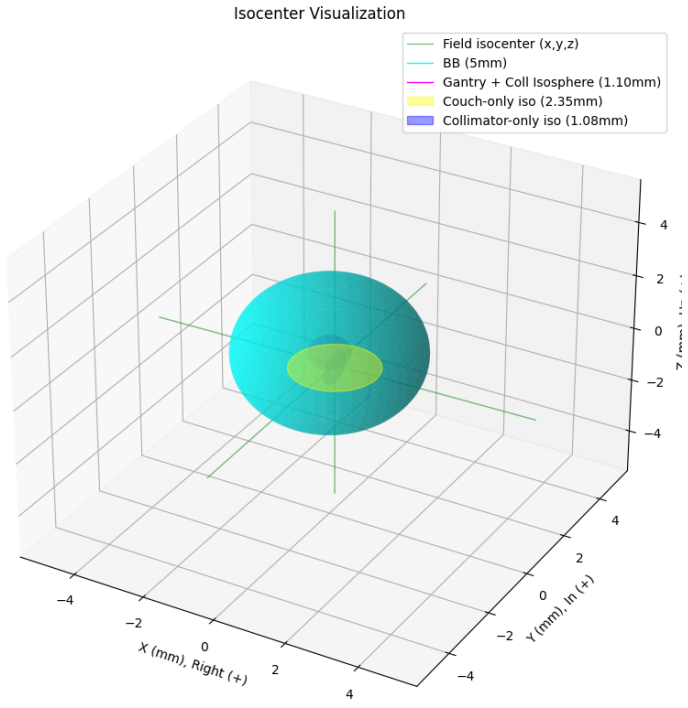


Figure 9.2: Isocenter plot of demo test

Analysis

As can be seen, the results obtained indicate that the module is working correctly. The results are consistent with those published in the module's documentation [26].

9.1.2 Test on real images

This analysis is conducted on real DICOM images obtained from a W-L test session at the National Institute of Oncology in Kraków (section 4.3).

Inputs

- 22 images with dimensions 640x640 px.
- BB size of 2mm

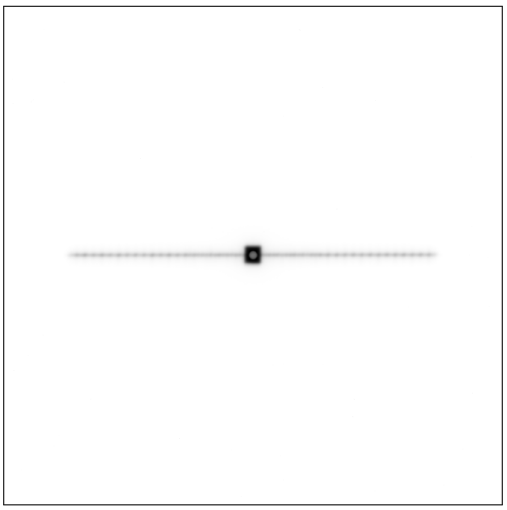


Figure 9.3: One of the input images

Numerical results

- Couch shift instructions: LEFT 0.06mm; IN 0.23mm; DOWN 0.09mm
- Maximum 2D CAX to BB distance: 0.99mm
- Median 2D CAX to BB distance: 0.34mm
- Mean 2D CAX to BB distance: 0.51mm
- Maximum 2D CAX to EPID distance: 1.07mm
- Median 2D CAX to EPID distance: 0.44mm
- Mean 2D CAX to EPID distance: 0.58mm
- Gantry 3D isocenter diameter: 0.26mm
- Maximum Gantry RMS deviation: 0.29mm
- Maximum Collimator RMS deviation: 0.34mm
- Maximum Couch RMS deviation: 0.99mm
- Maximum EPID RMS deviation: 0.43mm
- Gantry + Collimator 3D isocenter diameter: 0.48mm
- Collimator 2D isocenter diameter: 0.33mm
- Couch 2D isocenter diameter: 1.33mm

Plots

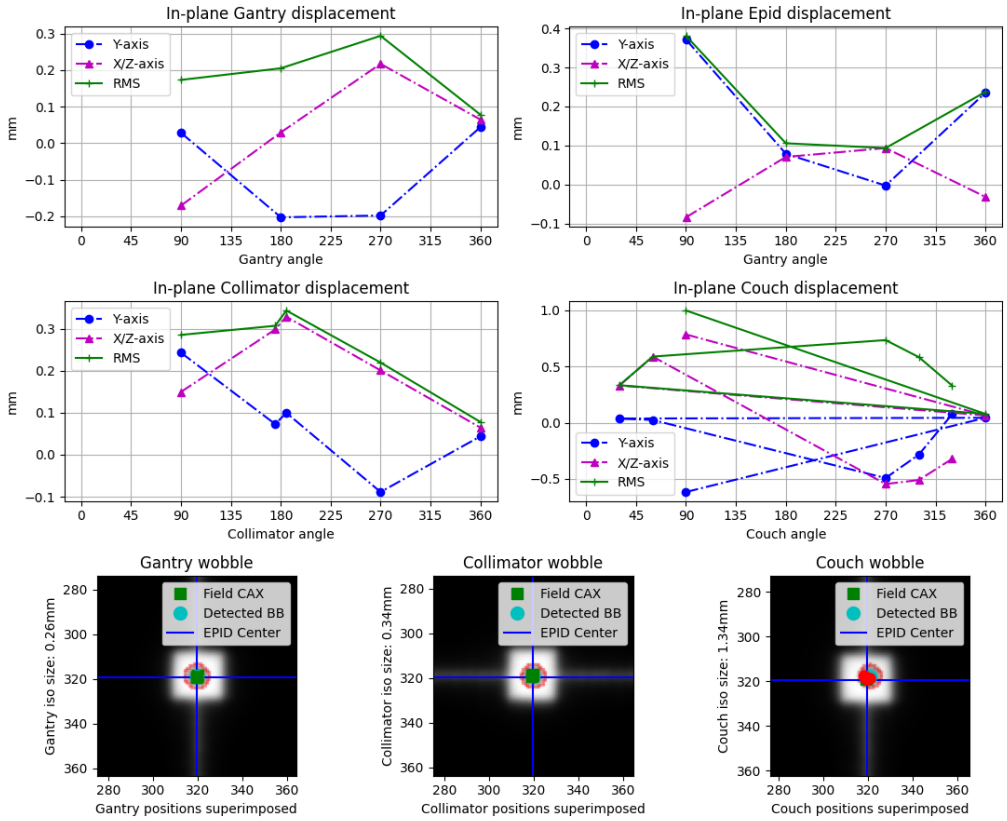


Figure 9.4: Summary plot of test on real images

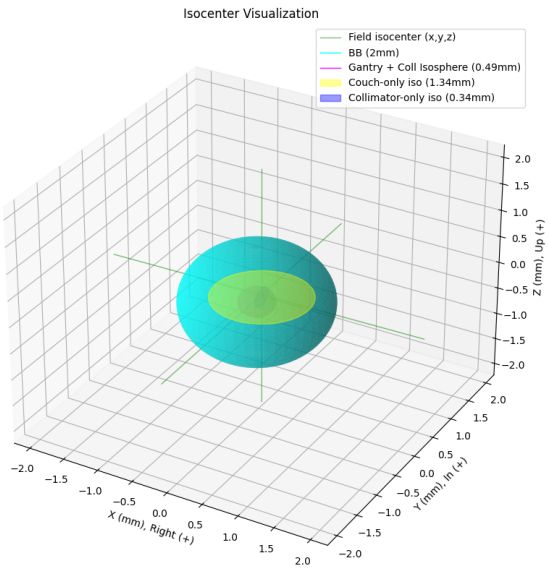


Figure 9.5: Isocenter plot of test on real images

Analysis

As can be observed, the displacement values are relatively small, both in terms of the shift instructions and the individual detected BB distances from the CAX from which they result. This aligns with the initial hypothesis, as an examination of the images manually reveals that the BB is situated close to the centre of the radiation field in all images. Additionally, the visualisation of isocenters demonstrates that they are properly aligned. Consequently, it can be concluded that the machine has been correctly calibrated.

9.1.3 Test on artificial images

This analysis is conducted on a generated set of images that have been modified using Pylinac's Offset BB function, which was originally developed for use in benchmarking. [23]. This method involves artificially moving the detected BB by a given distance in space, to evaluate the impact of this manipulation on the results.

Inputs

- 18 generated images, where the position of BB has been shifted 2 mm in Y direction, 4 mm in X direction, and 6 mm in Z direction
- BB size of 4 mm

Numerical results

- Couch shift instructions: RIGHT 2.01mm; OUT 4.06mm; DOWN 6.17mm
- Maximum 2D CAX to BB distance: 7.60mm
- Median 2D CAX to BB distance: 4.54mm
- Mean 2D CAX to BB distance: 5.38mm
- Maximum 2D CAX to EPID distance: 7.34mm
- Median 2D CAX to EPID distance: 4.54mm
- Mean 2D CAX to EPID distance: 5.27mm
- Gantry 3D isocenter diameter: 0.07mm
- Maximum Gantry RMS deviation: 7.34mm
- Maximum Collimator RMS deviation: 4.54mm
- Maximum Couch RMS deviation: 4.54mm
- Maximum EPID RMS deviation: 0.00mm
- Gantry + Collimator 3D isocenter diameter: 0.28mm
- Collimator 2D isocenter diameter: 0.04mm
- Couch 2D isocenter diameter: 9.08mm

Plots

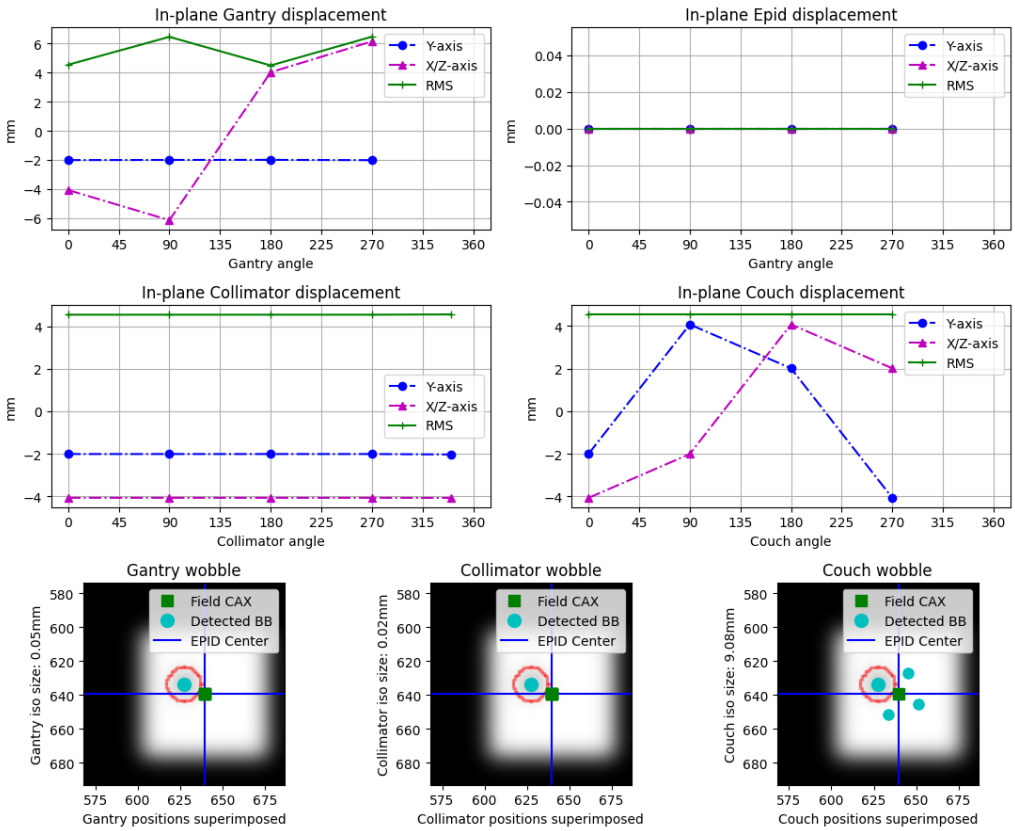


Figure 9.6: Summary plot of test with artificial images

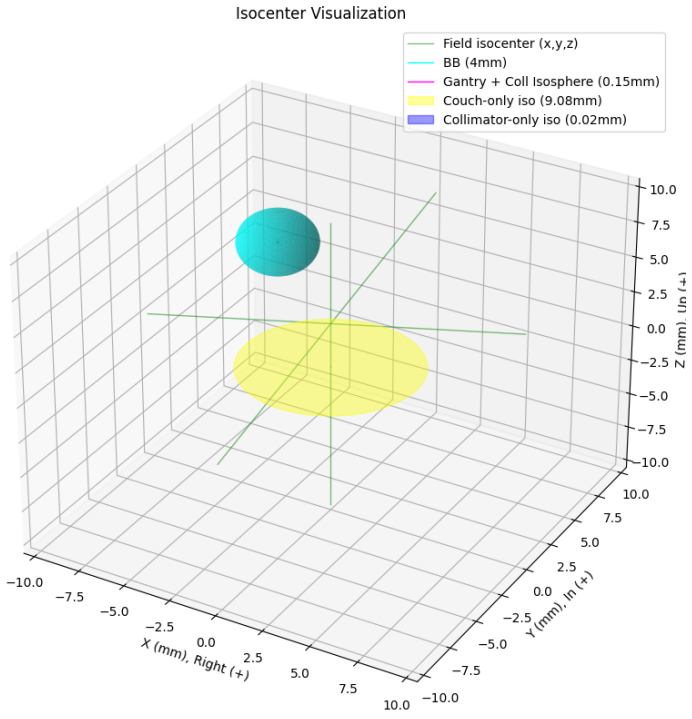


Figure 9.7: Isocenter plot of test with artificial images

Analysis

The results are highly in line with the predictions, showing a significant deviation of BB from the norm. As can be seen in the visualisation, with these parameters the device, the isocentres are not aligned, and thus it can be concluded, that the device is not correctly calibrated, as anticipated.

9.2 Leaves analysis

9.2.1 Real image with correctly calibrated leaves

This analysis is conducted on a real DICOM image obtained with the device from the National Institute of Oncology in Kraków (section 4.3).

Numerical inputs

- Horizontal real size of an image (x): 428.8mm
- Vertical real size of an image (y): 428.8mm
- Tolerance x: 2 px
- Tolerance y: 2 px
- Permitted errors per leaf: 1
- Binarisation threshold: 215
- SE size: 25 px
- Sobel kernel size: 1 px

Leaves definition

Leaves have been defined to correspond to their intended arrangement (Figure 9.8). The radiation field comprises a total of 60 leaves, of which 32 are 2.5 mm wide and 28 are 5 mm wide. The thinner leaves are concentrated in the centre of the field, allowing more precise modelling of the field. The thicker leaves are located at both ends of the field, with 14 at the top and 14 at the bottom. In the middle of the radiation field, the leaves form a square with a side of 10mm.

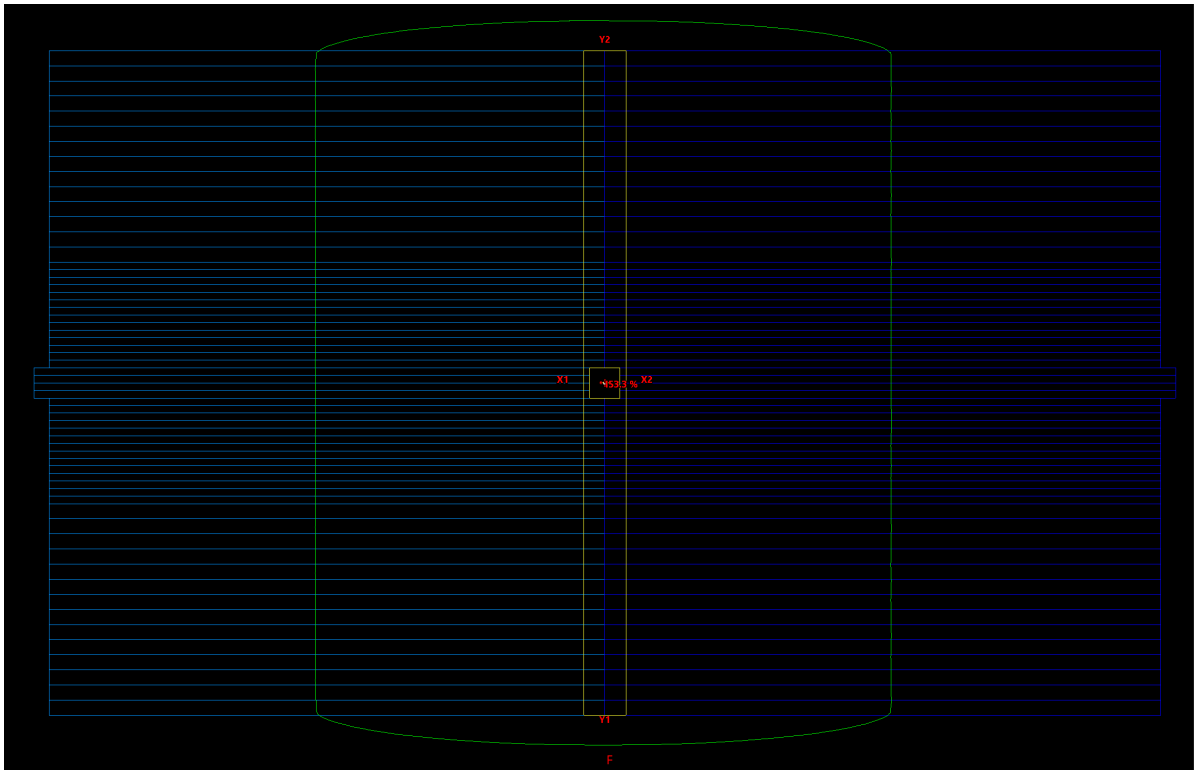


Figure 9.8: Leaves alignment from example plan

Input image

Figure 9.9: LA analysis input image

Results

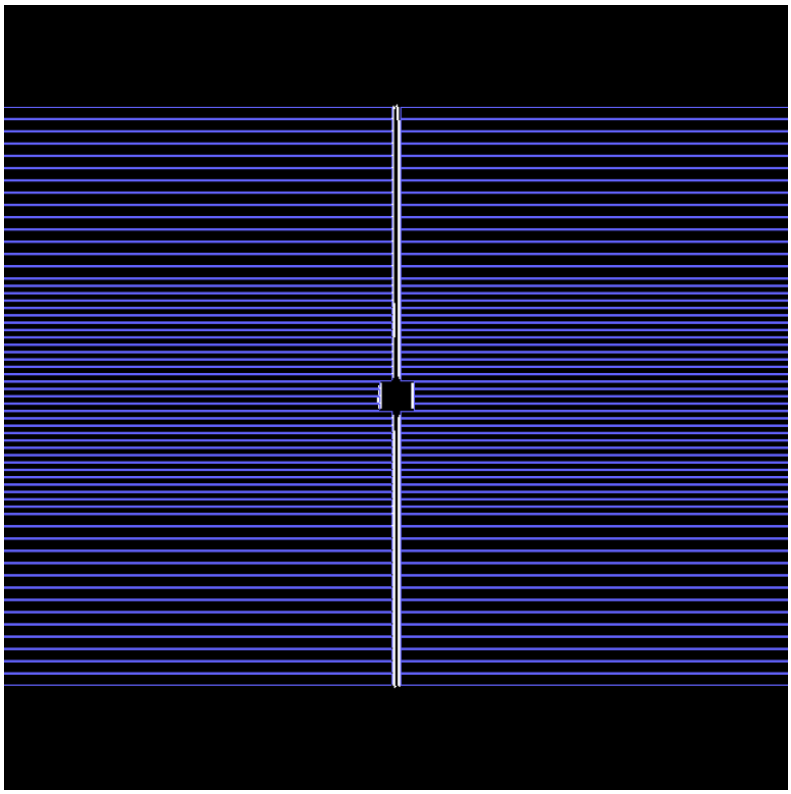


Figure 9.10: Preprocessed image

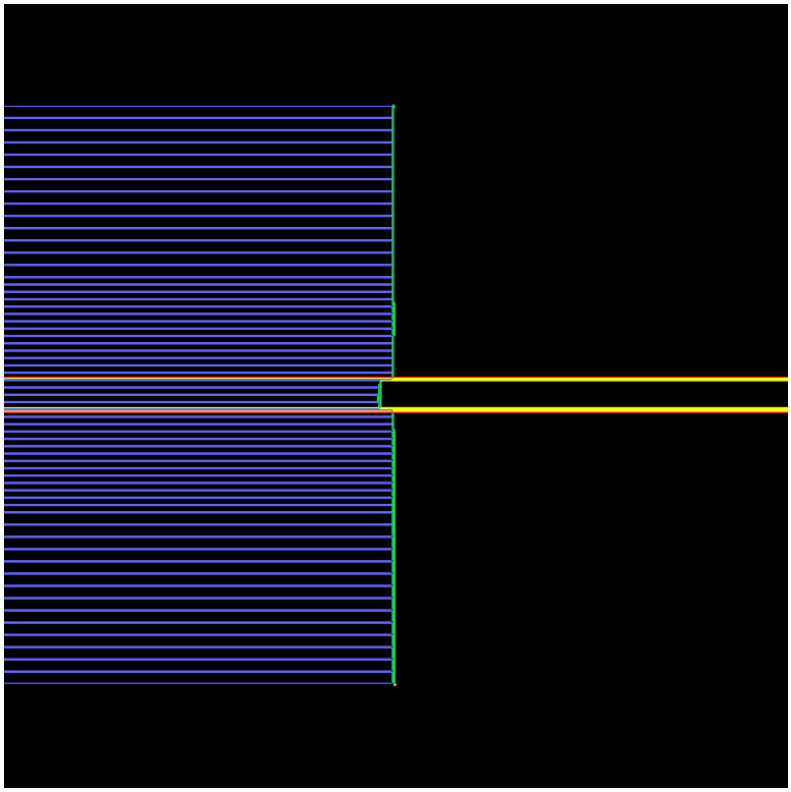


Figure 9.11: Analyzed leaves left

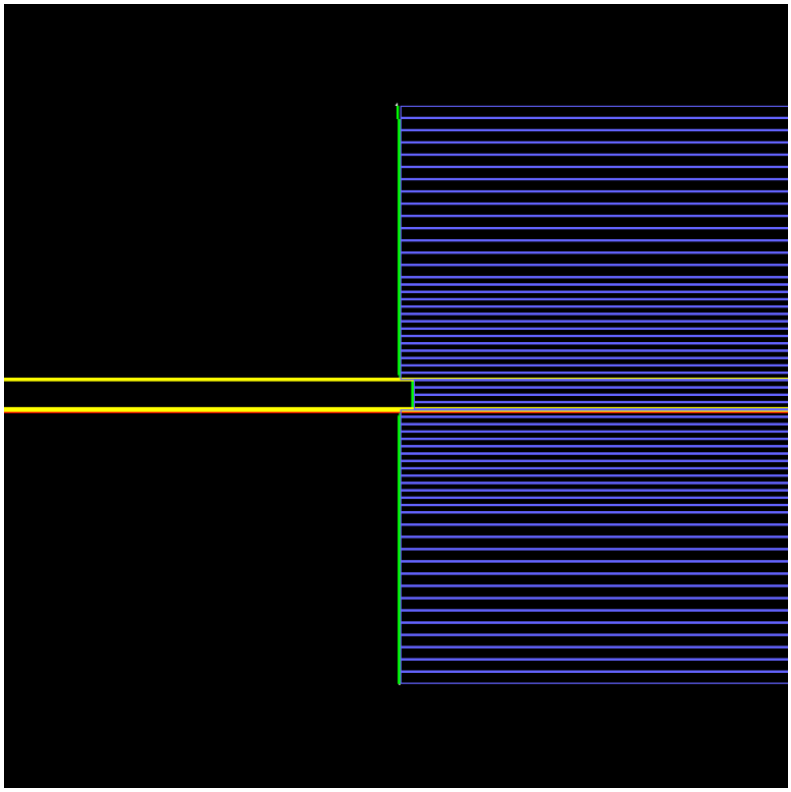


Figure 9.12: Analyzed leaves right

Detected faulty leaves

The algorithm detected no faulty leaves.

Analysis

As can be seen, no faulty leaves were detected in this case. In the resulting images, two places can be seen, near the edges of the radiation field, where no edge was detected (yellow and red horizontal stripes). This is due to imperfections in the edge detection algorithm at the point of sharp changes in their direction. However, these imperfections are within the tolerances, that is, the number of counted errors (red) is less than the limit, and are therefore not considered as non-calibrated leaves. Erroneous values close to the edges of the leaves have been correctly marked in yellow and are not counted as errors. Consequently, based on the test, it can be concluded that the leaves are correctly calibrated.

9.2.2 Real image with incorrectly calibrated leaves

In this analysis, the input image is identical to that used in subsection 9.2.1, but the definition of the leaves has been modified artificially to introduce faulty leaves to demonstrate how the algorithm will behave and what the results will look like in such a situation.

Inputs definition

The input image and numerical inputs are the same as in subsection 9.2.1. The left side of the leaves definition is similar to that of subsection 9.2.1, with the difference that the x position of the edge of the two leaves has been changed. One has been extended to the left by 50 pixels, while the other has been extended to the right by 10 pixels. The right side has been replaced by 3 big leaves which group together adjacent leaves, to visualise the problem better.

Result images

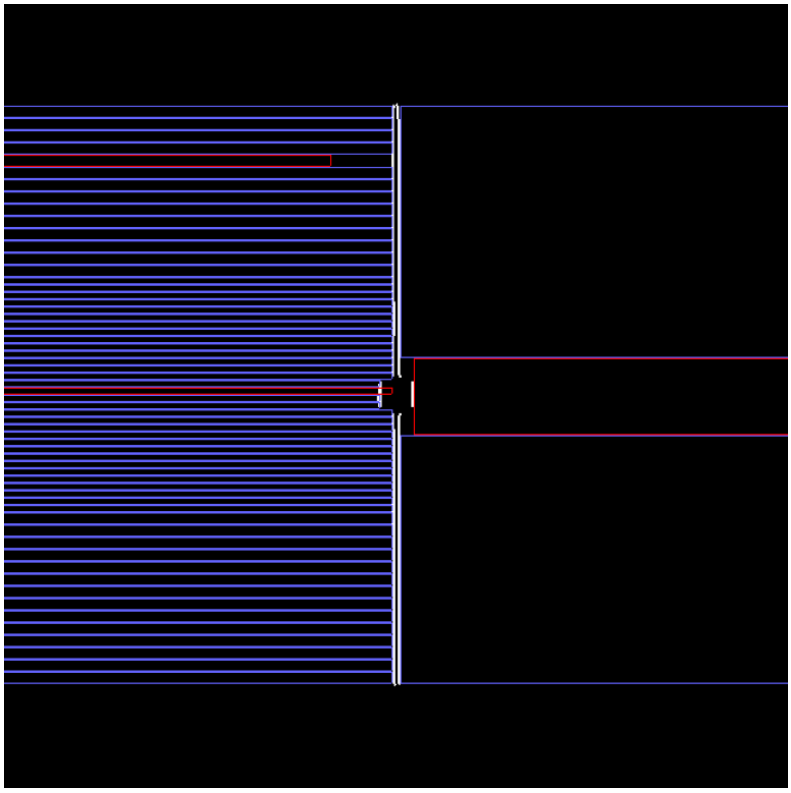


Figure 9.13: Preprocessed image

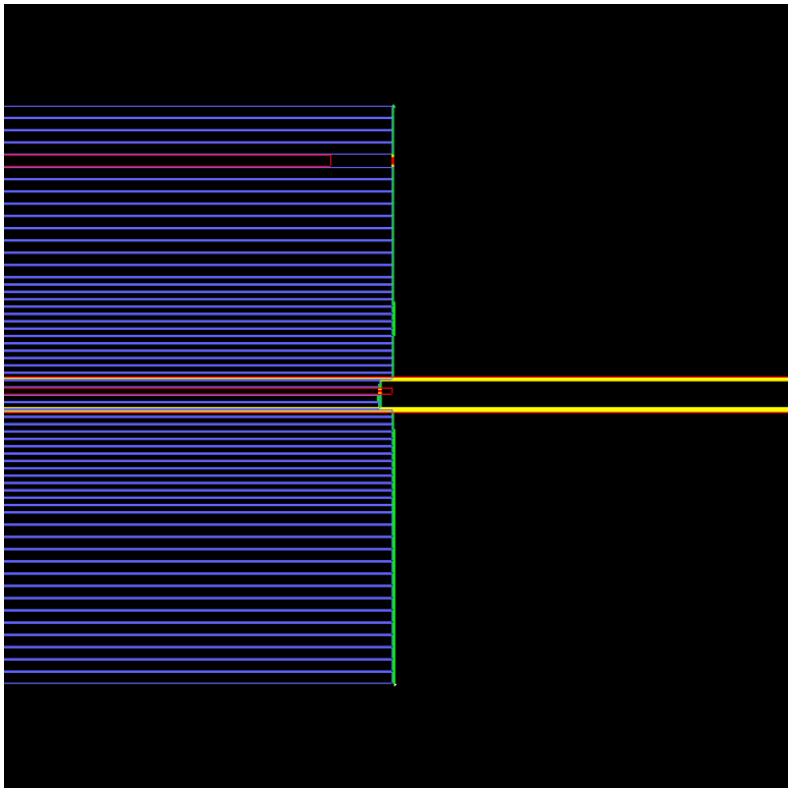


Figure 9.14: Analyzed leaves left

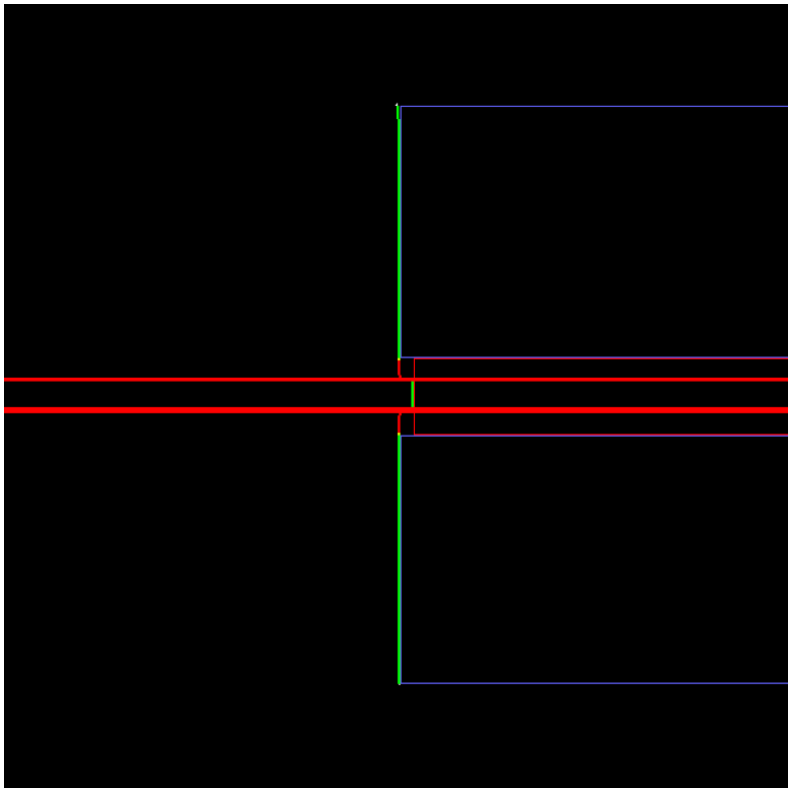


Figure 9.15: Analyzed leaves right

Faulty leaves

The algorithm detected 3 faulty leaves, 2 on the left and 1 on the right. The top left faulty leaf has a mean deviation of 33.4mm (50px), the bottom left faulty leaf has a mean deviation of 6mm (9px), and the right faulty leaf has a mean deviation of 8mm (12px).

Analysis

As can be observed, the algorithm correctly identified all three miscalibrated leaves. Both left leaves were successfully detected, along with their respective deviations. For the upper leaf, the detected deviation is precisely the amount by which the leaf was shifted. For the lower leaf, there is a discrepancy of one pixel, from the set up and detected positions. This is expected, given that the detected leaf edge thickness exceeds one pixel, thereby reducing the detected distance.

As for the right side, the algorithm also correctly detected a slightly more difficult case, where the detected edges indicate that the leaf should be thinner than defined. Even though the centre of the leaf is at the correct offset, the number of erroneous values closer to the edge exceeds the allowed amount, with the result that the leaf is classified as faulty. It is worth noting that also in this case, the algorithm correctly maintains the tolerance of the nearest points near the edge of the leaf definition.

Chapter 10

Conclusions

The objective set out in the introduction has been met. The tests conducted demonstrate that the developed modules operate as intended and can be used to perform the Winston-Lutz test and leaf alignment analysis.

The fact that the tools have been developed as part of a web application makes them highly accessible for use on any device with access to a modern browser. Moreover, the application's open-source nature opens up a wide range of possible applications in situations where it was previously impractical, frequently because of resource constraints.

However, it should be mentioned that the application is currently in the prototype stage and has not been certified. As such, it should not be applied to the calibration of medical equipment meant to be utilized during procedures on human beings. At this stage of development, the application is only a proof of concept. Nevertheless, it does demonstrate the potential of such solutions, which, with sufficient effort, could greatly facilitate the operation of numerous facilities utilising linear accelerators, given that such devices will require calibration regularly.

The analysis of the results of the W-L tests reveals that in instances where the test is expected to yield accurate results, it does so. Furthermore, in cases that should identify the machine as uncalibrated, results also match. This proves that the module has been correctly embedded in the application.

Observing the analysis of the results for LA in an analogous way shows the correct implementation of the module, giving the expected results. Further interpretation, however, leads to the conclusion that the more complex the shape, the greater the problem the edge detection algorithm will have, which can lead to erroneous indications.

The outcomes of the work has been reviewed by Damian Kubat, the head of the Department of Medical Physics at the National Institute of Oncology in Kraków. He provided a positive feedback on its implementation and indicates that it is a valuable tool, especially if developed further.

Bibliography

[1] *Axios vs. Fetch API: Selecting the Right Tool for HTTP Requests*. URL: https://medium.com/@johnnyJK/axios-vs-fetch-api-selecting-the-right-tool-for-http-requests-ecb14e39e285 (visited on 08/19/2024).

[2] *Blob Detection Using OpenCV*. 2024. URL: https://www.geeksforgeeks.org/blob-detection-using-opencv/ (visited on 08/17/2024).

[3] *Coding for Web Design 101: How HTML, CSS, and JavaScript Work*. URL: https://blog.hubspot.com/marketing/web-design-html-css-javascript (visited on 08/19/2024).

[4] *DaisyUI: The most popular component library for Tailwind CSS*. URL: https://daisyui.com/ (visited on 08/19/2024).

[5] Robert E. Drzymala Daniel A. Low Zuofeng Li. "Minimization of target positioning error in accelerator-based radiosurgery". In: *Medical Physics* 22 (4 Apr. 1995). DOI: 10.1118/1.597475. URL: https://doi.org/10.1118/1.597475.

[6] *DICOM specification - DICOM File*. 2024. URL: https://dicom.nema.org/medical/dicom/current/output/html/part12.html#sect_K.1.2 (visited on 08/05/2024).

[7] *DICOM specification - Introduction and Overview*. 2024. URL: https://dicom.nema.org/medical/dicom/current/output/chtml/part01/chapter_1.html (visited on 08/05/2024).

[8] *Django REST framework*. URL: https://www.django-rest-framework.org/ (visited on 08/19/2024).

[9] *Django the web framework for perfectionists with deadlines*. URL: https://www.djangoproject.com/ (visited on 08/19/2024).

[10] Weiliang Du et al. "On the selection of gantry and collimator angles for isocenter localization using Winston-Lutz tests". In: *Journal of applied clinical medical physics / American College of Medical Physics* 17 (Feb. 2016), p. 5792. DOI: 10.1120/jacmp.v17i1.5792.

[11] *Du et al.* URL: http://scitation.aip.org/content/aapm/journal/medphys/37/5/10.1118/1.3397452.

[12] James R. Egner. *Abeloff's Clinical Oncology*. Vol. 303. 11. Elsevier, Mar. 2010, pp. 1097–1097. DOI: 10.1001/jama.2010.288. eprint: https://jamanetwork.com/journals/jama/articlepdf/185518/jbk0317_1097_1097.pdf. URL: https://doi.org/10.1001/jama.2010.288.

[13] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN: 013168728X.

[14] Robert Hirsch. *Exploring Colour Photography : A Complete Guide*. Laurence King, London, 2005.

[15] *How does the Linear Accelerator work?* URL: https://www.radiologyinfo.org/en/info/linac (visited on 09/05/2024).

[16] *HTML basics*. 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (visited on 08/19/2024).

[17] Stephen Johnson. *Stephen Johnson on Digital Photography*. O'Reilly Media, Inc., 2006. ISBN: 059652370X.

[18] Reinhard Klette. *Concise Computer Vision*. Springer London, 2014. DOI: 10.1007/978-1-4471-6320-6.

[19] W Lutz, KR Winston, and N Maleki. "A system for stereotactic radiosurgery with a linear accelerator". In: *International journal of radiation oncology, biology, physics* 14.2 (Feb. 1988), pp. 373–381. ISSN: 0360-3016. DOI: 10.1016/0360-3016(88)90446-4. URL: https://doi.org/10.1016/0360-3016(88)90446-4.

[20] *Matlab Help Center: Image Inversion*. 2024. URL: https://www.mathworks.com/help/ecoder/beaglebone/ref/image-inversion.html (visited on 08/17/2024).

[21] *OpenCV*. URL: https://opencv.org/ (visited on 08/19/2024).

[22] *Pylinac*. 2024. URL: https://pylinac.readthedocs.io/en/release-v3.25.0/ (visited on 08/05/2024).

[23] *Pylinac Benchmarking Algorithm*. URL: https://pylinac.readthedocs.io/en/release-v3.23/winston_lutz.html#offset-bb (visited on 08/27/2024).

[24] *Pylinac documentation: Winston-Lutz module overview, v3.32*. 2024. URL: https://pylinac.readthedocs.io/en/release-v3.25.0/winston_lutz.html#module-pylinac.winston_lutz (visited on 08/10/2024).

[25] *Pylinac documentation: Winston-Lutz module overview, v3.32*. 2024. URL: https://pylinac.readthedocs.io/en/release-v3.23/winston_lutz.html#algorithm (visited on 08/14/2024).

[26] *Pylinac W-L test demo*. URL: https://pylinac.readthedocs.io/en/release-v3.23/winston_lutz.html#running-the-demo (visited on 08/28/2024).

[27] *React Router Feature Overview*. URL: https://reactrouter.com/en/main/start/overview (visited on 08/19/2024).

[28] *React, The library for web and native user interfaces*. URL: https://react.dev/ (visited on 08/19/2024).

[29] *RESTful API*. 2024. URL: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API (visited on 08/18/2024).

[30] Pejman Rowshanfarzad et al. "Isocenter verification for linac-based stereotactic radiation therapy: Review of principles and techniques". In: *Journal of applied clinical medical physics / American College of Medical Physics* 12 (Nov. 2011), p. 3645. DOI: 10.1120/jacmp.v12i4.3645.

[31] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2021.

[32] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[33] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2004. ISBN: 978-3-662-05088-0.

[34] *Tailwind: Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/ (visited on 08/19/2024).

[35] *TanStack Query*. URL: https://tanstack.com/query/v5 (visited on 08/19/2024).

[36] *The Open Source Definition*. 2024. URL: https://opensource.org/osd (visited on 08/18/2024).

[37] *TypeScript for the New Programmer*. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (visited on 08/19/2024).

[38] Scott E. Umbaugh. *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools, Second Edition*. 2nd. USA: CRC Press, Inc., 2010. ISBN: 143980205X.

[39] *Varian TrueBeam*. URL: https://www.varian.com/products/radiotherapy/treatment-delivery/truebeam (visited on 08/21/2024).

[40] *Web Application Development: Trends and Facts for 2024*. 2024. URL: https://www.geeksforgeeks.org/blob-detection-using-opencv/ (visited on 08/17/2024).

[41] *What is a user interface (UI)?* URL: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI (visited on 09/02/2024).

[42] *What is a Web Application?* 2023. URL: https://aws.amazon.com/what-is/web-application/ (visited on 08/17/2024).

[43] *What is Python? Executive Summary*. URL: https://www.python.org/doc/essays/blurb/ (visited on 08/19/2024).

[44] Peter Winkler et al. "Introducing a system for automated control of rotation axes, collimator and laser adjustment for a medical linear accelerator". In: *Physics in Medicine & Biology* 48.9 (Apr. 2003), p. 1123. DOI: 10.1088/0031-9155/48/9/303. URL: https://dx.doi.org/10.1088/0031-9155/48/9/303.

Images Attributions

[45] AGH. https://www.agh.edu.pl/o-agh/multimedia/znak-graficzny-agh/znak-graficzny-bez-nazwy.

[46] Pylinac source code. https://github.com/jrkerns/pylinac/blob/release-v3.25.0/docs/source/images. MIT License: Copyright (c) 2014-2022 James Kerns Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

[47] *DICOM coordinate system definition*. URL: https://dicom.nema.org/medical/dicom/current/output/chtml/part03/sect_c.8.8.25.6.html.

[48] OpenCV Documentation. https://docs.opencv.org/4.10.0/d7/d4d/tutorial_py_thresholding.html. Apache 2 Licence.

[49] Weiliang Du et al. "On the selection of gantry and collimator angles for isocenter localization using Winston-Lutz tests". In: *Journal of applied clinical medical physics / American College of Medical Physics* 17 (Feb. 2016), p. 5792. DOI: 10.1120/jacmp.v17i1.5792.

[50] Michael Plotke. https://commons.wikimedia.org/w/index.php?curid=24288958. Own work, CC BY-SA 3.0.

[51] Renatokeshet. https://commons.wikimedia.org/w/index.php?curid=14661158. CC BY-SA 2.0.

[52] Renatokeshet. https://commons.wikimedia.org/w/index.php?curid=14661947. Public Domain.

[53] Simpsons. https://commons.wikimedia.org/w/index.php?curid=8904364. CC BY-SA 3.0.

[54] Simpsons. https://commons.wikimedia.org/w/index.php?curid=8904663. CC BY-SA 3.0.

Appendix A

Packages and Versions

This section describes tools and packages used in the application, broken down into a backend, where the server and technologies needed for image analysis are included, and a frontend, where the tools needed for display UI are included.

Frontend

Nodejs 20

- https://nodejs.org/en

Typescript 5.2.2

- https://www.typescriptlang.org/
- https://www.npmjs.com/package/typescript/v/5.2.2

React 18.2.0

- https://react.dev/
- https://www.npmjs.com/package/react/v/18.2.0

React Router 6.16.0

- https://reactrouter.com/en/main
- https://www.npmjs.com/package/react-router-dom/v/6.16.0

Tanstack React Query 4.35.3

- https://tanstack.com/query/v4
- https://www.npmjs.com/package/@tanstack/react-query/v/4.35.3

Axios 1.5.0

- https://github.com/axios/axios
- https://www.npmjs.com/package/axios/v/1.5.0

daisyUI 3.7.6

- https://v3.daisyui.com/
- https://www.npmjs.com/package/daisyui/v/3.7.6

tailwindcss 3.3.3

- https://tailwindcss.com/
- https://www.npmjs.com/package/tailwindcss/v/3.3.3

Backend

Python 3.11

- https://www.python.org/

Django 4.2.5

- https://www.djangoproject.com/
- https://pypi.org/project/Django/4.2.5/

Django REST framework 3.14.0

- https://www.django-rest-framework.org/
- https://pypi.org/project/djangorestframework/3.14.0/

Daphne 4.0.0

- https://github.com/django/daphne
- https://pypi.org/project/daphne/4.0.0/

Pylinac 3.23.2

- https://pylinac.readthedocs.io/en/release-v3.23
- https://pypi.org/project/pylinac/3.23.2/

Pydicom 2.4.4

- https://pydicom.github.io/pydicom/stable/
- https://pypi.org/project/pydicom/2.4.4/

OpenCV 4.10.0

- https://opencv.org/
- https://pypi.org/project/opencv-python/4.10.0.84/

NumPy 1.26.4

- https://numpy.org/
- https://pypi.org/project/numpy/1.26.4/