

2016•2017
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
*master in de industriële wetenschappen: nucleaire
technologie*

Masterproef

Automatization of mechanical quality assurance of a linac using MATLAB

Promotor :
Prof. dr. Brigitte RENIERS

Promotor :
ing. KENNY GEENS

Copromotor :
MSc. ALEXANDRA JANKELEVITCH

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

Jelle Smeulders

*Scriptie ingediend tot het behalen van de graad van master in de industriële
wetenschappen: nucleaire technologie*

2016•2017

Faculteit Industriële

ingenieurswetenschappen

*master in de industriële wetenschappen: nucleaire
technologie*

Masterproef

Automatization of mechanical quality assurance of a
linac using MATLAB

Promotor :
Prof. dr. Brigitte RENIERS

Promotor :
ing. KENNY GEENS

Copromotor :
MSc. ALEXANDRA JANKELEVITCH

Jelle Smeulders

*Scriptie ingediend tot het behalen van de graad van master in de industriële
wetenschappen: nucleaire technologie*

Acknowledgements

The idea to automatize the periodic mechanical QA analysis was introduced to me by Ir. Marc Orlandini and Ing. Kenny Geens of Limburgs Oncologisch Centrum (L.O.C.). Together with the other employees of L.O.C. they introduced me into the workflow of the radiotherapy department of Jessa hospital (Hasselt). They also guided me, and shared their knowledge with me in order to successfully accomplish my goal, writing this master's thesis.

Therefore, I would like to thank everyone who has helped me do this. Above all, I would like to thank my external and internal promotors, Ing. Kenny Geens and Dr. Brigitte Reniers, for the continuous support of my master's thesis, for their patience and the sharing of their knowledge and experience. Also a special thanks to my external co-promotor MSc. Alexandra Jankelevitch for the help with the tests and for the rectification of this dissertation.

Furthermore, I would like to thank my fellow students Michiel Darcis and Gert Leurs for pointing me in the direction of Ghostscript as the solution for the concatenation of individual PDF files. A final thanks goes to my family and friends for the support throughout the year.

Table of contents

Acknowledgements	1
List of tables	7
List of figures	9
Abbreviations	11
Abstract	13
Abstract in Dutch	15
Introduction	17
1 Radiotherapy: basic principles	19
1.1 Cancer treatment.....	19
1.2 Aim of radiotherapy treatment.....	19
1.3 Patient data acquisition – simulation.....	19
1.4 Treatment planning.....	20
2 Medical linear accelerators	21
2.1 LINAC composition.....	21
2.2 Photon beam production and treatment head.....	22
2.3 Lasers, range finder and field defining light.....	23
3 LINAC quality assurance	25
3.1 Need for quality assurance.....	25
3.2 Sources of errors.....	25
3.3 Goals of a QA programme.....	25
4 Mechanical QA protocol	27
4.1 Mechanical QA tools.....	27
4.1.1 Radiographic film.....	27
4.1.2 Electronic portal imaging devices.....	28
4.1.3 Iso-align device.....	28
4.2 Mechanical checks (monthly).....	29
4.2.1 Isocentre.....	30
4.2.2 Mechanical isocentre.....	30
4.2.3 Radiation isocentre.....	30
4.2.4 Coincidence light and radiation field.....	31
4.2.5 Gantry and collimator angle indicators.....	32
4.2.6 Jaw position indicators.....	32
4.2.7 Localizing lasers.....	32
4.2.8 Treatment couch position indicators.....	33
5 Image processing	35
5.1 DICOM images.....	35
5.2 Basics of digital image processing.....	36
5.2.1 What is a digital image?.....	36

5.2.2	Images in MATLAB and coordinate conventions.....	36
5.2.3	Image classes and image types in MATLAB.....	37
5.3	Intensity transformations and filtering in the spatial domain	38
5.3.1	Intensity transformation.....	38
5.3.2	Median filter.....	39
5.4	Morphological operations	39
5.4.1	Dilation and erosion.....	39
5.4.2	Structuring element characteristics and shapes	40
5.5	Image segmentation	41
5.5.1	Point and line detection.....	42
5.5.2	Edge detection and MATLAB implementation.....	42
6	The mechanical QA protocol of L.O.C.....	45
6.1	Distance between radiation- and mechanical isocentre	45
6.2	Deviation on table position indicators in three dimensions.....	46
6.3	Deviation on asymmetrical field sizes	46
7	Materials & Methods	48
7.1	Distance between radiation- and mechanical isocentre	48
7.1.1	Pre-processing radiation isocentre detection.....	49
7.1.2	Field edge algorithm.....	50
7.1.3	Computing radiation isocentre	52
7.1.4	Pre-processing mechanical isocentre detection.....	52
7.1.5	Circle detection procedure.....	54
7.1.6	Computing the distance between both isocentres	55
7.1.7	Alternative algorithm for detection of the radiation isocentre.....	56
7.2	Deviation on the table position indicators in three dimensions	57
7.2.1	Image fusing and pre-processing.....	57
7.2.2	Circle detection procedure and distance computation.....	59
7.3	Deviation on asymmetrical field sizes	60
7.3.1	Computation of asymmetrical field sizes and deviations	60
7.4	Architecture of the GUI	61
7.4.1	Open	62
7.4.2	Perform analysis and output file	62
7.5	Accuracy and reproducibility.....	63
7.5.1	Accuracy.....	63
7.5.2	Reproducibility.....	63
8	Results and discussion.....	65
8.1	Algorithm timing considerations	65
8.2	Accuracy	65
8.2.1	Clinac	65
8.2.2	Truebeam.....	68

8.3	Reproducibility.....	70
8.3.1	Clinac	70
8.3.2	Truebeam.....	71
9	Conclusions.....	73
10	References	75
11	Appendices	77
11.1	Appendix A: periodically mechanical QA protocol of L.O.C.....	77
11.2	Appendix B: source code.....	82
11.3	Appendix C: Example of output file.....	136

List of tables

Table 4.1: Relationship between the likelihood of an undetected malfunction, the gravity of the malfunction for the patient or personnel and the minimum test frequency [11: p.4].....	27
Table 4.2: Most common mechanical checks that are recommended to be performed monthly [9-11], [16].....	29
Table 6.1: The 13 different setups (of L.O.C.) for the determination of the distance between radiation- and mechanical isocentre.....	45
Table 8.1: Results of testing the accuracy of the circle detection algorithm using a known displacement of the treatment table of the CLINAC. Note that the treatment table position indicators have an uncertainty of ± 2 mm.	65
Table 8.2: The mean deviation, SD and maximum deviation in reproducibility in performing the circle detection algorithm using a CLINAC linear accelerator (N=8).	70
Table 8.3: The mean deviation, SD and maximum deviation in reproducibility in computing the radiation isocentre using a CLINAC linear accelerator (N=8).	70
Table 8.4: The mean deviation, SD and maximum deviation in reproducibility in performing the circle detection algorithm using a Truebeam linear accelerator (N=8).....	71
Table 8.5: The mean deviation, SD and maximum deviation in reproducibility in computing the radiation isocentre using a Truebeam linear accelerator (N=8).	71

List of figures

Figure 1.1: Treatment plan for two bilateral arcs of 120° each [2: p150].....	20
Figure 1.2: Treatment plan of a 4-field box [2: p.145].....	20
Figure 2.1: Accelerating waveguide is in the gantry parallel to the isocentre axis; electrons are brought to the movable target through a beam transport system; the rf-power generator is located in the gantry stand; machine can produce megavoltage x rays as well as electrons. [4: p. 636].....	21
Figure 2.2: A crossbeam profile of a flattening filtered 10MV photon beam (dashed line) compared to the crossbeam profile of an unflattened 10MV photon beam (solid line). The unflattened beam has approximately four times higher dose rate at central axis [6: p.65].....	22
Figure 2.3: A multileaf collimator shaping an irregular beam shape [7: p.42].	23
Figure 2.4: Schematic survey of a treatment head of a LINAC [8: p.896-902].....	23
Figure 4.1: Cross-sectional view of an amorphous silicon EPID. Only one pixel is shown [14].....	28
Figure 4.2: An iso-align device with 15×15 ; 10×10 and 5×5 cm^2 markers. [16].....	29
Figure 4.3: Graphical representation of asymmetrical field size measurements.....	32
Figure 5.1: A representation of the pixel index coordinate system (left) and the spatial coordinate system (right) [22].	36
Figure 5.2: A 3×3 neighbourhood centred at (x, y) in an image $f(x, y)$ [26].....	38
Figure 5.3: The different transfer curves that are available in function imadjust [27].....	39
Figure 5.4: The original image (left) and the output image after performing a dilation with a 3×3 square structure element [28].	40
Figure 5.5: The original image (left) and the output image after performing an erosion with a 3×3 square structure element [28].	40
Figure 5.6: The different coordinate systems in filter techniques. w denotes the filter (or mask) coordinates and $f(x, y)$ denotes coordinates of the underlying original image [29].	41
Figure 5.7: A 3×3 neighbourhood and the indices used to specify locations in the neighbourhood [30].....	44
Figure 6.1: Result of miscalibrated jaw on the place and size of the radiation isocentre. Note how these points can be averaged out in order to calculate asymmetrical field size parameters.	47
Figure 7.1: Architecture of the algorithm for computing the distance between the radiation- and mechanical isocenter.	48
Figure 7.2: Left: The histogram of the input image. Right: The histogram of the input image after contrast stretching. The x-axis denotes the 65536 shades of grey and the y-axis visualizes the amount of pixels with a certain intensity value.....	49
Figure 7.3: On the left an input image is shown where the blue line defines the range over which the spectrum is computed. On the right the resulting crossbeam profile is shown.	50
Figure 7.4: Visualization of the edge scanning procedure, input image with ranges over which crossbeam profiles are computed.	51
Figure 7.5: The result of performing the field edge algorithm in combination with the radiation isocentre computation.....	52
Figure 7.6: On the left the input image is shown and the right image shows the result of the edge detection operation.....	53
Figure 7.7: On the left the result of the closing operation is shown. The image at the right shows the result of the erosion and dilation operation.	54
Figure 7.8: The left image shows the result of the function regionprops and the right image shows the result of the iterative process for selecting the desired circle.....	55
Figure 7.9: The image on the left shows an example of an output file visualizing both the radiation- and mechanical isocentre. The image on the right shows a zoomed view of this output image.....	55
Figure 7.10: The architecture of the discarded alternative algorithm for the computation of the distance between the radiation- and mechanical isocentre. Only the left branch (radiation isocentre) differs from the architecture of Figure 7.1	56
Figure 7.11: Architecture of the algorithm for computing the deviation on the table position indicators in three dimensions.....	57

Figure 7.12: On the left the reference image is shown, the right image shows the situation after applying the longitudinal displacement.	57
Figure 7.13: Both images of Figure 7.12 fused into one image.....	58
Figure 7.14: The result of performing the whole pre-processing step on the fused image. The yellow arrows point to the radio-opaque spheres that are chosen to be detected.....	58
Figure 7.15: Left shows the result of performing the algorithm for computing the deviation on the table position indicators for the longitudinal direction. The image on the right is an detail image of one of the detected circles.....	59
Figure 7.16: Architecture of the algorithm for computing deviations on asymmetrical field sizes.....	60
Figure 7.17: The resulting output image after performing the asymmetrical field size algorithm.....	60
Figure 7.18: Screenshot of the graphical user-interface.....	61
Figure 7.19: Architecture of the main file of the graphical user-interface.....	61
Figure 8.1: Comparison of the results of measuring the distance between the radiation- and mechanical isocentre manually vs. with the software for a CLINAC. The image indices are the same as those in Table 6.1.....	66
Figure 8.2: Comparison of the results of measuring the deviation on the table position indicators manually vs. with the software for a CLINAC.....	66
Figure 8.3: Comparison of the results of measuring the deviation on asymmetrical field size parameters manually vs. with the software for a CLINAC. The first bar for each parameter denoted the results for the 5x5 cm ² field. The second and third bar of each parameter denote the results for the 10x10 and 18x18 cm ² fields respectively.....	67
Figure 8.4: Comparison of the results of measuring the distance between the radiation- and mechanical isocentre manually vs. with the software for the Truebeam.....	68
Figure 8.5: Comparison of the results of measuring the deviation on the table position indicators manually vs. with the software for a Truebeam.....	68
Figure 8.6: Comparison of the results of measuring the deviation on asymmetrical field size parameters manually vs. with the software for a Truebeam. The first bar for each parameter denoted the results for the 5 x 5 cm ² field. The second and third bar of each parameter denote the results for the 10 x 10 and 18 x 18 cm ² fields respectively.....	69

Abbreviations

CT	Computed tomography
CTV	Clinical target volume
DICOM	Digital imaging and communications in medicine
EPID	Electronic portal imaging device
GTV	Gross tumour volume
ICRU	International commission on radiation units & measurements
IGRT	Image guided radiotherapy
ITV	Internal treatment volume
LINAC	Linear accelerator
L.O.C.	Limburg Oncologisch Centrum
MLC	Multileaf collimator
MRI	Magnetic resonance imaging
NCS	Nederlandse commissie voor stralingsdosimetrie
OAR	Organs at risk
PDF	Portable document format
POI	Point of interest
PR	Pixel resolution
PSI	Pixel size at isocentre
PTV	Planned treatment volume
QA	Quality assurance
RF	Radio-frequency
ROI	Region of interest
SAD	Source to axis distance
SID	Source to imager distance
SSD	Source to surface distance
TPS	Treatment planning system

Abstract

During the periodical mechanical quality assurance (QA) checks, the radiotherapy department of Limburgs Oncologisch Centrum (L.O.C.) has to measure certain parameters manually (e.g. distances). Most parameters are defined using Megavolt EPID images of a mechanical positioned radio-opaque sphere.

This visual determination could introduce various errors and inconsistencies. Therefore this master's thesis focuses on the automatization of these determinations by means of the development of an image processing application using MATLAB.

The image processing algorithms have been developed and tested using MATLAB and specifically using the Image Processing Toolbox. These algorithms have been embedded in a user-friendly user interface which generates one single PDF as output file. This PDF file encloses a summary of the results of the QA analysis, these include:

- distances between radiation and mechanical isocentre of multiple fields;
- deviation on the mechanical movement of the table in three dimensions;
- deviation on the asymmetrical field sizes for three different fields.

The developed algorithms have proven to be ten times faster with respect to manual measurements. Furthermore, these algorithms yield results which averagely deviate 0,273 mm from those acquired manually. This was tested using a Varian CLINAC and Truebeam from which the first one was equipped with an EPID of higher resolution. It was observed that the difference in resolution resulted in the same reproducibility and accuracy.

Abstract in Dutch

Tijdens de periodische mechanische kwaliteitscontrole (QA) moet het personeel van de afdeling radiotherapie van het L.O.C. bepaalde parameters (vb. afstanden) visueel afleiden. De meeste van deze parameters worden afgeleid uit Megavolt EPID beelden van een mechanisch geïntegreerde metalen bol. Deze methode heeft als gevolg dat er mogelijk fouten kunnen geïntroduceerd worden en de metingen niet consistent zijn. Daarom wordt er in deze masterthesis gefocust op het automatiseren van deze metingen door middel van een beeldverwerkingssoftware ontwikkeld met behulp van MATLAB.

De ontwikkelde algoritmes zijn samengebracht in een userinterface dewelke één enkele PDF file wegschrijft als output. Deze PDF file omvat een samenvatting van de resultaten van de QA analyse en bevat meer bepaald:

- afstanden tussen stralings- en mechanisch isocentrum van meerdere velden;
- afwijkingen op de mechanische beweging van de tafel in drie dimensies;
- afwijkingen op de asymmetrische veldgroottes.

De software heeft ongeveer tien keer minder tijd nodig om een analyse te voltooien ten opzichte van de manuele bepaling. Bovendien wijken de bekomen resultaten niet meer dan 0,273 mm af van de manueel bepaalde resultaten. Dit werd getest gebruikmakend van een Varian CLINAC en Truebeam waarvan de eerst genoemde EPID foto's leverde van hogere resolutie. Er kon worden vastgesteld dat dit verschil in resolutie geen aanleiding gaf tot significant verschillende resultaten voor de reproduceerbaarheid en accuraatheid van de software.

Introduction

The research is completed in association with the radiotherapy department of Limburgs Oncologisch Centrum located in Jessa hospital Hasselt. This department uses linear accelerators (LINAC) to treat malignant tumours. It's obvious, when working with patients, that the quality of the treatment should be checked on well determined frequencies by means of quality assurance (QA) tests.

During the periodical mechanical QA checks, certain parameters have to be visually determined from EPID images. Most of these parameters are defined by radio-opaque spheres located in a specialized phantom. This visual determination could introduce various errors and/or inconsistencies. Therefore it may be a good idea to automate this analysis by means of an image processing software. The automatization of the manual QA procedure aims on the improvement of consistency and accuracy. In addition, it could also save a significant amount of time since a manual measurement takes about 20 to 30 minutes.

The main objective is to develop image processing software which analyses the images of the mechanical QA tests. The software must be able to extract the following parameters out of the tests:

- distance between the radiation- and mechanical isocentre;
- deviation on the table position indicators in three dimensions;
- deviation on asymmetrical field size parameters.

The image processing software was developed using MATLAB. This program offers a wide array of opportunities as it gives the user the chance to easily build a graphical user interface. Moreover, the image processing toolbox allows the user to perform image processing.

The first two sections of this dissertation give a brief introduction to the workflow of a radiotherapy department and the (mechanical) components of a linear accelerator. The next three sections discuss the need for quality assurance in radiotherapy and how it is implemented. Subsequently, chapter 6 outlines the mechanical QA protocol of L.O.C. Furthermore, the structures of the different algorithms are explained and their performance regarding accuracy and reproducibility is evaluated.

1 Radiotherapy: basic principles

1.1 Cancer treatment

Three major modalities are used for the treatment of cancer, these are: surgery, radiotherapy and chemotherapy. Choosing a suitable treatment method should take into account the location, type and size of the tumour within the patient. In some cases a combination of different modalities may also be used [1].

1.2 Aim of radiotherapy treatment

Radiotherapy refers to a technique which uses a beam of radiation to effectively kill tumour cells and can be delivered by either an external radiation beam or an internal source (Brachytherapy). The latter one will not be dealt with and therefore the word radiotherapy, in this dissertation, refers to external beam radiotherapy. There are three main kinds of radiotherapy treatment: curative, adjuvant and palliative treatment [1].

During curative treatment the target volume, which contains the tumour, is exposed to a certain radiation dose. The radiation dose during curative treatment often approaches the normal tissue tolerance and is restricted by the dose limitations of the organs at risk. On the other hand, adjuvant treatment is less likely to make use of radiation doses exceeding the normal tissue tolerance. Finally palliative treatment should be performed with regard to relieving pain or avoiding symptomatic injury to healthy tissues [1].

1.3 Patient data acquisition – simulation

In order to treat the tumour accurately one needs certain data concerning the patient to make sure the patient can be properly positioned and the tumour volume is sufficiently irradiated. For this reason, the first step in external beam treatment is simulation.

The role of simulation in the treatment process consists of: the determination of patient treatment position and tumour location and the acquisition of patient data for treatment planning. Current simulation systems are based on computed tomography (CT) which provide anatomical information in the form of transverse slices with high resolution and contrast. The electron densities can also be extracted from these slices which are needed for treatment planning [2].

Image modalities such as magnetic resonance imaging (MRI) provide a better soft tissue contrast in areas such as the brain which allows the operator to detect small lesions easier. However, MRI cannot be used for simulation for several reasons. First, no electron density information is available which is needed for dose calculations in the treatment planning system. Further, MRI is prone to geometrical artefacts and noise which may influence the accuracy of the treatment. Therefore CT-MR images are often fused to combine the accurate volume definition from MR with the electron density information from CT [2].

1.4 Treatment planning

After simulation the retrieved data is transferred to the treatment planning system (TPS). The treatment plan is extremely important because it will determine the execution and it therefore has an influence on the quality of the treatment. A successful treatment plan requires a certain minimum dose within the target volume and a dose as low as possible in healthy tissues around the tumour. To achieve this goal often rotational techniques are used as well as combinations of static beams [2].

In treatment planning, the volume that has to be treated is defined by means of the so called ICRU-volumes. The gross treatment volume (GTV) corresponds to the gross tangible extent and location of malignant disease. The clinical treatment volume (CTV) encloses the GTV and the areas surrounding this GTV which may contain microscopic disease or that are considered to be at risk and require treatment. The internal treatment volume (ITV) encloses the CTV plus an internal margin which is chosen to take in consideration the variations in size and position of the CTV. These variation can be due to organ motion such as breathing motion, bladder filling, etc.. Finally, the planned treatment volume (PTV) is described to choose proper beam arrangements taking into account possible geometrical variations. The inclusion of these geometric uncertainties, which consist of set-up variations, machine inaccuracies, etc., has to ensure that the prescribed dose is actually delivered to the CTV. Usually one PTV is used to encompass one or several CTV's which has to be targeted by one or several fields. Some organs at risk (OAR) might surround the PTV. An OAR is an organ which is sensitive to radiation. The dose from the treatment plan must be compared to its restriction values [2].

Sparing these OAR's while respecting the minimum dose requirement in the PTV requires the examination of different beam arrangements. Single fields are often used for palliative treatments or superficial lesions. Deeper lesions are mostly treated with a combination of two or more beams to concentrate the dose in the PTV. In Figure 1.2 an example is given of a 4-field box which allows a very high dose to be delivered at the intersection of the beams. Today, rotational techniques are also popular, Figure 1.1 shows the isodose curves for two bilateral arcs of 120° each. Note that the isodose curves become tighter along the angles avoided by the arcs, which allows more accurate dose delivery in the PTV and sparing of the OARs [2].

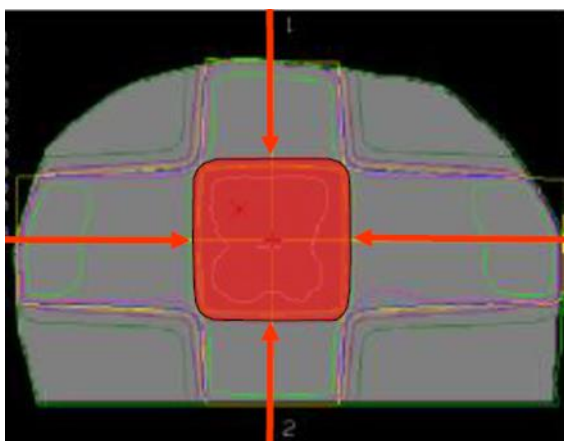


Figure 1.2: Treatment plan of a 4-field box [2: p.145].

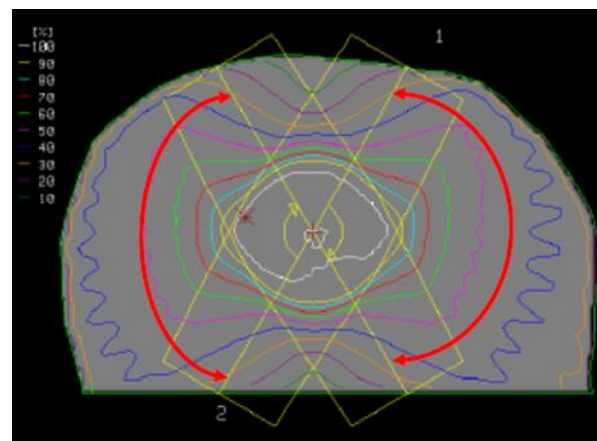


Figure 1.1: Treatment plan for two bilateral arcs of 120° each [2: p150].

2 Medical linear accelerators

Medical linear accelerators (often called LINACs) use radio-frequency (RF) electromagnetic waves to accelerate electrons to kinetic energies from 4 to 25 MeV. The electrons follow straight trajectories through vacuum constructions called accelerating waveguides. Typical modern LINACs can provide several electron energies (e.g. in the range 6 - 22 MeV) and several photon energies (typical 6, 10 and 15 MV) [3]. Linear accelerators are the most common devices to treat tumours with external beam radiation. The final photon beam is shaped as it exits the LINACs gantry and is aimed towards the tumour of the patient. The gantry can be fully rotated around the patient to irradiate the target volume from many different angles. It's clear that the patient should be properly positioned before the start of the treatment. The positioning happens with the help of lasers, image guided radiotherapy treatment (IGRT) tools, and a moveable treatment couch, which can move in three to six different dimensions.

2.1 LINAC composition

The basic composition of a LINAC is given in Figure 2.1. The RF-power generator is situated in the stand of the LINAC. The gantry is isocentrically mounted to the stand and contains an electron gun, accelerating waveguide and several beam transport systems. The electron gun thermionically emits electrons which are accelerated in the accelerating waveguide [3]. As seen in Figure 2.1, the accelerating waveguide is positioned perpendicular to the direction of the final beam. Bending magnets make sure that the electron beam strikes the X-ray target by bending the electron beam 270 degrees. Finally, the intersection of the couch axis and the gantry axis defines the isocentre [4].

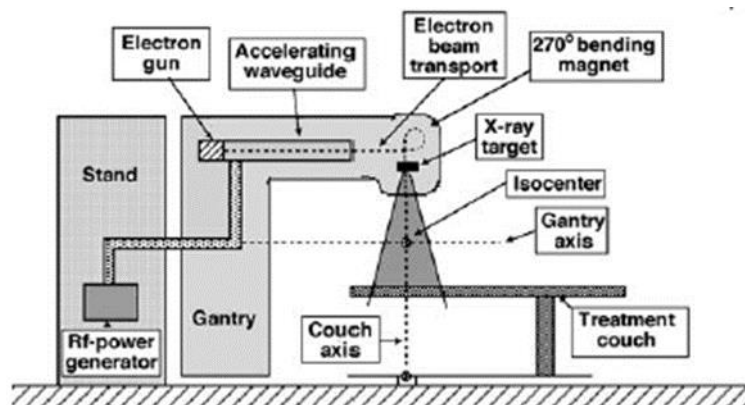


Figure 2.1: Accelerating waveguide is in the gantry parallel to the isocentre axis; electrons are brought to the movable target through a beam transport system; the rf-power generator is located in the gantry stand; machine can produce megavoltage x rays as well as electrons. [4: p. 636]

2.2 Photon beam production and treatment head

When a beam of electrons strikes the X-ray target coulomb interactions with the target nuclei will transform a small portion (in the order of 10%) of the electrons' kinetic energy into bremsstrahlung X rays. The intensity spectrum of these bremsstrahlung X-rays is strongly forward peaked [4]. Therefore flattening filters are used to shape up the spectrum [3]. An example of a flattening filtered and a flattening filter free cross beam profile is given in Figure 2.2. For more information about the composition and use of X-ray targets and flattening filters the reader is referred to [3: p.125-127], [5] and [3: chapter 5] respectively.

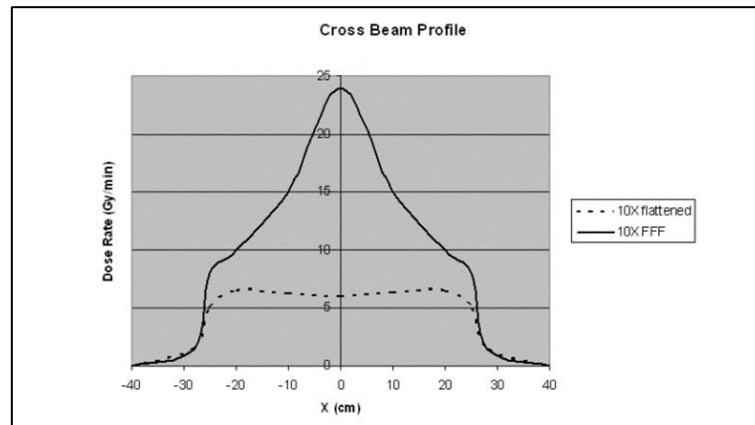


Figure 2.2: A crossbeam profile of a flattening filtered 10MV photon beam (dashed line) compared to the crossbeam profile of an unflattened 10MV photon beam (solid line). The unflattened beam has approximately four times higher dose rate at central axis [6: p.65].

The collimation of the photon beam in modern LINACs is provided by three components: the primary collimator, the adjustable secondary collimator, and the multileaf collimator (MLC). First, a circular (maximum) field is produced by a fixed primary collimator. The secondary collimator exists of two lower and two upper independently adjustable jaws which can produce rectangular or square fields with dimensions up to 40 x 40 cm² at the isocentre [4]. The MLC is the final collimation system which a beam passes before reaching the patient. The use of MLC allows the production of irregularly shaped radiation fields. It is based on two arrays of narrow collimator leaves, each leaf equipped with its own computer-controlled motor and control circuit. Current models incorporate usually 120 leaves (60 pairs) which can cover radiation fields up to 40 x 40 cm² [4]. Each leaf is made of a tungsten alloy and has a width of 1 cm or less (projected at the isocentre) and a thickness of 6 – 7,5 cm. The interleaf X-ray transmission is typically below 3% [7]. A schematic survey of the treatment head of a LINAC is given in Figure 2.4 and an example of a MLC is given in Figure 2.3.

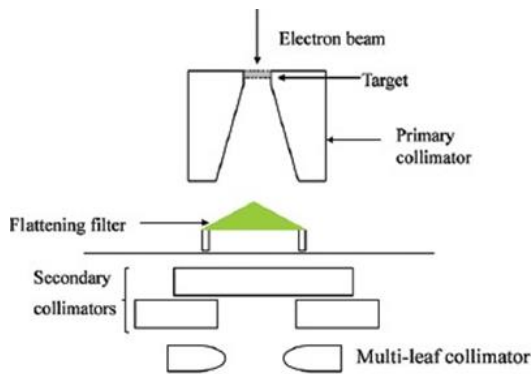


Figure 2.4: Schematic survey of a treatment head of a LINAC [8: p.896-902].

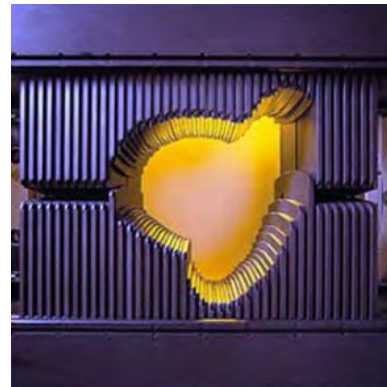


Figure 2.3: A multileaf collimator shaping an irregular beam shape [7: p.42].

2.3 Lasers, range finder and field defining light

The lasers, range finder and field defining light provide visual methods for correctly positioning the patient, which is very important for modern radiotherapy to ensure the accuracy of dose deposition. Laser positioning devices are used as an indication of the position of the machines isocentre in the treatment bunker. The range finder is used to place the patient at the correct distance from the treatment head by projecting a (centimetre) scale on the patient's skin which indicates the vertical distance from the isocentre. Finally, the field light indicates the area that coincides with the radiation field, it's a visual representation of the radiation field [4].

3 LINAC quality assurance

3.1 Need for quality assurance

The dose deposition in radiotherapy should be as accurate as possible. Quality assurance (QA) can improve the results of therapy because it focusses on minimizing the errors in treatment planning and dose delivery. A high degree of accuracy and consistency should be reached to fully exploit the equipment, QA makes this possible. Finally, QA allows the comparison of results among different radiotherapy centres which may be meaningful [1].

3.2 Sources of errors

The uncertainty in the delivered dose is due to several different types of errors that may occur. These errors can occur at different stages in the treatment process. Errors during the determination of the patients anatomy may be due to patient positioning or defining OARs and target volumes. Also during treatment planning several errors could occur like deviations in beam data or anomalies in computer software and hardware. These aberrations may be the result of mistakes, inattention or misunderstanding [1]. Finally, the performance of the delivery equipment may introduce several errors. The functional performance of the LINAC can change suddenly due to phenomena as component failure, malfunction of electronic circuits or mechanical breakdown. Also slow changes in performance can appear due to fatigue and aging of the components [9]. The above summary of possible errors shows the complexity of QA in radiotherapy and points out that a QA programme is essential to assure the quality of radiotherapy treatment [1].

3.3 Goals of a QA programme

“Every patient with cancer deserves to receive the best possible management to achieve cure, long-term tumour control or palliation, this is the major goal of cancer management (ISCRO, 1986)” [9]. “Quality” of treatment is defined as the totality of features of the treatment that bears on its ability to achieve the stated goal of patient care. On the other hand, “quality assurance” is all the planned interventions that are necessary to achieve sufficient confidence that treatment will satisfy the stated achievements for quality care [9].

The report of task group 40 (TG-40) recommends that the dose delivered to the patient be within $\pm 5\%$ of the prescribed dose. Taking into account the amount of steps in the treatment process, each step should be performed with an accuracy better than 5% to achieve this goal. Therefore the main goal of a QA program is to assure that the machine characteristics do not deviate significantly from their baseline values (those acquired at time of acceptance testing) [10].

4 Mechanical QA protocol

Quality assurance checks of a LINAC can be subdivided in dosimetry checks and mechanical checks. According to NCS' report 9 [11] and AAPM's task group 142 report [10] the minimum frequency of these checks depends on the likelihood for a malfunction to occur, the chance the malfunction will not be noticed during treatment and the likelihood and severity of the effects of these unnoticed malfunctions [10]. These criteria are summarized in Table 4.1 together with the three test frequencies: daily, monthly and annual. This dissertation only covers the monthly mechanical QA procedures of the LINAC. For a complete overview of QA procedures we refer the reader to the reports of TG-40 [9]. and the updated version of TG-142 [10].

Table 4.1: Relationship between the likelihood of an undetected malfunction, the gravity of the malfunction for the patient or personnel and the minimum test frequency [11: p.4].

	low likelihood of undetected malfunction	high likelihood of undetected malfunction
no direct harmful effects	annual	monthly
possible harmful effects	monthly	daily

4.1 Mechanical QA tools

Before describing specific mechanical QA test procedures an introduction of several QA tools that ease the measurements is necessary. This paragraph will briefly define these QA tools and outline their contribution to the QA procedure.

4.1.1 Radiographic film

Radiographic films are used to perform portal imaging (discussed below) and film dosimetry. Film dosimetry is a technique used to obtain the distribution of absorbed dose delivered by an external radiation source. When radiographic films are exposed to radiation they discolour and the amount of discoloration can be linked to the absorbed dose to the film. After irradiation the dose distribution can be checked using a flatbed scanner and an appropriate software packet for analysis. In other words, radiographic films represent a graphical presentation of the amount of radiation delivered to the film.

Today GAFChromic EBT-3 films are frequently used, which have been designed to address the needs of medical physicists and dosimetrist working in a radiotherapy environment. This type of film is popular due to its excellent specifications including its wide dose range (0.1 Gy – 20 Gy), near tissue-equivalency, water resistance and energy-independence [12].

4.1.2 Electronic portal imaging devices

“Portal imaging is the acquisition of images with a radiotherapy beam” [13: p.789]. There are several kinds of portal imaging media such as radiographic films and electronic portal imaging devices (EPID). EPID has many advantages over radiographic film. The obtained images are digital and therefore directly available and they can be used to adjust patient or beam position during treatment delivery. The digital character of the EPID images also allows easier image processing and contrast enhancement. The major disadvantage of EPIDs relative to radiographic film is that they provide images with disappointing image quality. During the history of EPID generations this is changing due to the introduction of new technologies, such as amorphous silicon-based devices, but it still remains an issue [13].

There are several different EPID systems commercially available such as camera-based systems, Liquid ion chamber array systems and amorphous silicon system. This paragraph will briefly discuss the construction of amorphous silicon systems. For a complete discussion of EPID systems the reader is referred to the review article “Portal imaging” of K.A. Langmack [13]. The amorphous silicon EPID system with the most development uses a front metal sheet, usually 1 mm of copper, with a gadolinium oxysulphide phosphor to convert the X rays to visible light. This light is detected by an array of hydrogenated amorphous silicon photodiodes and thin film transistors. The photodiodes are read electronically and form the pixels of the acquired image [13]. The basic composition is given in Figure 4.1.

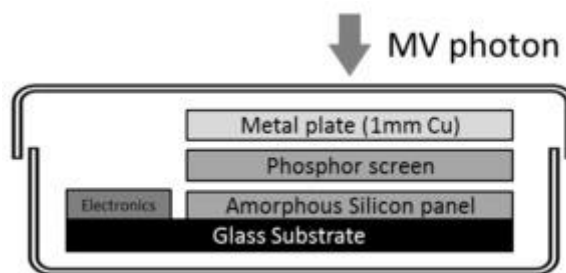


Figure 4.1: Cross-sectional view of an amorphous silicon EPID. Only one pixel is shown [14].

4.1.3 Iso-align device

An iso-align device, shown in Figure 4.2, is a multi-functional and precision device for quality assurance. It can be used for checking many mechanical parameters including the alignment of lasers, light field and radiation field coincidence and treatment table position indicators. The plane of the iso-align is inscribed with lines which define several field sizes. The plane can be rotated (usually with increments of 45°) and inside there are tungsten or lead balls embedded which can easily be distinguished on film or EPID images. Radiochromic films can be inserted in between the two plates [15].

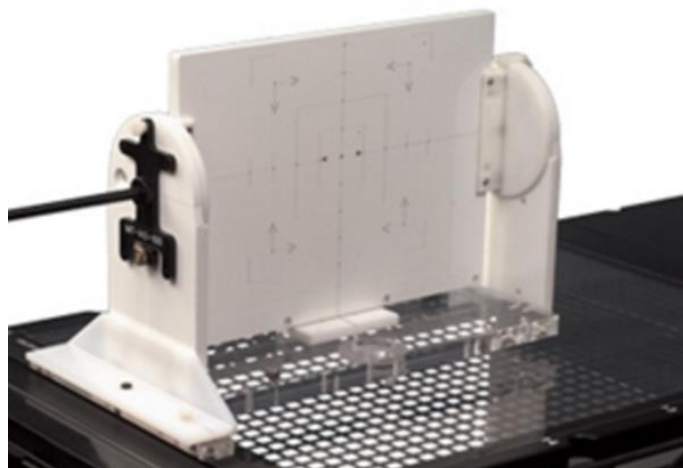


Figure 4.2: An iso-align device with 15 x 15; 10 x 10 and 5 x 5 cm² markers. [16]

4.2 Mechanical checks (monthly)

The verification of mechanical parameters of a LINAC is performed to assure an accurate treatment delivery as well as to obtain information about changes in time due to wear of mechanical components [11]. The mechanical checks that are recommended to be performed monthly are listed in Table 4.2. This table is an assembly of the most imported recommendations of NCS report 8 [17] and 9 [11] and AAPM's task group 40 [9]. and 142 [10] reports. The tightness of the tolerance values depends on the type of treatment delivery. For example when performing QA for stereotactic radiosurgery (SRS), which is delivered by a single high dose fraction, the tolerance values will be stricter and certain tests may be different in comparison to a QA procedure for fractionated treatment [10]. For simplicity Table 4.2 only covers the tolerance values for treatment units which are used for fractionated treatment.

Table 4.2: Most common mechanical checks that are recommended to be performed monthly [9-11], [17]

mechanical check	tolerance value
crosshair centring (walk out)	1 mm
deviation mechanical isocentre	1 mm
light/radiation field coincidence	2 mm or 1 % on a side
light/radiation field coincidence (asymmetric)	1 mm or 1 % on a side
Gantry/collimator angle indicators	1.0°
Jaw position indicators (symmetric)	2 mm
Jaw position indicators (asymmetric)	1 mm
Localizing lasers	± 1 mm
Verification of treatment couch axes	2 mm/1°
Treatment couch position indicators	2 mm/1°

There are two different ways of interpreting a tolerance value. In some papers in literature the tolerance values are guidelines for acceptable deviations while in others actions are required when the tolerance value is exceeded [11]. In what follows all the stated tolerance values must be regarded as guidelines for acceptable deviations, deviations which require immediate action will be represented by action levels (which by exceeding an appropriate action is necessary). Note that when several consecutive measurements are close to the tolerance value also a corrective action may be required.

The last decades new technologies were introduced in radiotherapy treatment such as image guided radiotherapy (IGRT). IGRT basically is the use of imaging during radiotherapy treatment for the purpose of improving precision and accuracy. The LINAC used to deliver treatment is equipped with imaging modules to allow one to image the tumour directly before or during treatment. These images are then automatically compared to the reference images made during simulation. The necessary adjustments concerning positioning are then automatically made based on this comparison [18].

4.2.1 Isocentre

The isocentre of a linac is a very important mechanical parameter in radiotherapy treatment delivery and QA. The isocentre is defined as “the centre of the smallest sphere through which the axes of the radiation beams pass in all conditions” [10: p.19]. The concept isocentre can be distinguished in two terms: the mechanical isocentre and the radiation isocentre, which are described below.

4.2.2 Mechanical isocentre

The mechanical isocentre is defined as being the point of intersection of the gantry rotation axis, the collimator rotation axis and the table rotation axis [10]. Due to a difference in bending of the radiation head at different gantry angles this point cannot be determined unambiguously and therefore one should define a sphere that envelops the isocentre [11].

4.2.3 Radiation isocentre

The radiation isocentre is defined as the point of intersection of the radiation beam axes at different gantry, collimator and table angles [10]. One can determine the location of the radiation isocentre by using the Winston–Lutz method [19].

The Winston-Lutz method can be performed by placing a small radio-opaque sphere (e.g. 6 mm diameter) at the mechanical isocentre of the linac. This sphere is irradiated with a small collimated beam at each available energy (e.g. 6 and 15 MV) and at different gantry, collimator and table angles, additionally EPID images are acquired for each setup. The distance between the centre of the radiation beam and the centre of the sphere defines the shift between the radiation and mechanical isocentre. This distance is measured for each image acquired by the EPID and shifts should be less than 1 mm [19], [20].

Special attention should be given to the field defining jaws while performing the Winston-Lutz test. One should check if the jaws open symmetrically because when this is not the case it will directly influence the size of the radiation isocentre sphere which could lead to a misinterpretation of the results [19].

4.2.4 Coincidence light and radiation field

The alignment of the radiation field can be established by the light field. It is therefore important to check their coincidence. The collimator jaws of a linac rotate on a circular bearing which is attached to the gantry. The axis around which the collimator rotates is defined as the collimator axis of rotation (AOR) and is given by the intersection of the crosshairs in the light field. Before determining the coincidence between the light and radiation field one should raise the table to isocentric height while the gantry angle is 0° (gantry is oriented with the collimator AOR directed vertically downward). Now the iso-align device is placed on the treatment table (positioned plane parallel to the table) and it should be oriented in such a way that the collimator AOR (crosshair) coincides with the centre of the iso-align device. The projected image of the crosshair should not walkout more than 1mm from this point as the collimator is rotated through its full range. Note that this check also can be performed with for example graph paper [3], [10].

Now coincidence of light and radiation field can be checked by setting up the iso-align device in such a way that the marks (e.g. the $15 \times 15 \text{ cm}^2$ field size indicators) coincide with the edges of the light field. After irradiation at SSD 100cm and EPID imaging the dimensions of the light field can be measured and the deviations in both directions can be computed. Note that this check alternatively can be performed with a radiographic film by punching holes through the film in the corners of the light field. Several plastic slabs are placed on top of the film to position the film near to the depth of dose maximum [3]. Both dimensions of the light field should not deviate more than 2mm, as denoted by Table 4.2 [10].

Like denoted in Table 4.2 the coincidence of light and radiation field should also be checked asymmetrically. The test procedure is fully analogous to the symmetrically determination (described above) [10]. To determine the asymmetric deviations the radiation field size is measured by measuring the distance between the isocentre and the field edge like demonstrated in Figure 4.3. The sum of distances x_1 and x_2 gives the total field size in the x-direction, the same holds for the y-direction. It should be clear one can maintain the symmetrical deviations by summing the asymmetrical deviations and therefore only one (asymmetric) measurement is required. The asymmetric deviation should not be greater than 1 mm at each side of the isocentre. Any symmetric deviation greater than 2 mm (symmetric) or 1 mm (asymmetric) may point out that the collimator AOR is not aligned with the central axis of the radiation field or that the jaws are incorrectly calibrated [3].

It is important to notice that the coincidence of radiation and light field is becoming less important due to the arrival of IGRT. Nowadays the patient is initially positioned using the lasers and afterwards this position is adjusted automatically using image guidance. On the other hand it is still recommended to check several configured field sizes asymmetrically as given in Figure 4.3.

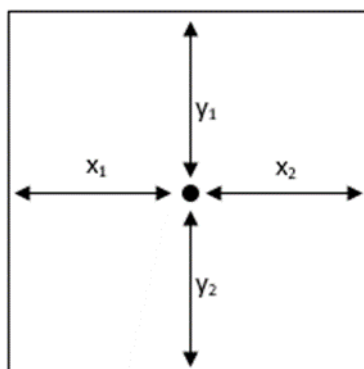


Figure 4.3: Graphical representation of asymmetrical field size measurements..

4.2.5 Gantry and collimator angle indicators

The accuracy of the digital gantry and collimator angle indicators can be checked by using a spirit level [3]. At each cardinal angle (0, 90, 180 and 270°) the value indicated by the spirit level should not deviate more than 1.0° [10]. Today many digital spirit level can measure any angle between 0 and 360 degrees, with an accuracy of 10⁻¹ or better, and therefore one may consider checking several angles between the cardinal angles (e.g. 165°) [3].

4.2.6 Jaw position indicators

The jaw position indicator accuracy (symmetrically and asymmetrically) can be checked by performing the split field test or by measuring the asymmetric field size parameters of Figure 4.3. Considering the split field test, gaps or overlaps up to 2 mm (symmetric) or 1 mm (asymmetric) are usually tolerated. The split field test will not be discussed in detail because it will be of little interest in this dissertation. For more information the reader is referred to the work of Klein et al. [21] and a brief explanation of the split field test is also given by the medical physicists of the KFJ hospital in Vienna [22].

4.2.7 Localizing lasers

The accuracy check of the localizing lasers can be divided into two tests. First one should make sure that the laser beams perfectly describe horizontal and vertical planes. This can be done by comparing the projection of the beams with certain reference points on the floor and walls. Deviations, with respect to this reference point, up to 2 mm are usually tolerated for this test. Secondly, the intersection of the laser beams should coincide with the isocentre and therefore one should check the distance between this point of intersection and the isocentre [11]. This deviation should not be greater than 1 mm [10].

4.2.8 Treatment couch position indicators

The accuracy of the couch position indicators should be checked for the lateral, longitudinal and vertical motion as well as for the rotational motion. The vertical position indicator accuracy can be checked by placing the iso-align on top of the treatment couch with its plane perpendicular to the couch, the gantry can be placed either at 90° or 270° . During the irradiation of the iso-align with a predefined field size (e.g. $10 \times 10 \text{ cm}^2$) EPID images are acquired. Then the treatment table is moved vertically over a certain distance (e.g. 5 cm) and again the iso-align is irradiated and EPID images are acquired. Now one can compute the real distance travelled by comparing certain points (e.g. metallic dots in the phantom) on both images. The procedures for checking accuracy of the lateral and longitudinal are analogous but it should be clear other iso-align orientations and gantry angles should be used [11]. The travelled distance should not deviate more than 2 mm from the value given by the position indicators [10]. The rotational position indicators can also be checked by comparing EPID images acquired at different angles, the deviation should not be more than 1° [10].

5 Image processing

The following section will discuss some basic principles of image processing. The intention of this section is to give the reader an introduction to the techniques used in this dissertation. It is certainly not a complete guide on digital image processing¹.

5.1 DICOM images

The images acquired from LINAC QA procedures, CT-scans, etc. are stored, transferred and reviewed as DICOM-files. DICOM stands for Digital Imaging and Communication in Medicine and it is not just an image or file format. Instead it is a data transfer, storage and display protocol which is designed to include all aspects of digital imaging. Nowadays DICOM is a universal standard in digital medical imaging. All current clinical imaging devices produce DICOM files and transfer these files through a DICOM network. DICOM therefore implicitly controls the medical workflow [23].

DICOM also offers full support for the storage of multiple parameters of different data types. A DICOM does not only store the images, it also stores parameters such as patient ID, patient position, several imaging device parameters, and so on. All these parameters (attributes) are encoded based on the DICOM data dictionary² which contains more than 2000 standardized attributes designed to enclose all aspects of medical imaging and diagnostics [23].

Each of these attributes has its own (also standardized) tag so it is easily accessible. This tag is has the form of a (x, y) coordinate where both x and y are hexadecimal numbers. The first hexadecimal number defines the group where the attribute belongs to and the second refers to the attribute itself. Each attribute is formatted in one of the 27 data types or value representations (VR's). For example, the attribute "content time" gives the time at which the data was acquired and its data type (VR) is Time (TM). This means according to the DICOM standard that it is represented as HHMMSS where H,M and S stands for hours, minutes and seconds respectively [23].

Finally, DICOM images show excellent image quality. DICOM provides up to 65,536 shades of grey (16 bits) and therefore captures the slightest nuances. In comparison, for example a JPEG file is always limited to 256 shades of grey (8 bits) which makes them often unsuitable for diagnostic purposes [23].

¹ For a more complete overview the reader is referred to: [24].

² The DICOM data dictionary is accessible at: <http://dicom.nema.org/standard.html>.

5.2 Basics of digital image processing

5.2.1 What is a digital image ?

An image can be represented by a function, $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at a certain point defines the intensity (or grey level) of the image at that point. The image is called a digital image when all x , y and amplitude values are finite, discrete quantities. A digital image therefore is composed of a certain finite amount of elements which each have a position (x, y) and a certain intensity value. These elements are called picture elements or pixels [24]. From now on in this dissertation, the term image always refers to a digital image.

Digital image processing envelops all processes or operations whose inputs and outputs are images and also operations that extract data from the image, including object recognition [24].

5.2.2 Images in MATLAB and coordinate conventions

A digital image $f(x, y)$ is represented in MATLAB by a two-dimensional array (matrix). Assuming the image has m rows and n columns, we can say the image is of size $m \times n$. MATLAB's image processing toolbox uses two different coordinate systems: one based on pixel indices and one based on spatial coordinates [24], [25].

The most common method for indicating certain positions in an image is by means of pixel indices. This method concerns the image to be a matrix of discrete elements, with its rows ordered from top to bottom and its columns from left to right. Another method for expressing positions in an image is to use continuous coordinates instead of discrete indices. In this spatial coordinate system locations are positions on a plane described in terms of x and y rather than speaking about rows and columns (pixel indexing system) [25]. Both coordinate systems are shown in Figure 5.1.

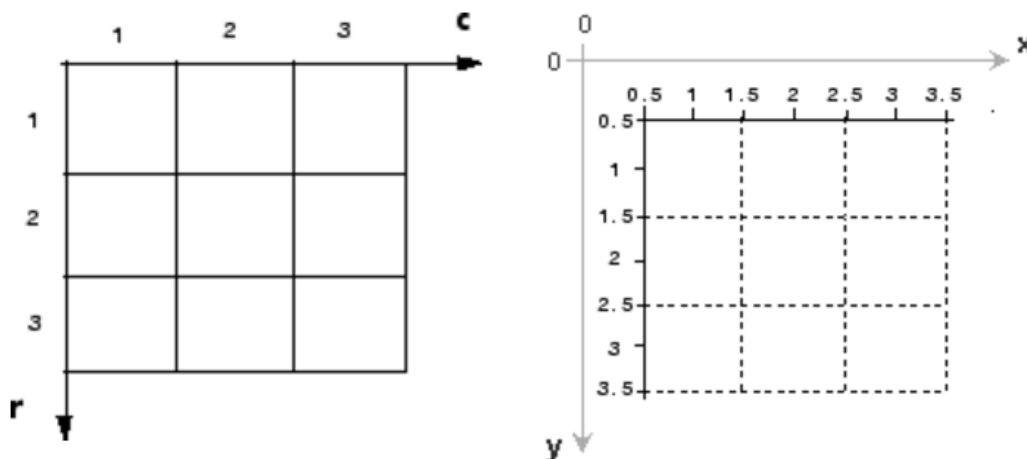


Figure 5.1: A representation of the pixel index coordinate system (left) and the spatial coordinate system (right) [22].

The preceding discussion together with the discussed coordinate systems leads to the following representation for a $m \times n$ digital image using the spatial coordinate system:

$$f(x, y) = \begin{bmatrix} f(0,0) & f(1,0) & \dots & f(n-1,0) \\ f(0,1) & f(1,1) & \dots & f(n-1,1) \\ \dots & \dots & \dots & \dots \\ f(0,m-1) & f(1,m-1) & \dots & f(n-1,m-1) \end{bmatrix} \quad (1)$$

A digital $m \times n$ image can also be represented in MATLAB using the pixel index system as followed:

$$g(r, c) = \begin{bmatrix} g(1,1) & g(1,2) & \dots & g(1,n) \\ g(2,1) & g(2,2) & \dots & g(2,n) \\ \dots & \dots & \dots & \dots \\ g(m,1) & g(m,2) & \dots & g(m,n) \end{bmatrix} \quad (2)$$

Both representations are identical except for the shift in origin, note that $f(0,0) = g(1,1)$ [24].

5.2.3 Image classes and image types in MATLAB

MATLAB and the image processing toolbox provide multiple classes for representing pixel intensity values. Classes *uint8*, *uint16* and *logical* are the most common. An image of class *uint8* represents the intensity value of a pixel as an unsigned 8-bit integer in the range $[0; 2^8 - 1]$ and therefore uses one byte to represent each pixel. In medical imaging techniques *uint16* is much more common than *uint8* because these techniques often require a higher dynamic range. The class *uint16* provides this by using two bytes to represent each pixel, thus as an unsigned 16-bit integer in the range of $[0; 2^{16} - 1]$. Finally, class *logical* uses one byte to represent each pixel value as either one or zero [24].

Besides different image classes, there are also different image types being: grey-scale images, binary images and RGB images. Grey-scale images are data matrices whose values represent different shades of grey. Binary images have a specific meaning in MATLAB. It is defined as a logical array of zeros and ones. For example, an array of zeros and ones of class *uint8* is not considered to be a binary image. Finally a RGB image is simply a colour image. RGB images will not be used in this dissertation and therefore they are not discussed [24].

5.3 Intensity transformations and filtering in the spatial domain

Intensity transformations and spatial filtering are operations in the spatial domain. Spatial operations are based on direct manipulation of pixels. Spatial operations can be represented as followed:

$$g(x, y) = T[f(x, y)] \quad (3)$$

where $f(x, y)$ and $g(x, y)$ are the input and output images respectively. The operator T is defined over a specified neighbourhood about point (x, y) . The basic approach is to define a rectangular neighbourhood centred at point (x, y) , as shown in Figure 5.2. This centre is moved from pixel to pixel throughout the image and so it will envelop different neighbourhoods during its travel. At each location (x, y) , operator T will compute its output g . The value of g at a certain point therefore depends on all the intensity values in the neighbourhood of f [24].

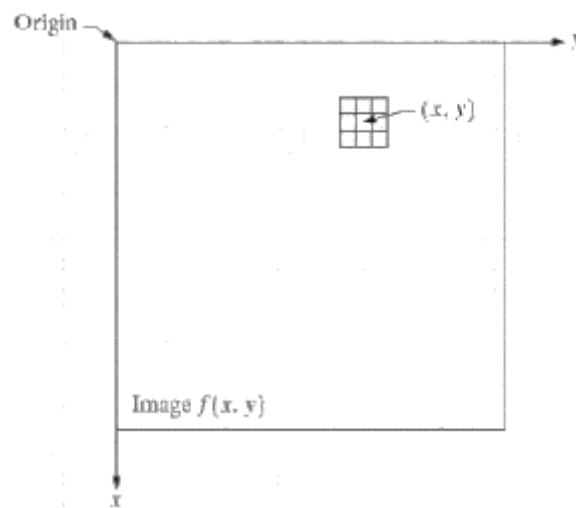


Figure 5.2: A 3x3 neighbourhood centred at (x, y) in an image $f(x, y)$ [26].

5.3.1 Intensity transformation

The intensity transformation is based on the simplest form of operator T , when the neighbourhood is just one pixel in size (1 x 1). When this is the case, the value of g , at a certain point, only depends on the value of f at that point [24].

MATLAB and the image processing toolbox provide the function `imadjust()` to perform intensity transformations, its syntax is the following:

$$g = \text{imadjust}(f, [low_in \ high_in], [low_out \ high_out], gamma) \quad (4)$$

where f and g are the input and output images respectively. This function transfers the intensity values in the input image to other values in the output image. Default, $gamma$ equals one and the intensities in the range $[low_in \ high_in]$ are linearly transferred to new values in the range $[low_out \ high_out]$.

The value of gamma specifies the shape of the curve that is used to transfer the values, as shown in Figure 5.3. for example, if gamma is greater than one the transfer is weighted towards lower (darker) output values and therefore dark details will become more visible [24].

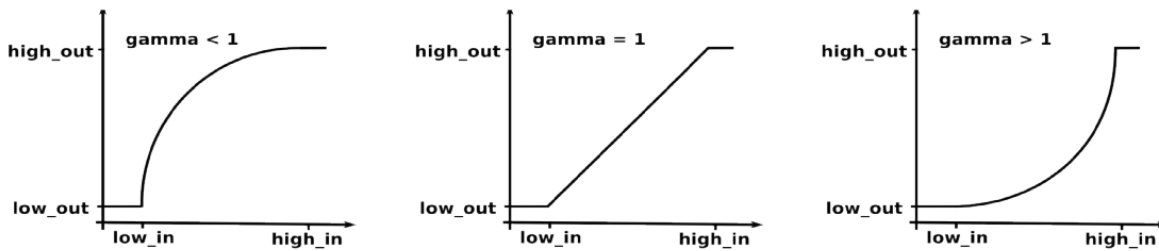


Figure 5.3: The different transfer curves that are available in function `imadjust` [27].

5.3.2 Median filter

Median filtering is used when an image is corrupted with noise, e.g. salt-and-pepper noise. The image processing allows to implement a 2D median filter using the following syntax:

$$g = \text{medfilt2}(f, [m, n], \text{padopt}) \quad (5)$$

where f and g are the input and output images respectively and the matrix $[m, n]$ defines the neighbourhood of size $m \times n$ over which the median has to be computed. Finally, the parameter `padopt` specifies a border padding option. One can choose to pad the borders with zeros, ones or the input image can be extended symmetrically by mirror-reflecting it across its border [24].

5.4 Morphological operations

Morphological operations can be used to extract components that are of interest from an image [24]. Two fundamental operations will be discussed being: dilation and erosion. Afterwards, an important characteristic and the different shapes of structuring elements are briefly explained. All the morphological operations discussed below are only applicable to binary images.

5.4.1 Dilation and erosion

Dilation is a morphological operation that thickens objects in an image. The specific way of this widening is specified by a so called structuring element. A 3×3 matrix containing all ones is a common used structuring element to perform dilation. The origin (the value at the centre) of the structuring element is translated throughout the image to check where the element overlaps one-valued pixels. The dilated image is one at each position of the origin of the structuring element such that the element overlaps at least one one-valued pixel of the input image [24]. Figure 5.4 shows the result of a dilation with a 3×3 structuring element.

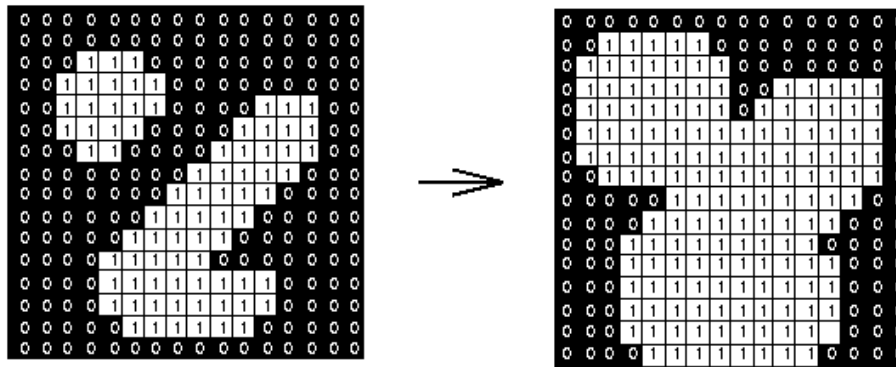


Figure 5.4: The original image (left) and the output image after performing a dilation with a 3x3 square structure element [28].

Erosion is the opposite of dilation and therefore this operation shrinks objects in a binary image. The manner of this shrinking is again controlled by a structure element. The structure element is translated throughout the image to see where it overlaps one-valued pixels. The output image has a value of one at each position of the origin of the structure element, in such a way that the element overlaps only one-valued pixels. In other words, where the element does not overlap any background pixels. The overall result is a shrinking of the objects in the image as showed in Figure 5.5.

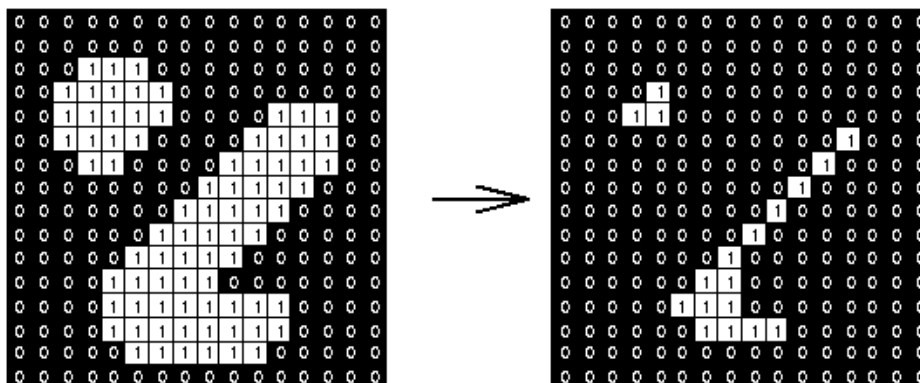


Figure 5.5: The original image (left) and the output image after performing an erosion with a 3x3 square structure element [28].

5.4.2 Structuring element characteristics and shapes

One important aspect from the dilation operation is that it is associative. Suppose having two structuring elements B_1 and B_2 and an input image A . The first dilating A with B_1 and dilating this result with B_2 gives the same result as first dilating A with B_2 and afterwards with B_1 . This associativity is an important characteristic as the time required to compute dilation or erosion is proportional to the amount of nonzero pixels in the structuring element. Let's assume a dilation with a 5 x 5 square structuring element filled with ones. One can obtain the same result performing two dilations, one with a 1 x 5 row element and one with a 5 x 1 row element. The 5 x 5 element contains 25 nonzero values while the two other elements together count 10 nonzero values and therefore the second method is 2,5 times faster than the first one [24].

There are many different structuring element shapes available in MATLAB, the imaging processing toolbox provides the following function.

$$se = strel(shape, parameters) \tag{6}$$

The parameter shape can be filled in with diamond, disk, square, rectangle, etc. to create different shapes. Besides the shape it may be necessary to define other parameters such as radius in case of a disk or size in case of a rectangular shape [24].

5.5 Image segmentation

In this paragraph some techniques are discussed for detecting isolated points, lines and edges in images. The most common used method to look for discontinuities is to translate a mask through the image in a manner similar to spatial filtering techniques. A filtering mask is similar to a structuring element in morphological operations. The mask is translated throughout the input image f going from pixel to pixel to yield at every position (x, y) of f the response R . For a 3×3 mask this involves the calculation of the sum of products of the coefficients with the intensity levels contained in the region enveloped by the mask. The response R at any location is given by:

$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

$$= \sum_{i=1}^9 w_i z_i \tag{7}$$

where w_i is the filter coefficient, as shown in Figure 5.6, the pixel intensities are denoted by z_i . The response of the mask is defined at its origin [24].

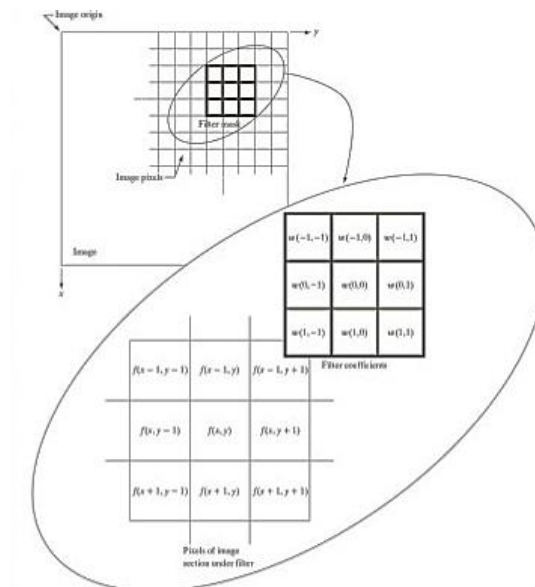


Figure 5.6: The different coordinate systems in filter techniques. w denotes the filter (or mask) coordinates and $f(x, y)$ denotes coordinates of the underlying original image [29].

5.5.1 Point and line detection

Isolated points that are embedded in an area with roughly constant intensity values can be detected by using the following mask.

$$W_{point} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (8)$$

An isolated point has been detected at these points where the mask is centred if the response meets the following criterion:

$$|R| \geq T \quad (9)$$

where T is a positive valued threshold. The strongest response of the mask should be at these points where the mask is centred on an isolated point and the response should be 0 at locations of constant intensity [24].

Lines in an image can be detected using the same method. It should be clear that for every orientation of the line that has to be detected one should use a different mask. Masks to detect horizontal lines and lines angled at 45° (with a width of one pixel) are given below.

$$W_{horizontalline} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad (10)$$

$$W_{Line45^\circ} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \quad (11)$$

5.5.2 Edge detection and MATLAB implementation

Edge detection is the most common method to detect discontinuities in intensity values in an image. This discontinuities are detected using the first- or second-order derivatives of the image. The most common first-derivative form used in image processing is the gradient. The gradient of a two dimensional function, $f(x, y)$, is defined as being the vector:

$$\nabla \mathbf{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (12)$$

and the magnitude of this vector is given in equation 13.

$$\begin{aligned} \nabla f = \text{mag}(\nabla \mathbf{f}) &= \left[g_x^2 + g_y^2 \right]^{\frac{1}{2}} \\ &= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right] \end{aligned} \quad (13)$$

For simplification, this magnitude is approximated by omitting the square-root or by using absolute values.

$$\nabla f \approx g_x^2 + g_y^2 \quad (14)$$

$$\nabla f \approx |g_x| + |g_y| \quad (15)$$

The approximation of the magnitude of the gradient behave in the same manner as the gradient; that is, as first-order derivatives. They are zero in regions of constant intensity and their values are related to the rate of change of intensity in other regions. The gradient vector points in the direction of the maximum rate of change of f at point (x, y) [24].

The aim of edge detection is to find places in an image where the intensity changes rapidly and more specific places where the magnitude of the first derivative is higher than a predefined threshold. The image processing toolbox provides multiple edge estimators, it is possible to choose to detect vertical or horizontal edges or both [24]. The general syntax is:

$$[g, t] = \text{edge}(f, 'method', parameters) \quad (16)$$

where f is the input image and parameters are additional parameters which are characteristic for the type of method that has been chosen and will be discussed in the following paragraphs. The output g is a logical image containing ones at places where edges are detected and zeros elsewhere. Finally, parameter t specifies the threshold used by function `edge` to determine which gradient values are strong enough [24].

In this dissertation the sobel operator is chosen to fill in the 'method' parameter in function `edges`. The sobel edge detector uses first-order derivatives which are digitally approximated by differences. This operator uses the following differences between rows and columns (see Figure 5.7).

$$\begin{aligned} \nabla f &= \left[g_x^2 + g_y^2 \right]^{\frac{1}{2}} \\ &= \left\{ \left[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \right]^2 \right. \\ &\quad \left. + \left[(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \right]^2 \right\}^{\frac{1}{2}} \end{aligned} \quad (17)$$

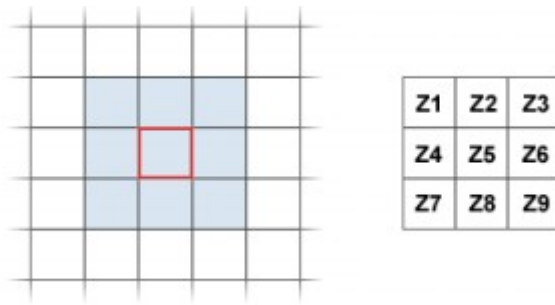


Figure 5.7: A 3x3 neighbourhood and the indices used to specify locations in the neighbourhood [30].

The general syntax for the Sobel detector is:

$$[g, t] = \text{edge}(f, 'sobel', T, dir) \quad (18)$$

where f is the input image, 'sobel' defines the gradient approximation method, T is a predefined threshold and dir specifies the direction of the detected edges and can be either horizontal, vertical or both [24].

6 The mechanical QA protocol of L.O.C.

This paragraph deals with the periodic mechanical QA protocol of L.O.C. Three main parts that have to be automated can be distinguished: the distance between radiation- and mechanical isocentre, the deviation on the table position indicators in three dimensions and the deviation on asymmetric field sizes. These three parts will be further outlined below³.

6.1 Distance between radiation- and mechanical isocentre

The test procedure for calculating the distance between the radiation- and mechanical isocentre can be compared with the Winston-Lutz test described in paragraph 4.2.3. The test procedure utilised by L.O.C. involves the irradiation of an iso-align device which is set up in such a way that the radio-opaque ball at its centre coincides with the mechanical isocentre which is defined by the centre of the light field. During irradiation EPID images are acquired and saved for analysis.

The radiation- and mechanical isocentre, like described earlier, cannot be determined unambiguously and therefore this test is performed for 13 different setups which are given in Table 6.1. Each setup involves a different combination of gantry, collimator and table angles. For each of these setups the radiation isocentre is determined by defining it as the centre of the radiation field, subsequently the mechanical isocentre can be found by determining the centre of the radio-opaque ball. When both the mechanical- and radiation isocentre are determined, the distance between the two can be measured.

The tolerance level for this distance is set on 1 mm and the action level amounts 2 mm. The rotational movement of the table (image 4 and 5 in Table 6.1) is an exception on these values and has a tolerance level of 2 mm and an action level of 3 mm for the distance between the radiation- and mechanical isocentre.

Table 6.1: The 13 different setups (of L.O.C.) for the determination of the distance between radiation- and mechanical isocentre.

Image no.	Gantry angle (°)	Collimator angle (°)	Table angle (°)
1	0	0	0
2	0	90	0
3	0	165	0
4	0	270	0
5	0	0	90
6	0	0	270
7	180	0	0
8	90	90	0
9	90	0	0
10	90	270	0
11	270	270	0
12	270	0	0
13	270	90	0

³ The entire mechanical QA protocol of L.O.C. is given in appendix A (in Dutch).

6.2 Deviation on table position indicators in three dimensions

The deviation on the treatment couch position indicators is determined in three dimensions being: longitudinal, lateral and vertical. L.O.C.'s protocol re-uses two images from Table 6.1, more specifically image 1 for the longitudinal and lateral motion and image 12 for the vertical table motion.

The procedure for the longitudinal table motion involves the irradiation of the iso-align in a setup similar to this of image 1 in Table 6.1 but this time the treatment couch is displaced 15 cm in the longitudinal direction. To determine how many cm's the couch has travelled one has to find a certain point on the image (e.g. one of the radio-opaque balls of the iso-align device) which is visible on both images. By merging both images one can measure the distance that the treatment table has travelled. The procedure for the lateral and vertical motion is completely identical to this described above but the procedure for the vertical motion uses image 12 from Table 6.1 as a reference instead of image 1.

The tolerance level for the deviation on the table motion is set to 1 mm and the action level amounts 2 mm, this applies to each of the three dimensions. When the table has moved vertically the longitudinal displacement of the table ideally is zero. In reality this is not the case and the tolerance value for this longitudinal displacement amounts 1° which corresponds to a longitudinal distance of 2,6 mm. The action level is set on 2° corresponding to a longitudinal displacement of 5,2 mm. The same holds for the lateral and longitudinal displacements when the table is moved in the longitudinal and lateral direction respectively.

6.3 Deviation on asymmetrical field sizes

The protocol of L.O.C. uses three different predefined square field sizes to check the size in both dimensions and check the symmetry of the field, these field sizes are: 5 x 5, 10 x 10 and 18 x 18 cm² and the EPID images are acquired at gantry-, collimator- and table angles of zero. To acquire these images there is no iso-align needed, only the radiation field is captured by the EPID. The radiation isocentre used for the asymmetrical field size measurements is the averaged isocentre of the first four images of Table 6.1 and not just the radiation isocentre of the first. This is done because while acquiring the first image (and the others) of Table 6.1 one of the jaws could be mispositioned. If this is the case this has a direct effect on the position and size of the radiation isocentre [19]. Therefore this effect has to be averaged out by rotating the collimator over its full range. This is visualised in Figure 6.1, the black lines define the normal situation at a collimator angle of zero degree. The vertical red line defines a mispositioned jaw and the intersection of the dashed diagonals define the corresponding radiation isocentre. Note how the isocentre describes a circle when rotating the collimator, the centre of mass of this circle coincides with the averaged isocentre discussed above. After this averaged radiation isocentre is established the 4 asymmetrical field sizes are determined in the same manner as given in Figure 4.3.

The tolerance value for the error on x_1 , x_2 , y_1 and y_2 has a value of 1 mm for the 5 x 5 and 10 x 10 cm² fields and these fields have an action level for these parameters which is set on 2 mm. The tolerance value for the 18 x 18 cm² field amounts 1 % of the total field size which is roughly 2 mm and the action level is set on 2 % of the total field size and amounts roughly 4 mm. The tolerance levels for the total field sizes in both direction are exactly the same as these stated above for the asymmetrical parameters.

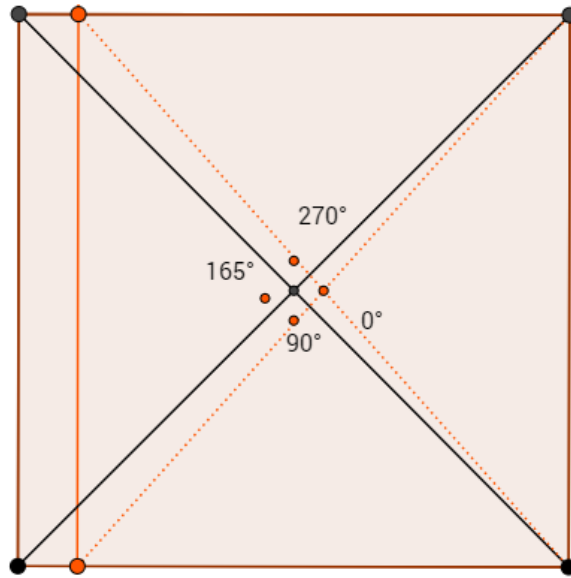


Figure 6.1: Result of miscalibrated jaw on the place and size of the radiation isocentre. Note how these points can be averaged out in order to calculate asymmetrical field size parameters.

7 Materials & Methods

The application was developed and tested using MATLAB. The algorithms were designed using standard MATLAB features combined with functionalities from the Image Processing Toolbox and Symbolic Math Toolbox. The images for testing were acquired using a Varian CLINAC as well as a Varian Truebeam linear accelerator which are both equipped with the same type of amorphous-silicon EPID. The only difference is that the EPID of the CLINAC provided images with a resolution of 768 x 1024, while the resolution provided by the EPID of the Truebeam only was 384 x 512. Throughout the next sections the terms half resolution- and full resolution images refers to the images acquired with the Truebeam and CLINAC device respectively. All the EPIDs used in this dissertation are calibrated to perform dosimetry and have a quasi linear dose response.

This section will describe the architectures of the different algorithms separately followed by a description of the structure of the application as a whole. Subsequently, the methods of determining accuracy and reproducibility of the software are briefly discussed. This section is rather mend to describe the structure of the application than to dreary enlarge on the source code itself. The source code is added to appendix B.

7.1 Distance between radiation- and mechanical isocentre

The problem of calculating the distance between radiation- and mechanical isocentre was solved using two different approaches. Both algorithms were tested and one has proven to be faster while the other has proven to be more accurate. Because time, speaking about ± 1 minute, is not an issue the most accurate algorithm is used for implementation. The architecture of this algorithm is given in Figure 7.1 and will be discussed in detail throughout this section. The algorithm that has been designated as inferior will be briefly discussed in section 7.1.7.

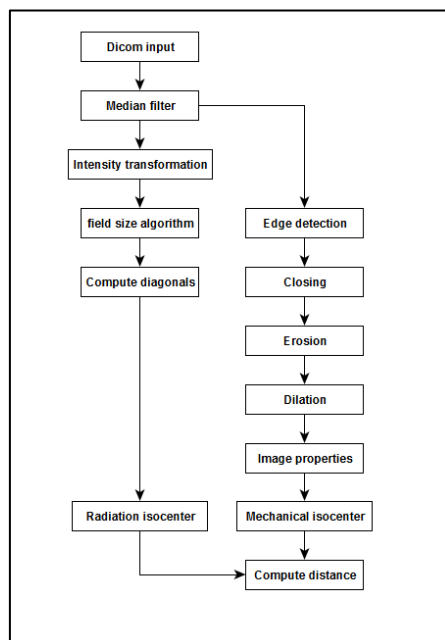


Figure 7.1: Architecture of the algorithm for computing the distance between the radiation- and mechanical isocenter.

7.1.1 Pre-processing radiation isocentre detection

The Pre-processing step can be divided into two parts: noise reduction and contrast stretching which are the first two blocks following the input file in Figure 7.1. Since the images obtained from the EPID can be noisy a median filter with a 3×3 filter mask is implemented to reduce the amount of noise in the input image.

Since a DICOM image uses 16 bits per pixel to visualise different shades of grey, it has a range of $[0 - 65\,535]$ which allows high contrast images to be represented [23]. But in this case the input image uses only one fifth of this range which results in a low dynamic range and contrast, an example of a histogram of an input image is given in Figure 7.2. In order to create a wider dynamic range an intensity transformation is implemented with the following MATLAB command:

$$g = imadjust(f, stretchlim(f), [_]); \quad (19)$$

here f and g are the input and output images respectively. The function *stretchlim* gives an output *low_High* from which the values specify the intensity levels that saturate the bottom and top 1% of all pixel values in the input image [24]. The third parameter in equation 19 is equivalent to $[0 \ 1]$ which means that the pixel values have to be mapped over the entire range being $[0 - 65\,535]$. The histogram of the result of this contrast stretching operation is given in Figure 7.2.

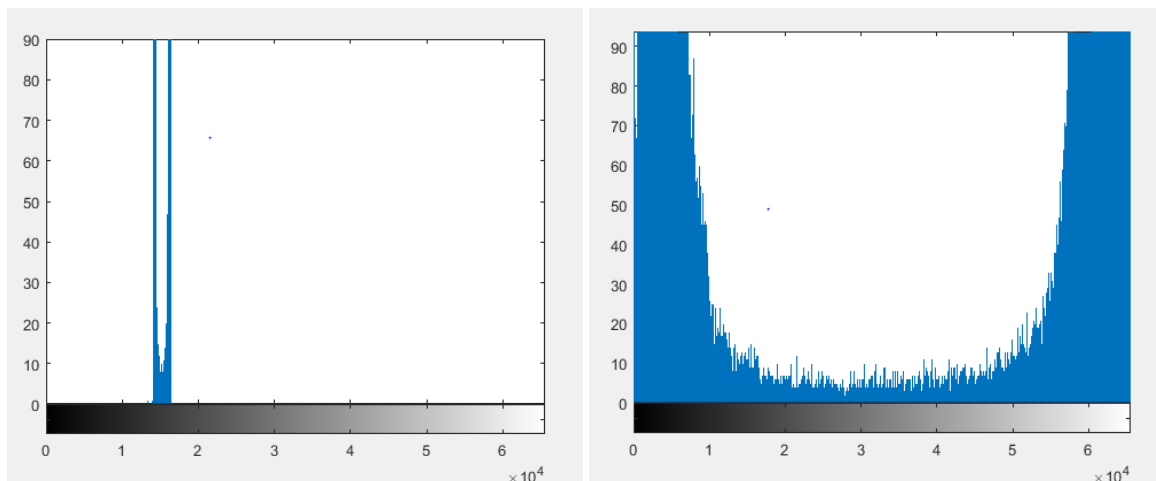


Figure 7.2: Left: The histogram of the input image. Right: The histogram of the input image after contrast stretching. The x-axis denotes the 65536 shades of grey and the y-axis visualizes the amount of pixels with a certain intensity value.

7.1.2 Field edge algorithm

In order to understand how a field edge can be computed one should be aware of the definition of field size. The field size is defined as the full width at half maximum (FWHM) of the cross beam profile (profile similar to Figure 2.2) [4]. Working with EPID images of EPIDs with a (quasi) linear dose response the field size is the FWHM of the intensity spectrum of the image. Figure 7.3 shows a pre-processed input image and the blue line defines the range over which the spectrum is computed, the resulting profile is also shown.

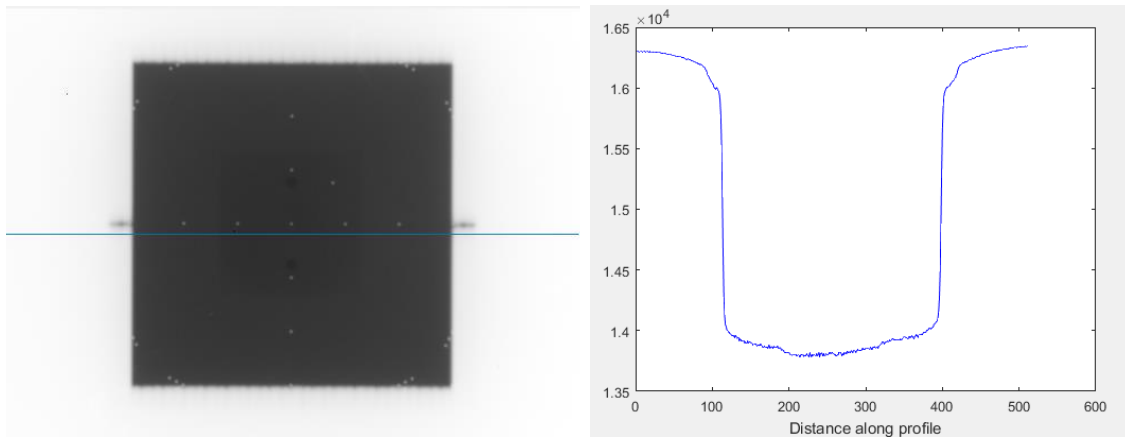


Figure 7.3: On the left an input image is shown where the blue line defines the range over which the spectrum is computed. On the right the resulting crossbeam profile is shown.

Two functions were created in order to compute the field edges individually, one for the vertical and one for the horizontal field edges. These functions acquire multiple crossbeam profiles and calculate the x -values for vertical and y -values for horizontal edges of places where the intensity is at the half of its minimum.

The first step of the field edge algorithm involves calculating the median intensity value of the intensity range of the pre-processed input image, this will be the target value. The next step involves guessing where the field edge is located and acquiring little crossbeam profiles at these places. Figure 7.4 shows the ranges over which the crossbeam profiles are computed (blue lines). The start and stop signs show where the scanning procedure starts and ends, in between multiple crossbeam profiles are acquired. The scanning procedure is divided in two sections to make sure the profiles do not cross one of the radio-opaque spheres or other structures of the iso-align because this could give rise to misdetections.

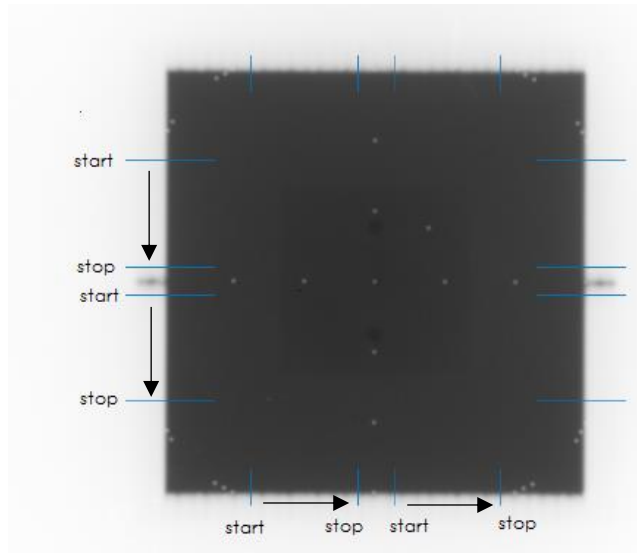


Figure 7.4: Visualization of the edge scanning procedure, input image with ranges over which crossbeam profiles are computed.

Each little crossbeam profile acquired by the algorithm yields a table with 100 x -values, y -values and the corresponding intensity values at those points. Assuming the computation of the left vertical field edge, for each profile the y -value is stored, subsequently, the intensity value which lays the closest to the target value is sought and the corresponding x -value is acquired. If this intensity value appears more than one time in a row their corresponding x -values are averaged. When searching for the target value, two different outcomes are possible:

1. the found value is exactly equal to the target value;
2. the found value is smaller or greater than the target value.

When the found value is exactly equal to the target value, the averaged x -value is simply stored together with the (already) stored y -value after which the next profile is acquired. When the found value is not exactly equal to the target value linear interpolation is performed. If the found intensity value is greater than the target value, the algorithm searches for the next intensity value in the table which is smaller than the target value and the corresponding x -value (whether or not averaged) is acquired. Subsequently linear interpolation is performed to calculate the x -value which corresponds to the target intensity value. Using linear interpolation assumes that the region in the intensity cross profile (Figure 7.3) where the intensity is at the half of its minimum can be approximated by a linear function. When interpolation is finished the x -value is stored together with the (already) stored y -value and the next profile is acquired. When the found intensity value is smaller than the target value the same interpolation procedure is performed but this time with an interpolation partner which is greater than the target value.

When the scanning procedure is finished and all the edge points are stored the average and standard deviation is computed for the x -value dataset. These two parameters are used to detect misdetections based on the detection of outliers in the dataset. A certain value is designated as an outlier when it is greater than the average value plus three times the standard deviation or when it is smaller than the average value minus three times the standard deviation. These outliers and their corresponding y -values are deleted from the dataset.

The final step of the field edge algorithm consists of fitting a linear curve through the found edge points. The curve fitting is performed using the least square method. In order to use the least square method to fit vertical functions the x - and y -values are exchanged and at the end the fitted curve is inverted.

The above described procedure assuming a vertical field edge is completely analogous for horizontal field edges. Note that the above procedure also could have been performed by means of edge-detection. The reason why this is not the case is because this requires working with the function *regionprops*, to measure the properties of the field, which does not allow to detect oblique field edges⁴.

7.1.3 Computing radiation isocentre

When all the field edges are computed, the intersections of these four functions are calculated in order to determine the diagonals of the square field. The intersection of these two diagonals defines the radiation isocentre of the setup (gantry, collimator and table angle combination). An example of the result of performing the field edge algorithm in combination with the radiation isocentre computation on an input image is given in Figure 7.5.

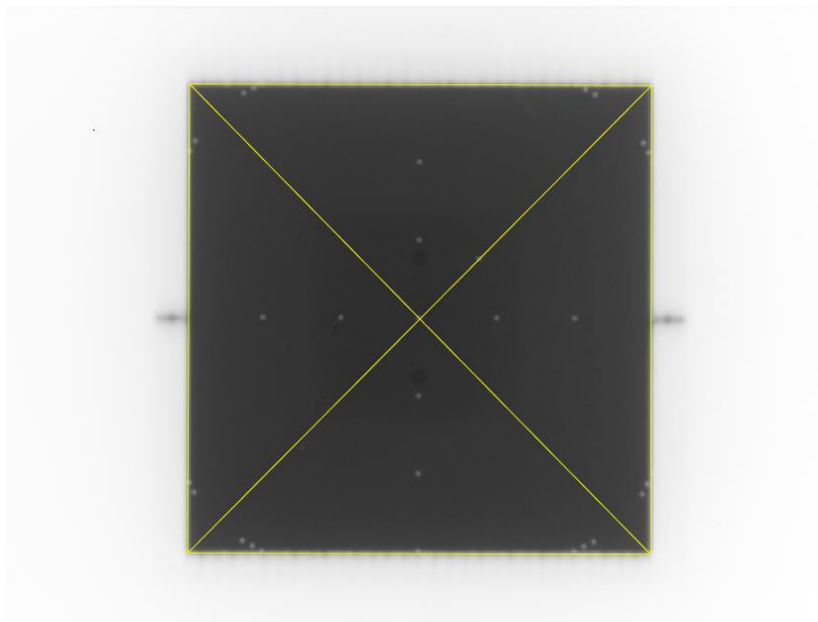


Figure 7.5: The result of performing the field edge algorithm in combination with the radiation isocentre computation.

7.1.4 Pre-processing mechanical isocentre detection

The mechanical isocentre of the gantry-, collimator- and table angle combination coincides with the central radio-opaque sphere of the iso-align device and thus appears as a circle on the EPID images. The process of finding the mechanical isocentre therefore is a process of circle detection. The first step of this procedure involves performing edge detection, followed by several morphological operations to finally detect the circle within a predefined region of interest (ROI).

The sobel operator, which is discussed in section 5.2.2, is used to implement edge detection. First the function *edge* is used to estimate a threshold value. This threshold value is multiplied by a predefined fudge factor from which the value is chosen by trial and error to give the best overall results. Subsequently this new threshold value is used in the *edge* function to perform edge detection using the sobel operator. The result of performing this procedure on an input image yields a binary image

⁴ More information about *regionprops* and the detection of oblique field edges is provided in sections 7.1.5 and 7.1.7.

and is given in Figure 7.6. This result was achieved using the following code:

```
[~, threshold] = edge(f, 'sobel');
corrFactor = 0.5;
g = edge(f, 'sobel', threshold * corrFactor);
```

where f and g are the input- and output images respectively and the fudge (or correction factor) is set to 0,5.

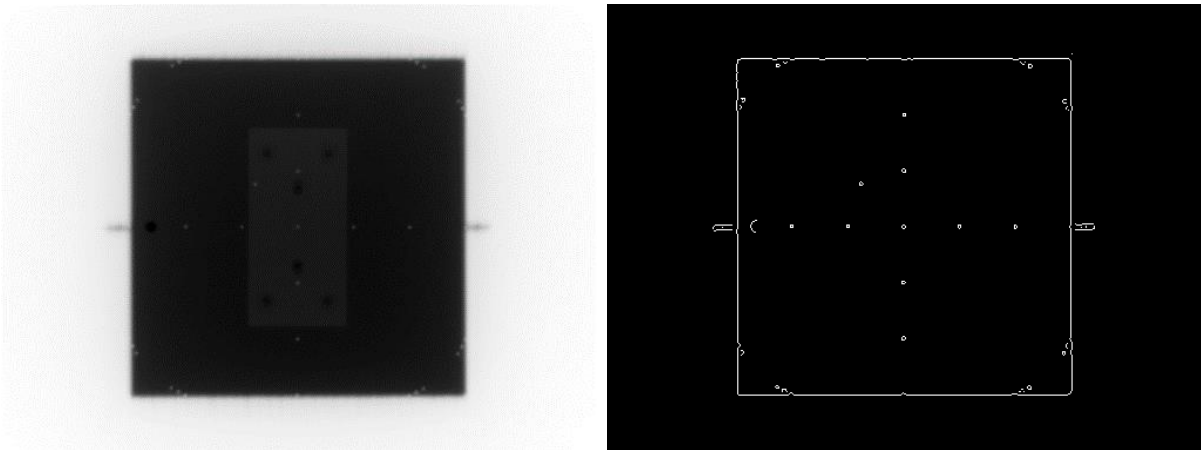


Figure 7.6: On the left the input image is shown and the right image shows the result of the edge detection operation.

The next pre-processing step involves a series of morphological operations. The first operation which is performed is morphological closing. Closing is the process of a dilation followed by an erosion [24]. The closing operation is implemented using a disk shaped structuring element with a radius of one like in equation 20.

$$s = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (20)$$

The result of performing morphological closing on the binary image in Figure 7.6 is given in Figure 7.7. Comparing both images it's clear that the encircled structures of Figure 7.6 are filled in (or closed) in the output image. The next operation that is performed is an erosion to erase little white dots which originate from noise and structure that are not of interest like the field edges. Subsequently a dilation is performed on the result to reconstruct the loss of detail induced by the erosion. Both erosion and dilation operations are performed with disk shaped structuring element with a radius of one like in equation 20. The result of performing both processes subsequently on the closed image is also given in Figure 7.7.

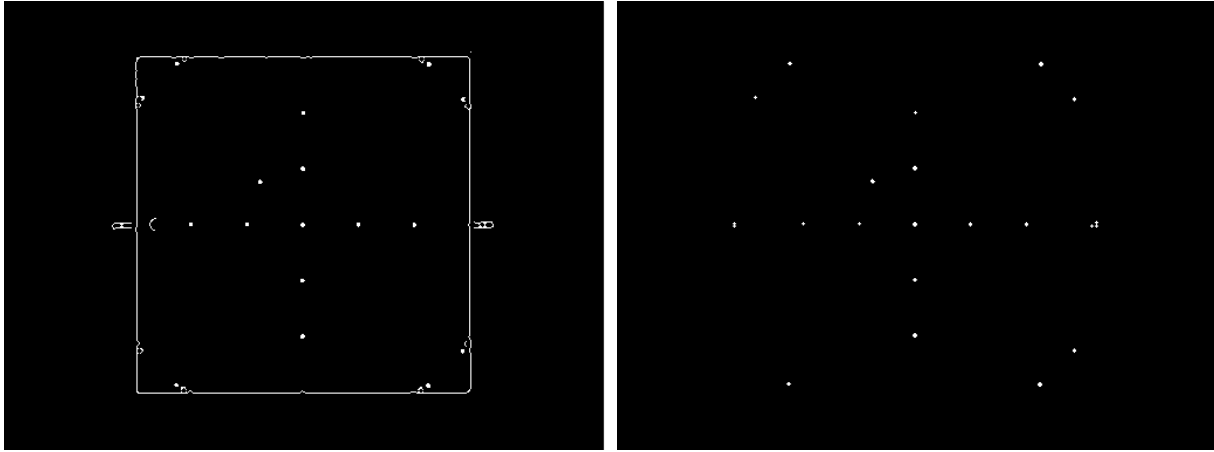


Figure 7.7: On the left the result of the closing operation is shown. The image at the right shows the result of the erosion and dilation operation.

7.1.5 Circle detection procedure

The eventual circle detection is performed with the function *regionprops* which is provided by the Image Processing Toolbox of MATLAB. The function is implemented in the following way:

$$stats = regionprops('table', f, 'Centroid', 'MajorAxisLength', 'MinorAxisLength'); \quad (21)$$

where *f* is the binary image on the right side of Figure 7.7. The parameter *table* denotes that the output *stats* will be represented as a MATLAB table. *Centroid* returns in this table the *x*- and *y*-values of the centres of mass of all the 8-connected regions in the input image. Eight-connectivity is based on pixel connections with one of their 8 neighbours (at the sides and edges) and is an extension of 4-connectivity which is based on pixel connections with only one of the four side neighbours. Finally *MajorAxisLength* and *MinorAxisLength* return a scalar that specifies the length of the major and minor axis length of the ellipse structures detected by *Centroid* [31]. The result of this circle detection operation after visualizing the detected ellipses on an *input* image is given in Figure 7.8.

It's clear, looking at the image at the left of Figure 7.8, that the algorithm still has to select the desired circle, more specifically this at the centre. To achieve this goal a point of interest (POI) is defined which is the radiation isocentre (computed previously). Around this POI a square ROI is established with a width of $2a$, where a is initially set to 0.5 and the POI describes the centre of the ROI. Subsequently the algorithm checks which of the centroids returned by the function *regionprops* lays within the ROI. If there is no centroid which fulfils this criteria the value of a is iteratively increased until there is exactly one centroid which lays inside the ROI. The result of this iterative process is also given in Figure 7.8 and shows only one detected circle, the mechanical isocentre of the setup.

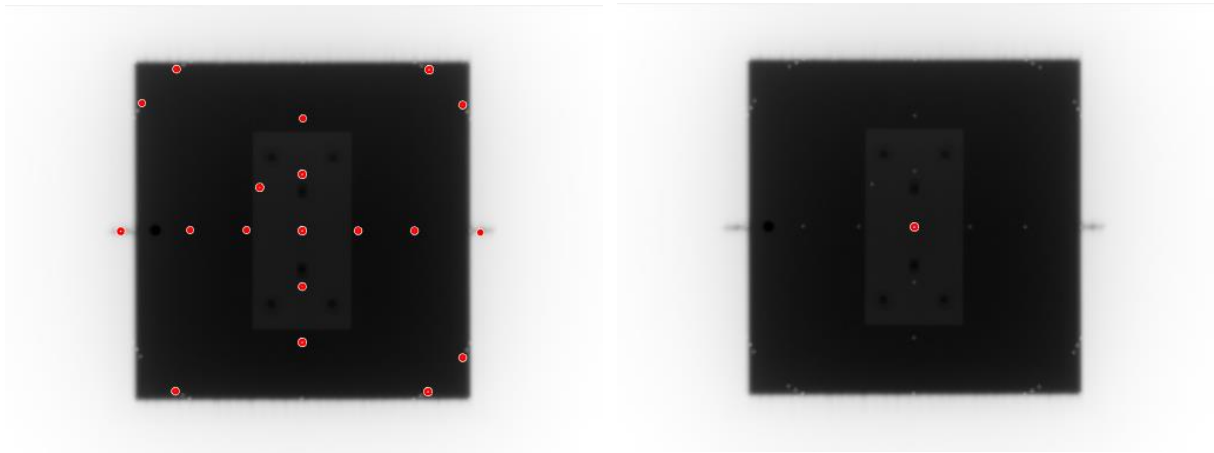


Figure 7.8: The left image shows the result of the function `regionprops` and the right image shows the result of the iterative process for selecting the desired circle.

7.1.6 Computing the distance between both isocentres

When both isocentres are detected these can be plotted onto the input image to yield the eventual output image which will be written to a PDF file. An example of such an output is given in Figure 7.9 together with a detailed image of both isocentres. Looking at Figure 7.9, the mechanical isocentre of the setup is defined by the centroid of the detected circle and the radiation isocentre of the setup is defined by the intersection of the diagonals which are displayed in yellow.

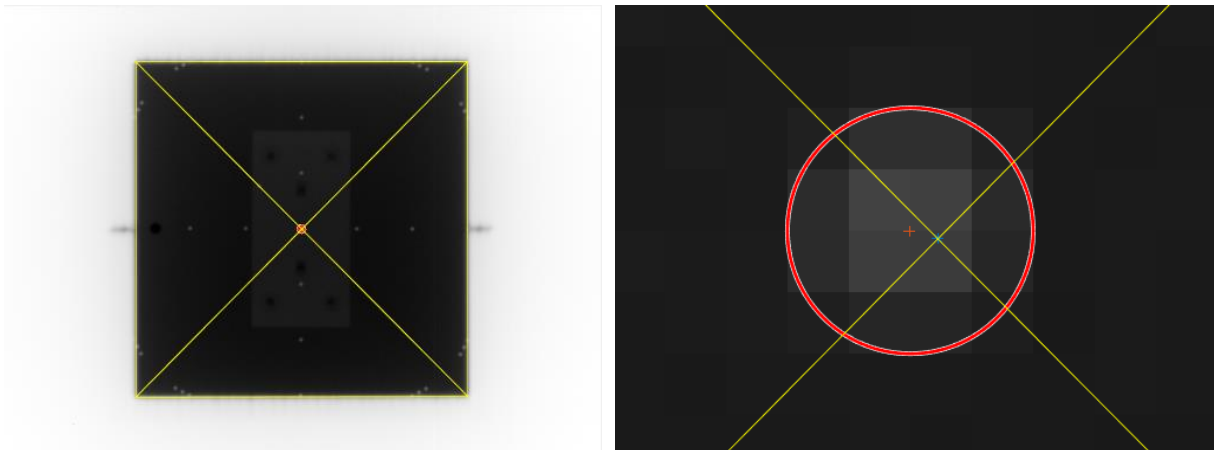


Figure 7.9: The image on the left shows an example of an output file visualizing both the radiation- and mechanical isocentre. The image on the right shows a zoomed view of this output image.

The last step of the algorithm is to compute the distance between the two found isocentres in millimetres. First the distance in pixels is calculated based on the standard mathematic formula of the distance between points and subsequently this result is multiplied by a certain conversion factor to yield the distance in millimetres. The conversion factor (in mm/pixel) is a magnification correction on the pixel size and using position information of the EPID and can be computed as followed.

$$PSI = PR \times \frac{SAD}{RTimageSID} \quad (22)$$

Where PSI stands for pixel size at isocentre, PR is the pixel resolution of the EPID, SAD is the source to axis distance (usually 100 cm, in the protocol of L.O.C. 140 cm) and $RTimageSID$ is the source to imager distance [20]. All these parameters can be found in the DICOM header of the EPID images and should be denoted in millimetre. The PSI in this work has a value of 0,5227 mm/pixel for images acquired from the EPID licenced with half resolution and 0,2613 mm/pixel for images acquired from the EPID licensed with full resolution. This slight difference in magnification correction factor is due to the difference in resolution licenses for the TrueBeam and CLINAC devices at L.O.C.

7.1.7 Alternative algorithm for detection of the radiation isocentre

An alternative algorithm for the detection of the radiation isocentre was designed and tested. The algorithm has proven to be much faster than the algorithm described above, its architecture is given in Figure 7.10. The previous algorithm needs 5.418 seconds to finish the analysis of one picture while this algorithm only needs 1.326 s. This algorithm uses image blurring by means of a Gaussian filter followed by automatic thresholding in combination with the earlier discussed function *regionprops* to detect the field edges. There is one major drawback to this method, that is that function *regionprops* always will draw a rectangle to visualize the field, even when the MLC fails and defines an oblique field edge. The field edge procedure, discussed in section 7.1.2, though can detect oblique field edges and therefore has been chosen to be superior to the fast detection algorithm because this is needed for analysis of images with non-cardinal collimator angles and to detect possible failures of the MLC.

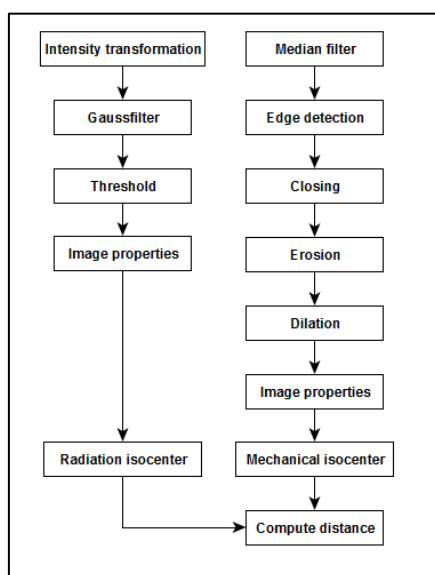


Figure 7.10: The architecture of the discarded alternative algorithm for the computation of the distance between the radiation- and mechanical isocentre. Only the left branch (radiation isocentre) differs from the architecture of Figure 7.1

7.2 Deviation on the table position indicators in three dimensions

The architecture of the algorithm to compute the deviation on the table position indicators is given in Figure 7.11. This algorithm, which is embedded in a function, is designed in threefold for the three dimensions: vertical, longitudinal and lateral. The architecture will only be briefly explained for the longitudinal dimension because the three algorithms only differ in the placements of POI's. The procedures explained in this section will be quite similar to these discussed earlier for the detection of the radiation- and mechanical isocentre.

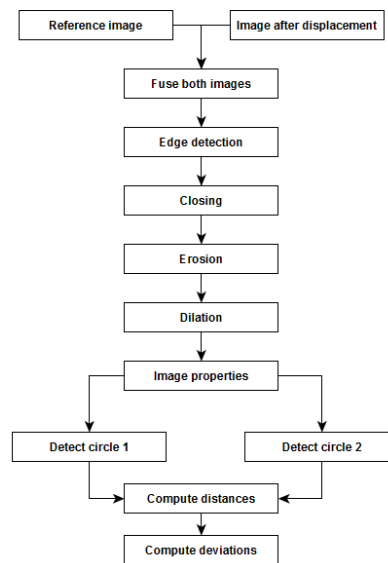


Figure 7.11: Architecture of the algorithm for computing the deviation on the table position indicators in three dimensions.

7.2.1 Image fusing and pre-processing

The first step of the algorithm is to fuse the reference image with the longitudinal displaced image using the function *imfuse* from the MATLAB Image Processing Toolbox. Both input images are shown in Figure 7.12, the fused image is given in Figure 7.13.

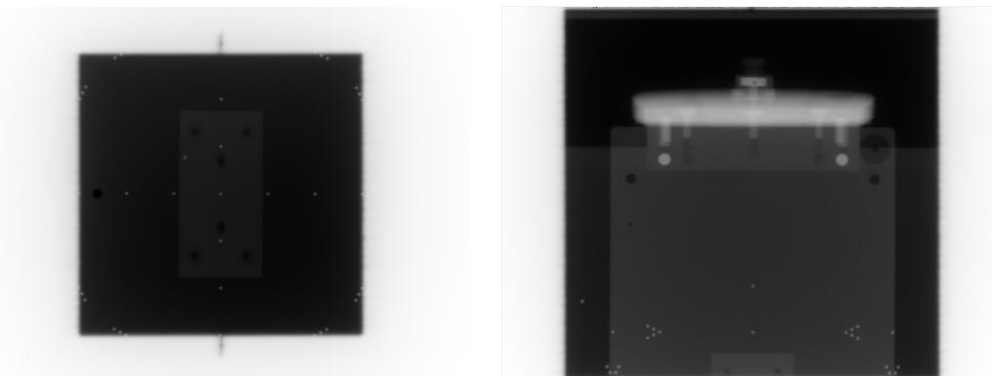


Figure 7.12: On the left the reference image is shown, the right image shows the situation after applying the longitudinal displacement.

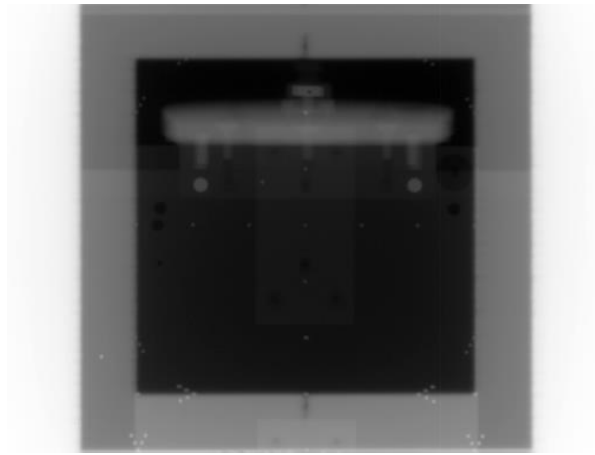


Figure 7.13: Both images of Figure 7.12 fused into one image.

The resulting fused image in Figure 7.13 is used for pre-processing towards the detection of two radio-opaque balls appearing as circles on the fused image. One of these circles must be visible on the reference image, the other one (the same radio-opaque ball but shifted 15cm) on the displaced image and both of the circles must be visible on the fused image. The pre-processing step consists again of the edge detection procedure together with the morphological closing followed by an erosion and subsequently a dilation like in section 7.1.4, the result is shown in Figure 7.14. The arrows in Figure 7.14 point at the radio-opaque ball which is chosen to be detected.

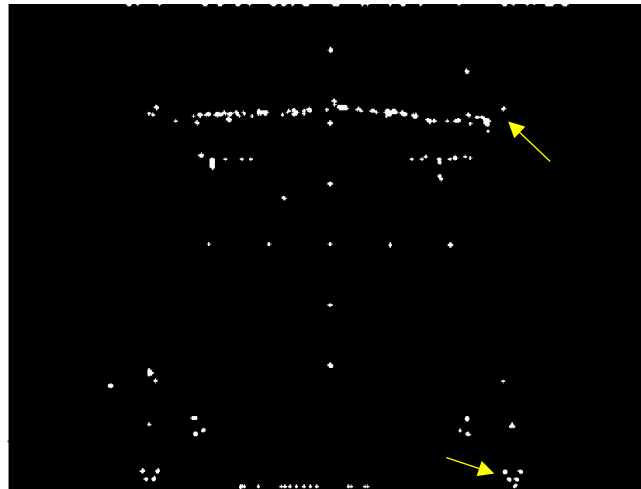


Figure 7.14: The result of performing the whole pre-processing step on the fused image. The yellow arrows point to the radio-opaque spheres that are chosen to be detected.

7.2.2 Circle detection procedure and distance computation

The eventual circle detection is completely analogous to the method described in section 7.1.5 and therefore it will not be discussed again in detail. It is, though, important to discuss the placements of the POI's for circle detection since the POI's in this case are not equal to the radiation isocentre.

The whole imaging procedure of the periodic mechanical QA protocol of L.O.C. is performed each time in exactly the same way since its embedded in a patient file which is loaded into the linac's operating system. For this reason it is safe to assume that the position of the circles that have to be located will be the same for each direction in which the table is shifted. These positions were obtained for both half and full resolution images as well as for both directions in which could be shifted (positive or negative shift). The algorithm first checks the resolutions of the input images and subsequently the direction of the shift in order to know which pair of POI's he needs to perform the iterative process of circle detection.

When both circles are detected the result can be plotted on the original fused image and the distance travelled in longitudinal direction can simply be computed by subtracting the y-values of the centroids of both circles. In the same fashion the lateral displacement, which is supposed to be zero, can be computed by subtracting the x-values of these centroids. Subsequently the deviation is computed in millimetres by subtracting the found longitudinal distance in millimetres from 150 mm. The result of the whole algorithm is given in Figure 7.15 together with a detail image of one of the detected circles.

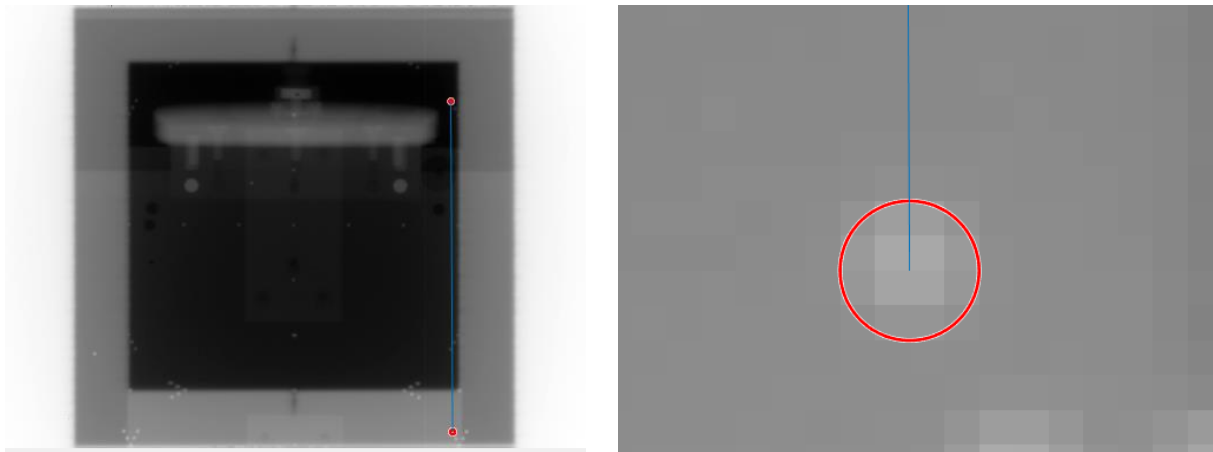


Figure 7.15: Left shows the result of performing the algorithm for computing the deviation on the table position indicators for the longitudinal direction. The image on the right is an detail image of one of the detected circles.

7.3 Deviation on asymmetrical field sizes

The algorithm for the asymmetrical field size computation is similar to this of the detection of the radiation isocentre, its architecture is given in Figure 7.16. The architecture will only be explained for a $5 \times 5 \text{ cm}^2$ field because the computations are exactly the same for all field sizes. Since the first three steps are completely analogous to these discussed in sections 7.1.1 and 7.1.2 their discussion will be skipped.

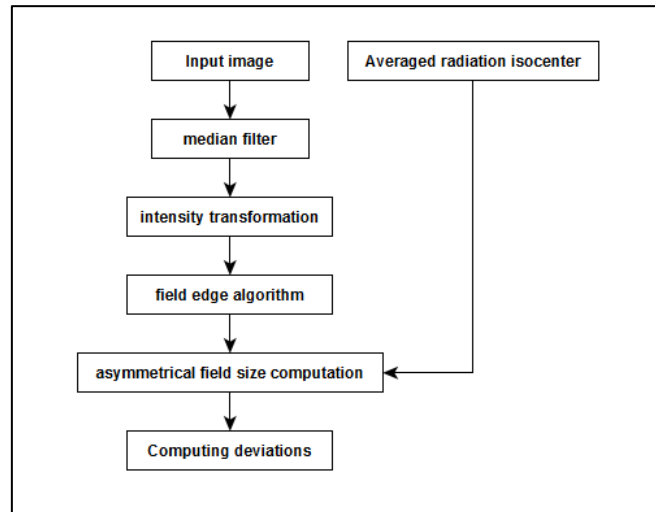


Figure 7.16: Architecture of the algorithm for computing deviations on asymmetrical field sizes.

7.3.1 Computation of asymmetrical field sizes and deviations

Two functions were developed to compute the asymmetrical field size, one for the x-direction and one for the y-direction. The function for the x-direction plots a line parallel to the x-axis starting from the averaged radiation isocentre⁵ towards the field edges. The two intersections are computed as well as the distance between those intersections and the averaged radiation isocentre which yield x_1 and x_2 . The procedure for the y-direction is completely analogous. Assuming a $5 \times 5 \text{ cm}^2$ field the deviation on the asymmetrical field size for x_1 in millimetres is computed by subtracting x_1 from 25 mm. The resulting output image of the algorithm is eventually given in Figure 7.17.



Figure 7.17: The resulting output image after performing the asymmetrical field size algorithm.

⁵ The averaged radiation isocentre is discussed in section 6.3

7.4 Architecture of the GUI

The different algorithms discussed above are embedded in a user-friendly graphical user-interface (GUI) from which the architecture is given in Figure 7.19 and a screenshot of the GUI is given in Figure 7.18. Notice that the GUI contains as few buttons as possible to make it easy to use and to prevent user induced mistakes. The complete source code is given in appendix B.

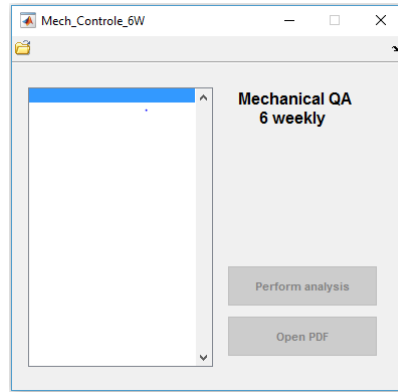


Figure 7.18: Screenshot of the graphical user-interface

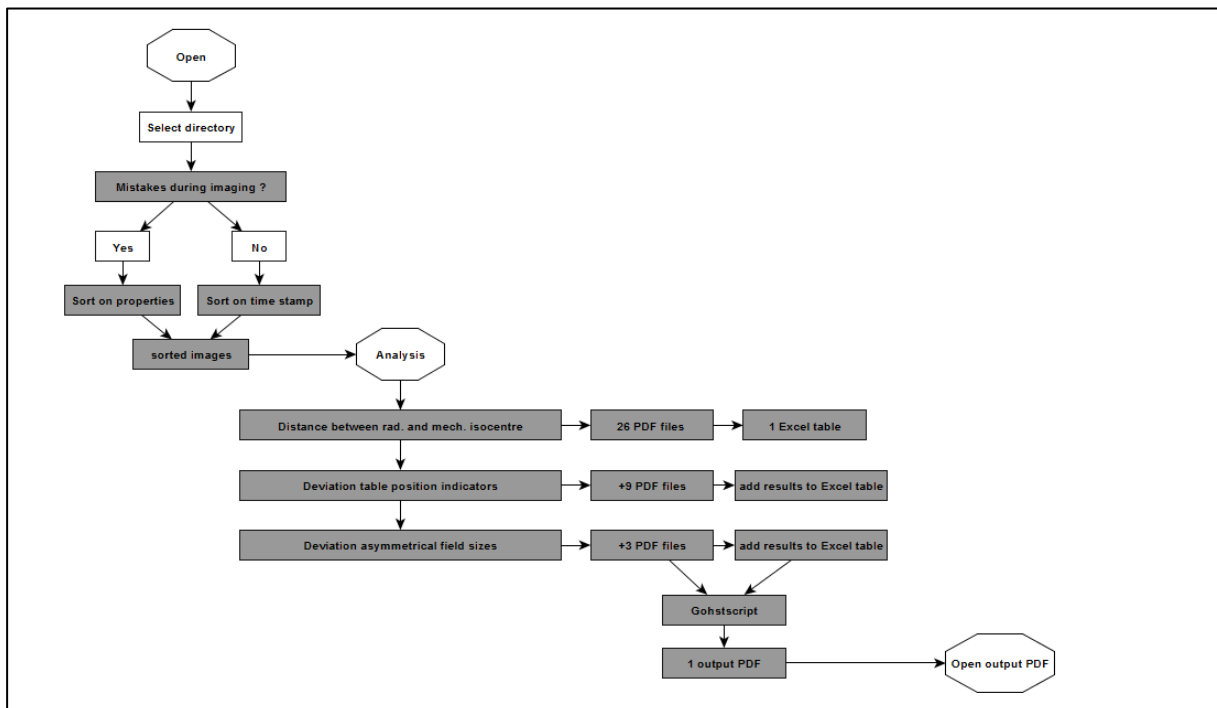


Figure 7.19: Architecture of the main file of the graphical user-interface

The octagons in Figure 7.19 represent the three push buttons in the graphical user-interface. All the white boxes and octagons indicate that the end user has to do a handling, in this case it refers to a mouse click. The grey boxes indicate processes performed by the programme which are not visible for the end user and do not need responses or handlings from the end user.

7.4.1 Open

The nineteen DICOM files that have to be analysed should all be located in one directory. When one clicks on the button with the map icon in the upper left corner of Figure 7.18 the programme will ask if there were any mistakes made during the acquisition of the pictures. When there were no mistakes made the images can easily be sorted based on the time stamp in the DICOM header since the measurements are always carried out in the same order following the protocol. In the other case, when mistakes were made, the images are sorted based on the combination of image properties like: gantry-, collimator-, table angles, longitudinal, lateral and vertical position. Once the directory paths to the images are sorted, this list is saved for further usage and the “perform analysis” button of Figure 7.18 is made active.

7.4.2 Perform analysis and output file

Once the “perform analysis” button is clicked, the m-file named analysis begins to run. The first step in this file is to divide the sorted image path list into 5 distinct groups. The first group consists of the first 13 images of the sorted list which, regarding to the protocol of L.O.C., have to be examined by the algorithm for the calculation of the distance between the radiation and mechanical isocentre. The second, third and fourth group each consists of one image, these are the vertical-, longitudinal- and lateral displaced images respectively. These have to be examined by the algorithm for computing the deviation on the table position indicators. The fifth and final group consists of the last three images of the sorted list which are the three asymmetrical field sizes that have to be analysed.

The first group of 13 images is implemented in a for-loop which the images one by one loads into the algorithm for the calculation of the distance between the radiation- and mechanical isocentre. This algorithm returns the found distance in millimetres, the place of the radiation isocentre, the radius and location of the detected circle and the four corner points of the field for visualization. For the first 4 images the radiation isocentre will be stored for averaging for the determination of the asymmetrical field sizes. Once the analysis is done a figure is created (invisible for end user) and the field, the diagonals and the detected circle together with its centre of mass are plotted. This figure is written to a PDF file. A detailed image on which there is zoomed in on the detected circle is also written to another PDF file. These images are meant to serve as a verification that analysis has been done correctly. Finally the gantry-, collimator- and table angles of the image are acquired from the DICOM header and are stored together with the found distance in an excel table. Each iteration of the for-loop therefore yields 2 PDF files. There are 13 iterations thus there will be 26 individual PDF files created at the end of this section of the analysis file.

The second, third and fourth group which contain respectively the vertical-, longitudinal- and lateral displaced images are loaded one by one (together with their reference images) in their corresponding algorithms for computing the deviation on the table position indicators. This algorithm returns the computed deviation in the displaced direction, the deviation in the direction perpendicular (in the same plane) to the displaced direction and the location of the two detected circles together with their radii. Again for each of the images a figure is created with the data plotted onto it. Each image yields three PDF files: one with the entire image and one detail image zoomed in on each of the two detected circles. The deviations are again written to the excel table.

The fifth and last group contains the three asymmetrical field size images and these are one by one loaded into the field size algorithm together with the average radiation isocentre by means of a for-loop. The algorithm returns the four asymmetrical field sizes, the total field size in two dimensions and the four corners of the field for visualization. Again the figure is created, data plotted and the figure is written to PDF. The results for the asymmetrical field sizes are written to the excel table.

When all the above groups are analysed the process has yielded 38 individual PDF files and one excel file. The excel table is first written to PDF and subsequently the 39 PDF files are concatenated into one single output PDF. The approach used to bring these files together into one single output PDF file was proposed by Michiel Darcis and Gert Leurs, fellow students ICT-electronics, and requires that the end user has got Ghostscript⁶ installed. The PDF files are concatenated using the function *appendPDFs* which is official courtesy of Oliver Woodford (2011). An example of a few pages of the output PDF file is given in appendix C.

7.5 Accuracy and reproducibility

7.5.1 Accuracy

The accuracy of the circle detection algorithm could be verified by displacing the iso-align a certain distance and measure this known displacement of the radio-opaque sphere using the software. Since accuracy should be tested on submillimetre or at least at millimetre level and in three dimensions (vertical, longitudinal and lateral). This means that the iso-align should be shifted for example 0,5 mm in longitudinal direction without applying any shift in the lateral direction. This could be done by using a micrometre but since the iso-align is a relatively big and unwieldy device this is not possible in reality without the risk of introducing errors. Another drawback of the iso-align device is that it cannot be adjusted in the vertical direction which makes it difficult to displace it vertically.

The accuracy was tested by applying a series of predefined shifts by means of displacing the treatment table. Since there is a certain uncertainty on the table position indicators these accuracy values should be taken with a grain of salt and are rather added to give an indication than to proof the accuracy of the algorithm.

Due to the above stated problems the accuracy of the algorithm could only be verified by comparing the values of the output file to these of the manual measurements carried out by the employees of L.O.C. This is done for CLINAC (full resolution) and Truebeam (half resolution) devices separately.

7.5.2 Reproducibility

The reproducibility of the circle detection algorithm and the algorithm for the detection of the radiation isocentre was tested by acquiring repeated images ($N = 8$) of the iso-align without altering the setup or position of the iso-align device. The displacement of the centroid of the detected circle and the displacement of the radiation isocentre both should be zero since no shifts were applied. This is done for both CLINAC and Truebeam devices, deviation were calculated in millimetre together with the standard deviation (also in mm) and are summarized in the next section.

⁶ Ghostscript is freely downloadable at: www.ghostscript.com

8 Results and discussion

8.1 Algorithm timing considerations

The time needed to estimate the distance between the radiation- and mechanical isocentre amounts 5,418 seconds for one single image. The algorithm for calculating the deviation on the table position indicators takes 0.918 seconds to analyse one image. The algorithm for calculating the deviation on the asymmetrical field sizes needs 6,850 seconds. The reason why the second algorithm above is much faster than the other two is because this algorithm does not compute any field edges. The field edge algorithm is the most time intensive part of the programme with a time of 4,333 seconds because it has to examine more than 600 intensity profiles per image. Finally, the whole process starting from opening the GUI until the creation of the output file takes 2 minutes and 45 seconds. Comparing this to the time it takes to analyse the 19 images manually (20 to 30 minutes) this is a considerable improvement in timing considerations.

8.2 Accuracy

8.2.1 Clinac

As denoted in section 7.5.1 the accuracy of the circle detection method was tested by displacing the iso-align several known distances and by subsequently estimating these shifts with the software. The iso-align was displaced using the treatment table which has an uncertainty on the position indicators of about ± 1 mm on 15 cm and therefore the results should be taken with a grain of salt. The results are given in Table 8.1 and are rather added to give an overall indication than to proof the integrity of the algorithm. The results are given for each dimension separately in the form of the mean value of the absolute values of the deviation on the known displacement in millimetres plus/minus the standard deviation in millimetres, the maximum deviation in millimetres is also shown. These results are acquired using a CLINAC device and therefore delivered full resolution images. In each dimension 6 different displacements where applied ($N = 6$).

Table 8.1: Results of testing the accuracy of the circle detection algorithm using a known displacement of the treatment table of the CLINAC. Note that the treatment table position indicators have an uncertainty of ± 2 mm.

	Lateral	Longitudinal	Vertical
Mean deviation \pm SD	0,57 \pm 0,2 mm	0,81 \pm 0,4 mm	0,37 \pm 0,2 mm
Max deviation	0,76 mm	1,44 mm	0,53 mm

The displacements used to obtain the results of Table 8.1 where ± 1 mm, ± 2 mm in only one direction and ± 1 mm in both dimensions of the imaged plane. With an uncertainty of ± 2 mm on 15 cm on the table position indicators these results are not representative for proving accuracy but they can give an indication that the circle detection algorithm fulfils its task with a sufficient amount of accuracy. The mean deviations in Table 8.1 are all under 0.81 mm which are reasonable results taking into account the uncertainty on the applied shift. The maximal deviation in the longitudinal direction is a bit high but could possibly be ascribed to the uncertainty on the table position indicators.

Because the above results are meaningless speaking about accuracy, the accuracy of the software was also tested by means of a comparison with manual results obtained by the employees of L.O.C. The results of the mechanical QA of 29/04/2017 were compared with the software measurements for a CLINAC device and the deviations of the manual vs. software measurements are given in Figure 8.1 to Figure 8.3.

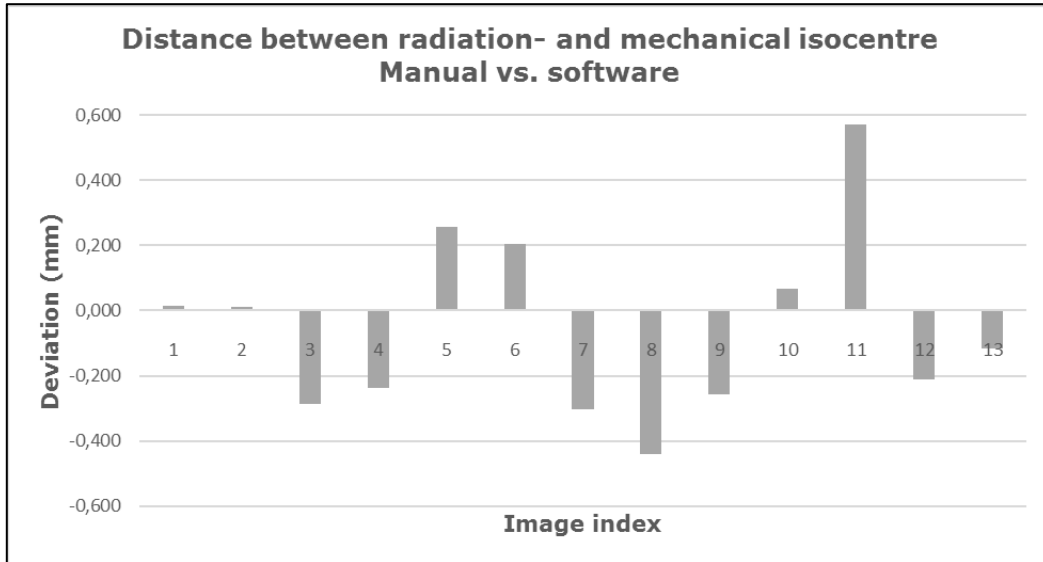


Figure 8.1: Comparison of the results of measuring the distance between the radiation- and mechanical isocentre manually vs. with the software for a CLINAC. The image indices are the same as those in Table 6.1.

Looking at the results in Figure 8.1 the deviation on the calculation of the distance between the radiation- and mechanical isocentre with respect to the manually measured values are almost all within $\pm 0,500$ mm. Two values exceed this boundary with one of them being the maximum deviation of 0,572 mm. Finally the mean deviation in absolute values is 0,229 mm, which is a considerably good result.

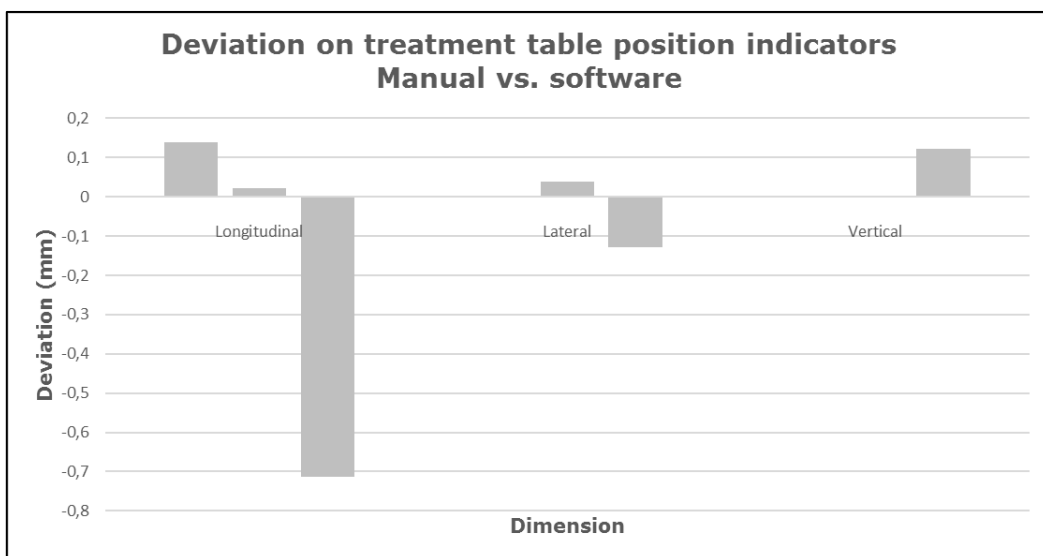


Figure 8.2: Comparison of the results of measuring the deviation on the table position indicators manually vs. with the software for a CLINAC.

The deviations for the comparison of the manual vs. software measurements for the calculation of the deviation on treatment table position of Figure 8.2 show that all, but one, deviations are within $\pm 0,200$ mm. The one outlier defines the maximum deviation which amounts 0,714 mm. Finally the mean deviation in absolute values is 0,194 mm. These results are again very acceptable regarding that the manual measurements are very user dependent.

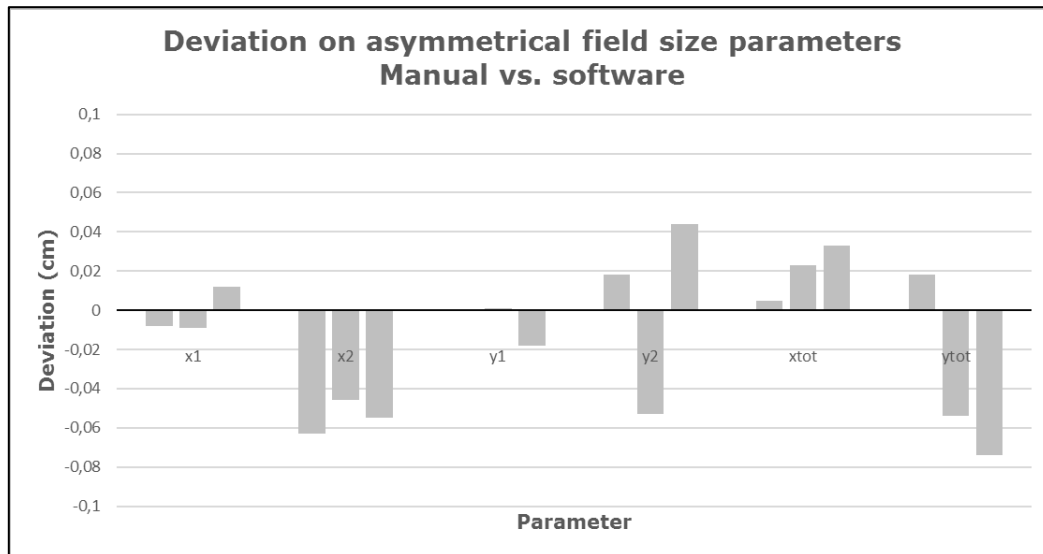


Figure 8.3: Comparison of the results of measuring the deviation on asymmetrical field size parameters manually vs. with the software for a CLINAC. The first bar for each parameter denoted the results for the 5x5 cm² field. The second and third bar of each parameter denote the results for the 10x10 and 18x18 cm² fields respectively.

The comparison of the manual vs. software measurements for the determination of the deviation on asymmetrical field size parameters in Figure 8.3 shows that all deviations for the asymmetrical field size parameters are within $\pm 0,500$ mm. On the other hand all deviations for the symmetrical (total) field size parameters are within $\pm 0,800$ mm. The mean deviation in absolute values for the asymmetrical field size parameters amounts 0,273 mm with a maximum deviation in absolute values of 0,630 mm. For the symmetrical field size parameters the mean deviation amounts 0,345 mm with a maximum deviation of 0,740 mm, both calculated using absolute values.

8.2.2 Truebeam

The accuracy of the software as a whole was also tested for a Truebeam device, with a half resolution licensed EPID, by comparing the output files for a certain periodic QA analysis with the manually obtained results by the employees of L.O.C. The results for the comparison of the results of the mechanical QA of 05/04/2017 for the Truebeam device are given in Figure 8.4 to Figure 8.6.

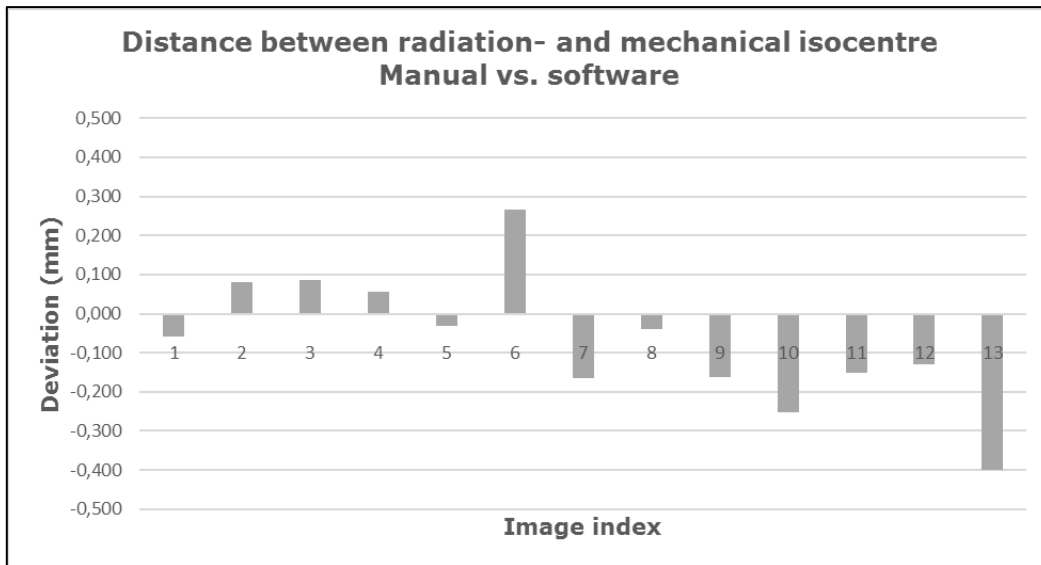


Figure 8.4: Comparison of the results of measuring the distance between the radiation- and mechanical isocentre manually vs. with the software for the Truebeam.

Looking at the results in Figure 8.4 the deviations on the computation of the distance between the radiation- and mechanical isocentre with respect to the manually measured values are all between $\pm 0,400$ mm. The mean deviation in absolute values amounts 0,146 mm with a maximum in absolute value of 0,400 mm. These are more than acceptable results taking into account that the same manual measurement performed by two different persons can easily deviate $\pm 0,2$ mm. These values are also in line with the results obtained for the CLINAC and are even slightly better. This is against expectations because one could think that a better resolution would yield an increase in accuracy.

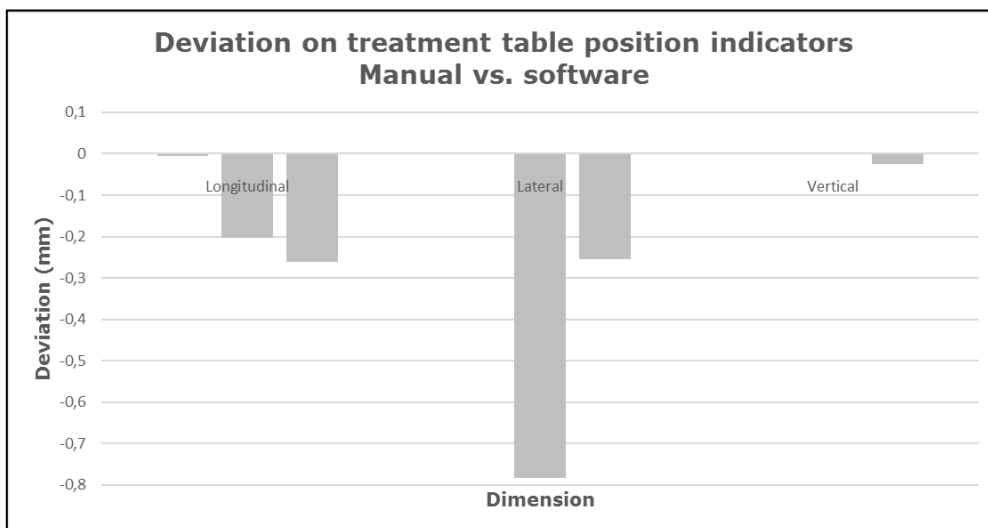


Figure 8.5: Comparison of the results of measuring the deviation on the table position indicators manually vs. with the software for a Truebeam.

The results in Figure 8.5 for the deviation on the treatment table position indicators shows that most of the deviations are within $\pm 0,300$ mm with the exception of the deviation of $-0,784$ mm, which also immediately is the maximum deviation. The mean deviation in absolute values is $0,255$ mm. These results are also very reasonable although they are slightly worse than the results for the CLINAC.

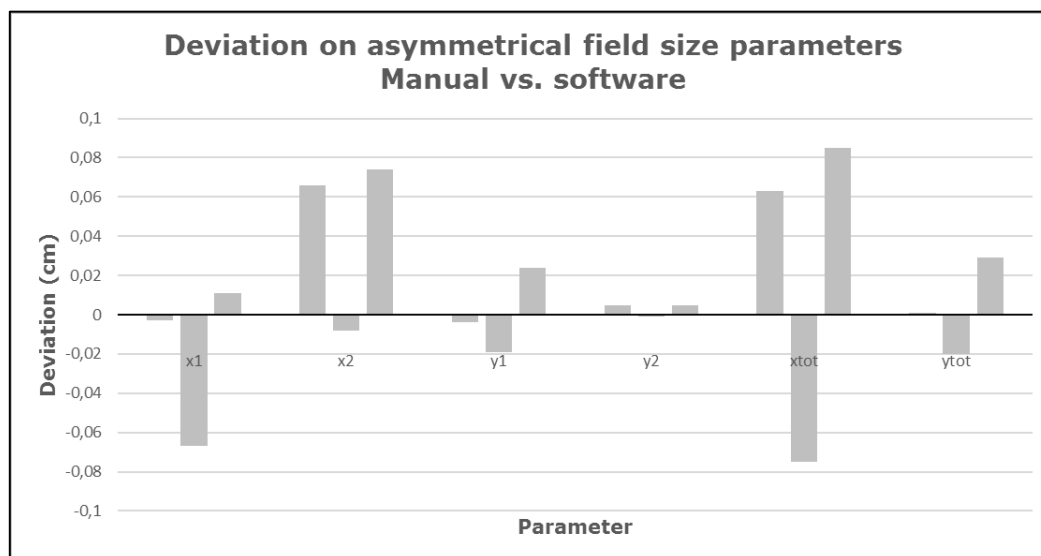


Figure 8.6: Comparison of the results of measuring the deviation on asymmetrical field size parameters manually vs. with the software for a Truebeam. The first bar for each parameter denoted the results for the 5×5 cm² field. The second and third bar of each parameter denote the results for the 10×10 and 18×18 cm² fields respectively.

The results for the deviation on the asymmetrical field size parameters of Figure 8.6 show that all the deviations are within $\pm 0,900$ mm. The first bar of each parameter in Figure 8.6 denotes the results for the 5×5 cm² field while the second and third bar of each parameter denote the results for the 10×10 cm² and 18×18 cm² field respectively. The mean deviation on the asymmetric parameters amounts $0,240$ mm with a maximum deviation of $0,744$ mm. Finally the mean deviation on the symmetrical total field size is $0,468$ mm with a maximum deviation of $0,852$ mm. Comparing these results to those obtained for a CLINAC the asymmetric parameters are slightly more accurately computed for a Truebeam, while the results for the symmetrical parameters are better for the CLINAC.

Since the results of the comparison of the manual vs. software measurements do not deviate significantly comparing CLINAC and Truebeam (and thus, in this case, full- and half resolution images) it is safe to say that the software has more or less the same accuracy for both devices. Therefore the effect of using half- or full resolution licensed EPIDs on the resolution is negligible, which is, like earlier denoted, not in line with the expectations.

8.3 Reproducibility

8.3.1 Clinac

The results of the reproducibility testing for the CLINAC device with a full resolution licensed EPID are summarized in Table 8.2 and Table 8.3. The results are shown in the form of the mean deviation in reproducibility plus minus the standard deviation (SD) together with the maximum deviation in absolute value.

Table 8.2: The mean deviation, SD and maximum deviation in reproducibility in performing the circle detection algorithm using a CLINAC linear accelerator ($N=8$).

	Reproducibility circle detection - CLINAC		
	Lateral	Longitudinal	Vertical
Mean deviation \pm SD	0,144 \pm 0,030 mm	0,051 \pm 0,037 mm	0,074 \pm 0,034 mm
Max. deviation	0,169 mm	0,133 mm	0,124 mm

Table 8.3: The mean deviation, SD and maximum deviation in reproducibility in computing the radiation isocentre using a CLINAC linear accelerator ($N=8$).

	Reproducibility radiation isocentre detection - CLINAC		
	Lateral	Longitudinal	Vertical
Mean deviation \pm SD	0,069 \pm 0,031 mm	0,059 \pm 0,026 mm	0,008 \pm 0,006 mm
Max. deviation	0,115 mm	0,096 mm	0,013 mm

The capacity to obtain the same locations correctly on repeated images for both the circle- and radiation isocentre detection algorithm is, according to Table 8.2 and Table 8.3, very appreciable. Finally, the overall mean deviations in reproducibility in all three dimensions are 0,080 mm \pm 0,051 mm and 0,053 mm \pm 0,034 mm for the circle- and radiation isocentre detection respectively. The radiation isocentre detection based on the field edge algorithm⁷ appears to have the better ability to locate the same positions on reproduced images for the Varian CLINAC accelerator. This also confirms the decision to reject the alternative field edge algorithm of section 7.1.7 because it relies on the same function *regionprops*⁸ as the circle detection algorithm.

⁷ The field edge algorithm is discussed in section 7.1.2.

⁸ The function *regionprops* is discussed in section 7.1.5.

8.3.2 Truebeam

The results of the reproducibility testing for the Varian Truebeam accelerator are represented in the same fashion as those of the CLINAC. These results, shown in Table 8.4 and Table 8.5, are more or less equivalent to these of the CLINAC and show even a slightly better performance in reproducibility for the circle detection algorithm. This is not in line with the expectations because the images acquired from the Truebeam have only half the resolution of the images acquired at a CLINAC. This means the circle detection algorithm has 4 times as much pixels on a CLINAC image over which it can compute the centre of mass of the circle in comparison to a Truebeam image. In other words, one could expect that the circle detection algorithm has a better reproducibility for CLINAC images in comparison to Truebeam images.

Table 8.4: The mean deviation, SD and maximum deviation in reproducibility in performing the circle detection algorithm using a Truebeam linear accelerator (N=8).

Reproducibility circle detection - Truebeam			
	Lateral	Longitudinal	Vertical
Mean deviation ± SD	0,040 ± 0,046 mm	0,050 ± 0,042 mm	0,030 ± 0,040 mm
Max. deviation	0,080 mm	0,089 mm	0,081 mm

Table 8.5: The mean deviation, SD and maximum deviation in reproducibility in computing the radiation isocentre using a Truebeam linear accelerator (N=8).

Reproducibility radiation isocentre detection - Truebeam			
	Lateral	Longitudinal	Vertical
Mean deviation ± SD	0,088 ± 0,015 mm	0,014 ± 0,010 mm	0,019 ± 0,010 mm
Max. deviation	0,100 mm	0,027 mm	0,023 mm

The overall mean deviations in reproducibility in all three dimensions for Truebeam devices are 0,049 mm ± 0,041 mm and 0,037 ± 0,038 mm for the circle- and radiation isocentre detection respectively. Both values are better than those for CLINAC devices which is, like already denoted, against the expectations. But again the capacity to obtain the exact same location on repeated images is better for the radiation isocentre detection algorithm.

9 Conclusions

The application developed for the evaluation of the images acquired during the periodic mechanical QA procedure needs 2 minutes and 45 seconds to finish this analysis. The time needed to perform the same analysis manually is 20 – 30 minutes which implies an improvement in timing considerations of, at least, 1000 %.

The accuracy of the three main algorithms was evaluated by comparing the software measurements with those measured manually. The full- and half resolution images were acquired using a Varian Truebeam and CLINAC accelerator respectively. The computation of the distance between the radiation- and mechanical isocentre using the software deviates averagely 0,229 mm and 0,146 mm from those acquired manually for full- and half resolution images respectively. Subsequently, the mean deviations of the measurements for the computation of the deviations on the table position indicators amount 0,194 mm and 0,255 mm, again for full- and half resolution images respectively. Furthermore, the asymmetrical field size parameters of square, open fields of full resolution images can be computed within 0,273 mm (average value) of the manually measured value. For images with half resolution this mean deviation amounts 0,240 mm.

The circle detection algorithm used for the detection of the mechanical isocentre has a mean reproducibility in three dimensions of $0,080 \text{ mm} \pm 0,051 \text{ mm}$ regarding full resolution images. Analysing half resolution images with the same algorithm shows a mean reproducibility of $0,049 \text{ mm} \pm 0,041 \text{ mm}$ (averaged over all three dimensions). Finally, the radiation isocentre can be detected with a mean reproducibility, in three dimensions, of $0,053 \pm 0,034 \text{ mm}$ and $0,037 \pm 0,038 \text{ mm}$ for full- and half resolution images respectively.

Since there were no significant differences in performance between half- and full resolution images for the accuracy- as well as the reproducibility tests the computations are assumed to be independent of the resolution in the range of $[384 \times 512 ; 768 \times 1024]$. The slightly better performance measuring half resolution images (acquired with Varian Truebeam) is a little strange but can be due to the fact that the Truebeam devices are more recent and accurate in comparison to the Varian CLINAC devices.

10 References

- [1] *Quality assurance in radiotherapy*, 1st ed. Genf: WHO, 1988, pp. 3-36.
- [2] W. Parker and H. Patrocínio, *Chapter 7: Clinical treatment planning in external photon beam radiotherapy*, Heidelberg, 2006.
- [3] E. Podgorsák, *Radiation physics for medical physicists*, 2d ed, Berlin: Springer, 2010, pp. 277-384.
- [4] E. Podgorsak, *Radiation oncology physics*, 2d ed. Vienna: International Atomic Energy Agency, 2005, pp. 123-271.
- [5] N. Juntong, K. Pharaphan, S. Light, and N. Ratchasima, "THE OPTIMIZED X-RAY TARGET OF ELECTRON LINEAR ACCELERATOR FOR RADIOTHERAPY," pp. 1-3, 1933.
- [6] B. M. Prendergast *et al.*, "Flattening filter-free linac improves treatment delivery efficiency in stereotactic body radiation therapy.," *J. Appl. Clin. Med. Phys.*, vol. 14, no. 3, pp. 4126, 2013.
- [7] V. M. Tello, "Medical Linear Accelerators and how they work," Florida Hosp. Cancer Inst. Kissimmee, 2014.
- [8] M. M. Mesbahi A, Dadgar H, Ghareh-Aghaji N, "Monte Carlo approach to lung dose calculation in small fields used in intensity modulated radiation therapy and stereotactic body radiation therapy.," *Cancer J.*, vol. 10, no. 4, pp. 896-902, 2014.
- [9] G. Kutcher *et al.*, "Report of AAPM Radiation Therapy Committee Task Group 40", *Med. Phys.*, vol. 21, no. 4, pp. 581-618.
- [10] E. E. Klein *et al.*, "Task Group 142 report: Quality assurance of medical accelerators," *Med. Phys.*, vol. 36, no. 9, p. 4197, 2009.
- [11] G. Meijer, H. Kleffens and B. Mijneer, *Quality control of medical linear accelerators*, 1st ed. Bilthoven: Nederlandse Commissie voor Stralingsdosimetrie, 1996.
- [12] "GAFChromic™ EBT3 film specifications". [Online]. Available at www.gafchromic.com [Accessed: 02- Apr- 2017].
- [13] K. A. Langmack, "Review article Portal imaging", *The British Journal of Radiology*, vol. 74, 2001, pp. 789-804.
- [14] H. Takei *et al.*, "Response of electric portal imaging devices to energy spectrum of therapeutic photons", Scientific Exhibit, Keio Univeristy Hospital, 2014.
- [15] Meditronix Corporation. "Fixed Lasers, Iso Align Device", 2008. [Online]. Available: <http://www.meditronixindia.com/Dosimetry23-FixedLaser-IsoAlignDevice.htm>. [Accessed: 05- Apr- 2017].
- [16] Civcort Radiotherapy. "Iso-Align™", 2017. [Online]. Available: <http://civcort.com/ro/physics-qa/isoalign/isoalign-MTIAD1.htm>. [Accessed: 05- Apr- 2017].
- [17] S. Feenstra, *Kwaliteitscontrole van medische lineaire versnellers*, 1st ed., Bilthoven: Nederlandse Commissie voor Stralingsdosimetrie, 1995.
- [18] R. (ACR), "Image-Guided Radiation Therapy", 2016. [Online]. Available: <https://www.radiologyinfo.org/en/info.cfm?pg=igrt>. [Accessed: 06- Apr- 2017].
- [19] Wiener Krankenanstaltenverbund, "Diameter of the radiation isocentre", 2015. [Online]. Available: http://www.wienkav.at/kav/kfj/91033454/physik/as500/aS500_sphere.htm. [Accessed: 06- Apr- 2017].

- [20] P. Ravindran, "A study of Winston–Lutz test on two different electronic portal imaging devices and with low energy imaging", *Australasian Physical & Engineering Sciences in Medicine*, vol. 39, no. 3, pp. 677-685, 2016.
- [21] E. E. Klein *et al*, "A quality assurance program for ancillary high technology devices on a dual-energy accelerator," *Radiotherapy and Oncology*, vol. 38, p.51-60, 1995..
- [22] Wiener Krankenanstaltenverbund, "Split-field test", 2015. [Online]. Available: http://www.wienkav.at/kav/kfj/91033454/physik/as500/aS500_split.htm. [Accessed: 06-Apr- 2017].
- [23] O. Pianykh, *Digital Imaging and Communications in Medicine (DICOM)*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 3-167.
- [24] R. Gonzalez, R. Woods and S. Eddins, *Digital image processing using MATLAB®*, 1st ed. United States: Gatesmark Publishing, 2010.
- [25] Mathworks United Kingdom, "Image Coordinate Systems", 2017. [Online]. Available: <https://nl.mathworks.com/help/images/image-coordinate-systems.html>. [Accessed: 07-Apr- 2017].
- [26] "SPATIAL FILTER", 2015. [Online]. Available: <https://www.slideshare.net/shaletks/spatial-filter-47299769>. [Accessed: 07- Apr- 2017].
- [27] "Histograms and contrast", 2012. [Online]. Available: <http://imgprocessing.tk/intro/histograms.html>. [Accessed: 07- Apr- 2017]
- [28] R. Fischer *et al*, "Morphology - Dilation", 2003. [Online]. Available: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm>. [Accessed: 08- Apr- 2017].
- [29] Danyal, "spatial filtering", 2013. [Online]. Available: <https://pt.slideshare.net/lineking/06-spatial-filtering-dip/16>. [Accessed: 08- Apr- 2017].
- [30] M. Béla, "Spatial Analysis 3", 2010. [Online]. Available: http://www.tankonyvtar.hu/en/tartalom/tamop425/0027_SAN3/ch01s03.html. [Accessed: 08- Apr- 2017].
- [31] "Measure properties of image regions - MATLAB regionprops - MathWorks United Kingdom", *MathWorks*, 2017. [Online]. Available: <https://nl.mathworks.com/help/images/ref/regionprops.html>. [Accessed: 25- Apr- 2017].

11 Appendices

11.1 Appendix A: periodically mechanical QA protocol of L.O.C.⁹

Protocol: Mechanische Controle (6-wekelijks)


Deel 1: In bunker


Begin de metingen met parameters $G = 0^\circ$, $C = 0^\circ$, $T = 0^\circ$. Zet de jaws volledig open en stel het Iso-Align toestel in op de reticel markeringen en $SSD = 100$ cm. Leg een 2^e lock bar onder het Iso-Align en controleer met de waterpas of het Iso-Align oppervlak parallel staat met de tafel.

 Vul in: uitlezing met waterpas op $G = 0^\circ$

 Kijk het lichtveld en reticel alignment na. De markeringen van het reticel mogen niet afbuigen op het Iso-Align toestel.

Draai gantry naar $G = 90^\circ$.

 Vul in: uitlezing met waterpas op $G = 90^\circ$

 Vul in: uitlezing met waterpas op $C = 0^\circ$

Draai collimator naar $C = 90^\circ$ en 270° .


 Vul in: uitlezing met waterpas op $C = 90^\circ$

 Vul in: uitlezing met waterpas op $C = 270^\circ$


Draai collimator terug naar $C = 0^\circ$. Draai nu ook het oppervlak van het Iso-Align toestel loodrecht en stel de hoogte in op de longitudinale reticel markering. Draai vervolgens de gantry naar $G = 270^\circ$ en middel de hoogte uit.

 Vul in: uitlezing met waterpas op $G = 270^\circ$

Draai gantry terug naar $G = 315^\circ$.

 Kijk lasers na (links, rechts, sagittaal) op het Iso-Align toestel

Draai het oppervlak van het Iso-Align toestel weer parallel met de tafel.

 Kijk lasers na (top) op het Iso-Align toestel

Draai gantry verder naar $G = 0^\circ$.

 Vul in: SSD na uitmiddelen

Deel 2: Aan bediening

Roep de patiënt "Mechanische Controle" op. Neem MV single exposures in QA. Doe indien nodig een override op de tafelparameters.

 $G = 0^\circ$, $C = 0^\circ$, $T = 0^\circ$, $X_{MLC} = 15$ cm, $Y_{MLC} = 15$ cm (*)¹⁰


Controleer of er geen distortie van het beeld gebeurd is door een gekende afstand op het Iso-Align toestel te meten. Meet de afstand tussen de bolletjes die een 10×10 cm² veld aanduiden.


 Vul in: gemeten lengte van een in werkelijkheid 10 cm lang lijnstuk op het Iso-Align toestel


⁹ This protocol is original courtesy of L.O.C.


¹⁰ Dit beeld wordt verderop in de metingen opnieuw gebruikt door aanduiding van (*)

Voer een 2D matching uit in "Offline Review", namelijk: zet de field edge (= blauwe rechthoek) op de randen van het gestraalde veld → Finish . Gebruik de functie "Draw a point" om een punt te plaatsen op het groene stralings isocenter (= verschoven graticule centrum). Dit punt zal verderop in de metingen gebruikt worden. Ga opnieuw naar 2D matching en druk op "reset anatomy" → Finish. Meet vervolgens de rechtstreekse afstand van het stralings isocenter tot het mechanisch isocenter (= bolletje).

 *Vul in: afstand van stralings isocenter tot mechanisch isocenter in referentiecondities*


 **G = 0°, C = 90°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


 **G = 0°, C = 165°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**

 **G = 0°, C = 270°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


Voer op elk van deze drie beelden een 2D matching uit zoals beschreven bij de eerste test. Vanaf nu is het niet langer nodig een punt te plaatsen. Meet de rechtstreekse afstand van het groene stralings isocenter tot het mechanisch isocenter.

 *Vul in: afstanden van stralings isocenter tot mechanisch isocenter op variërende collimatorhoeken*

 **G = 0°, C = 0°, T = 90°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


 **G = 0°, C = 0°, T = 270°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**

Voer op elk van deze twee beelden een 2D matching uit zoals beschreven bij de eerste test. Meet de rechtstreekse afstand van het groene stralings isocenter tot het mechanisch isocenter.

 *Vul in: afstanden van stralings isocenter tot mechanisch isocenter op variërende tafelhoeken*

Trek vervolgens een lijn door een rij bolletjes (verticaal of horizontaal) in het midden van het beeld. Kijk of deze lijn parallel loopt met het stralingsveld (verschoven graticule). Indien niet, meet de hoek tussen deze twee lijnen.

 *Vul in: afwijking in hoek van het Iso-Align toestel ten opzichte van het stralingsveld*

 **G = 180°, C = 0°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


Voer een 2D matching uit zoals beschreven bij de eerste test. Meet de rechtstreekse afstand van het groene stralings isocenter tot het mechanisch isocenter.


 *Vul in: Afstand van stralings isocenter tot mechanisch isocenter op G = 180°*


Draai in de bunker het oppervlak van het Iso-Align toestel loodrecht.

 *Vul in: uitlezing met waterpas op G = 180°*


Deel 3: Aan bediening


 **G = 90°, C = 90°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**

 **G = 90°, C = 0°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


 **G = 90°, C = 270°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**

Voer op elk van deze drie beelden een 2D matching uit zoals beschreven bij de eerste test. Meet de rechtstreekse afstand van het groene stralings isocenter tot het mechanisch isocenter.


 *Vul in: Afstanden van stralings isocenter tot mechanisch isocenter op variërende gantryhoeken*

 **G = 270°, C = 270°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**


 **G = 270°, C = 0°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm (**)¹¹**

 **G = 270°, C = 90°, T = 0°, X_{MLC} = 15 cm, Y_{MLC} = 15 cm**

Voer op elk van deze drie beelden een 2D matching uit zoals beschreven bij de eerste test. Meet de rechtstreekse afstand van het groene stralings isocenter tot het mechanisch isocenter.

 *Vul in: Afstanden van stralings isocenter tot mechanisch isocenter op variërende gantryhoeken*

Zet tafel los en verplaats deze +15 cm verticaal.


 **G = 270°, C = 0°, T = 0°, X_{MLC} = 20 cm, Y_{MLC} = 20 cm.**

¹¹ Dit beeld wordt verderop in de metingen opnieuw gebruikt door aanduiding van (**)


Gebruik de functie "Compare with" in "Offline Review" om dit beeld te vergelijken met het beeld voordat de tafel verschoven werd op $G = 270^\circ$ (**). Verschuif de beelden zodat de overeenkomstige bolletjes op mekaar liggen. Meet de verticale verschuiving van het graticule (= verplaatsing van de tafel).

 *Vul in: verticale verplaatsting van de tafel*


Indien er naast een verticale ook een longitudinale verschuiving te zien tussen beide graticules betekent dit dat de tafel niet loodrecht bewogen heeft. De longitudinale verschuiving moet kleiner zijn dan 0.26 cm om binnen de tolerantie van 1° afwijking te blijven.

 *Vul in: longitudinale afwijking van tafelhoek na verticale verplaatsing*

Zet tafel los en verplaats deze eerst terug -15 cm verticaal en vervolgens +15 cm longitudinaal. Draai het oppervlak van het Iso-Align toestel parallel met de tafel.

 **$G = 0^\circ, C = 0^\circ, T = 0^\circ, X_{MLC} = 20 \text{ cm}, Y_{MLC} = 20 \text{ cm}$**

Gebruik de functie "Compare with" in "Offline Review" om dit beeld te vergelijken met het beeld voordat de tafel verschoven werd op $G = 0^\circ$ (*). Verschuif de beelden zodat de overeenkomstige bolletjes op mekaar liggen. Meet de longitudinale verschuiving van het graticule (= verplaatsing van de tafel).

 *Vul in: longitudinale verplaatsting van de tafel*


Indien er naast een longitudinale ook een laterale verschuiving te zien tussen beide graticules betekent dit dat de tafel niet loodrecht bewogen heeft. De laterale verschuiving moet kleiner zijn dan 0.26 cm om binnen de tolerantie van 1° afwijking te blijven.

 *Vul in: laterale afwijking van tafelhoek na longitudinale verplaatsing*


Zet tafel los en verplaats deze eerst terug -15 cm longitudinaal en vervolgens +15 cm lateraal.

 **$G = 0^\circ, C = 0^\circ, T = 0^\circ, X_{MLC} = 20 \text{ cm}, Y_{MLC} = 20 \text{ cm}$**


Gebruik de functie "Compare with" in "Offline Review" om dit beeld te vergelijken met het beeld voordat de tafel verschoven werd op $G = 0^\circ$ (*). Verschuif de beelden zodat de overeenkomstige bolletjes op mekaar liggen. Meet de laterale verschuiving van het graticule (= verplaatsing van de tafel).


 *Vul in: laterale verplaatsting van de tafel*


Indien er naast een laterale ook een longitudinale verschuiving te zien tussen beide graticules betekent dit dat de tafel niet loodrecht bewogen heeft. De longitudinale verschuiving moet kleiner zijn dan 0.26 cm om binnen de tolerantie van 1° afwijking te blijven.

 *Vul in: longitudinale afwijking van tafelhoek na laterale verplaatsing*


Schuif de tafel/Iso-Align toestel volledig weg van de gantry.


 **$G = 0^\circ, C = 0^\circ, T = 0^\circ, X_{jaws} = 5 \text{ cm}, Y_{jaws} = 5 \text{ cm}$**


 **$G = 0^\circ, C = 0^\circ, T = 0^\circ, X_{jaws} = 10 \text{ cm}, Y_{jaws} = 10 \text{ cm}$**

 **$G = 0^\circ, C = 0^\circ, T = 0^\circ, X_{jaws} = 18 \text{ cm}, Y_{jaws} = 18 \text{ cm}$**

Gebruik de functie "Compare with" in "Offline Review" om elk van deze drie beelden te vergelijken met het referentiebeeld uit test 1 (*). Deze beelden vind je onder "session timeline". Zet het onderste window/level van elk van deze drie beelden op 50% van de dosis, zodat de effectieve veldgrootte overblijft. Gebruik nu het punt (stralings isocenter) dat in test 1 geplaatst werd. Meet nu de afstanden van het stralings isocenter tot aan de veldgrenzen in vier richtingen.


 *Vul in: afstand van stralings isocenter tot X_1, X_2, Y_1 en Y_2 voor een veldgrootte van $5 \times 5 \text{ cm}^2$*


 *Vul in: afstand van stralings isocenter tot X_1, X_2, Y_1 en Y_2 voor een veldgrootte van $10 \times 10 \text{ cm}^2$*


 *Vul in: afstand van stralings isocenter tot X_1, X_2, Y_1 en Y_2 voor een veldgrootte van $20 \times 20 \text{ cm}^2$*


Deel 4: Aan bediening

Roep nieuwe patiënt “Aansluiting Controle” op om aansluitingen te controleren. Neem hiervoor integrated images met EPID SSD = 100 cm.

 Kwadrant 1: $G = 0^\circ, C = 0^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$

 Kwadrant 2: $G = 0^\circ, C = 0^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$

 Kwadrant 3: $G = 0^\circ, C = 0^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$

 Kwadrant 4: $G = 0^\circ, C = 0^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


Creëer in “Portal Dosimetry” een samengesteld beeld (= Create Composite Image) van deze vier beelden.


Trek dosisprofielen door het samengestelde beeld en meet de dosisverschillen ter hoogte van elk van de vier aansluitingen.


 Vul in: dosisverschillen ($\Delta_{12}, \Delta_{23}, \Delta_{34}, \Delta_{41}$) bij over- of onderdosage ter hoogte van de aansluitingen


Uitbreiding “Aansluiting Controle” (jaarlijks):

De 6 wekelijkse mechanische controle wordt 1x/jaar uitgebreid met extra velden om aansluitingen extra te controleren. Verifieer op het metingen overzicht of deze metingen moeten gebeuren.

 Kwadrant 1: $G = 0^\circ, C = 90^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 2: $G = 0^\circ, C = 90^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


 Kwadrant 3: $G = 0^\circ, C = 90^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 4: $G = 0^\circ, C = 90^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


Creëer in “Portal Dosimetry” een samengesteld beeld (= Create Composite Image) van deze vier beelden.


Trek dosisprofielen door het samengestelde beeld en meet de dosisverschillen ter hoogte van elk van de vier aansluitingen.

 Vul in: dosisverschillen ($\Delta_{12, c90}, \Delta_{23, c90}, \Delta_{34, c90}, \Delta_{41, c90}$) bij over- of onderdosage ter hoogte van de aansluitingen

 Kwadrant 1: $G = 90^\circ, C = 0^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 2: $G = 90^\circ, C = 0^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


 Kwadrant 3: $G = 90^\circ, C = 0^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 4: $G = 90^\circ, C = 0^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


Creëer in “Portal Dosimetry” een samengesteld beeld (= Create Composite Image) van deze vier beelden.


Trek dosisprofielen door het samengestelde beeld en meet de dosisverschillen ter hoogte van elk van de vier aansluitingen.

 Vul in: dosisverschillen ($\Delta_{12, g90}, \Delta_{23, g90}, \Delta_{34, g90}, \Delta_{41, g90}$) bij over- of onderdosage ter hoogte van de aansluitingen

 Kwadrant 1: $G = 90^\circ, C = 90^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 2: $G = 90^\circ, C = 90^\circ, T = 0^\circ, X_1 = 7.5 \text{ cm}, X_2 = 0.0 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$


 Kwadrant 3: $G = 90^\circ, C = 90^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 7.5 \text{ cm}, Y_2 = 0.0 \text{ cm}$


 Kwadrant 4: $G = 90^\circ, C = 90^\circ, T = 0^\circ, X_1 = 0.0 \text{ cm}, X_2 = 7.5 \text{ cm}, Y_1 = 0.0 \text{ cm}, Y_2 = 7.5 \text{ cm}$

Creëer in “Portal Dosimetry” een samengesteld beeld (= Create Composite Image) van deze vier beelden.

Trek dosisprofielen door het samengestelde beeld en meet de dosisverschillen ter hoogte van elk van de vier aansluitingen.

 Vul in: dosisverschillen ($\Delta_{12, g90, c90}, \Delta_{23, g90, c90}, \Delta_{34, g90, c90}, \Delta_{41, g90, c90}$) bij over- of onderdosage ter hoogte van de aansluitingen

 Kwadrant 1: $G = 90^\circ, C = 0^\circ, T = 0^\circ, 6MV$

 Kwadrant 2: $G = 90^\circ, C = 0^\circ, T = 0^\circ, 15MV$

Creëer in “Portal Dosimetry” een samengesteld beeld (= Create Composite Image) van deze twee beelden. Trek dosisprofielen door het samengestelde beeld en meet de dosisverschillen ter hoogte van aansluitingen.

 Vul in: dosisverschillen ($\Delta_{6MV}, \Delta_{15MV}$) bij over- of onderdosage ter hoogte van de aansluitingen

Mechanische Controle: 6-Wekelijks

Datum: dinsdag 6 juni 2017

Toestel: Clinac 1 Clinac 2 Truebeam 3
 Clinac 4 Clinac 5

Uitgevoerd door:

Kleiver aangebracht (enkel voor Hasselt):

Opmerkingen:

11.2 Appendix B: source code¹²

11.2.1 Main file: GUI

```
function varargout = Mech_Control_e_6W(varargin)
% MECH_CONTROLE_6W MATLAB code for Mech_Control_e_6W.fig
%   MECH_CONTROLE_6W, by itself, creates a new MECH_CONTROLE_6W or raises the existing
%   singleton*.
%
%   H = MECH_CONTROLE_6W returns the handle to a new MECH_CONTROLE_6W or the handle to
%   the existing singleton*.
%
%   MECH_CONTROLE_6W('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in MECH_CONTROLE_6W.M with the given input arguments.
%
%   MECH_CONTROLE_6W('Property','Value',...) creates a new MECH_CONTROLE_6W or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before Mech_Control_e_6W_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to Mech_Control_e_6W_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Mech_Control_e_6W

% Last Modified by GUIDE v2.5 01-May-2017 20:31:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @Mech_Control_e_6W_OpeningFcn, ...
                  'gui_OutputFcn',  @Mech_Control_e_6W_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Mech_Control_e_6W is made visible.
function Mech_Control_e_6W_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

¹² Published with MATLAB® R2016b

```

% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Mech_Control_e_6w (see VARARGIN)

% Choose default command line output for Mech_Control_e_6w
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
set(handles.analysis, 'Enable', 'off');
set(handles.openPDF, 'Enable', 'off');
global list;

% --- Outputs from this function are returned to the command line.
function varargout = Mech_Control_e_6w_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% -----
function openImages_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to openImages (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
openFiles;%get the images
set(hObject, 'Enable', 'off');
global list;
% questdialog with two options
    choice = questdlg(['Were there any mistakes made that could have mixed up
the order of the timestamps?'] , ...
        'Order of timestamps', ...
        'Yes','No','No');
% Handle response
switch choice
case 'Yes'
    list = sortImages; %sort the images
case 'No'
    list = sortImagesTime; %sort the images on timestamp
end

set(handles.analysis, 'Enable', 'on');

% --- Executes on button press in analysis.
function analysis_Callback(hObject, eventdata, handles)
% hObject    handle to analysis (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global totalImages;
if (totalImages > 0)
    set(hObject, 'Enable', 'off');
    analysis
    set(handles.openPDF, 'Enable', 'on');
else

```

```

msgbox('There are not any DICOM files loaded.');
```

```

end

% --- Executes on button press in openPDF.
function openPDF_Callback(hObject, eventdata, handles)
% hObject    handle to openPDF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global exportloc;
if exist(exportloc)
    open(exportloc);
else
    m = msgbox('There were no outputfiles found.');
```

```

end

% --- Executes on selection change in listbox.
function listbox_Callback(hObject, eventdata, handles)
% hObject    handle to listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

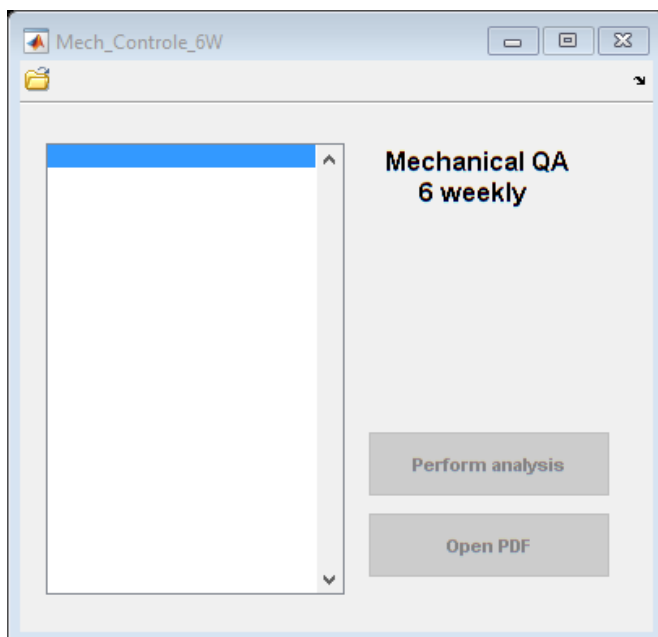
% --- Executes during object creation, after setting all properties.
function listbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```

end

```



11.2.2 OpenFiles.m

```
global path;
global imageFiles;
global imageIndex;
global totalImages;

path = uigetdir;
%If there is no directory selected, return messagebox
if(path == 0)
    msgbox('There were no images selected.');
```

end

```
imageFiles = dir([path '\*.dcm']);%The image files are dicomfiles

set(handles.figure1, 'pointer', 'watch');
imageIndex = find(~[imageFiles.isdir]); %get image indices
list = {imageFiles(~[imageFiles.isdir]).name}; %get list with image names
totalImages = length(imageIndex); %get total amount of images
set(handles.figure1, 'pointer', 'arrow');
```

%Display files in listbox

```
set(handles.listbox, 'String', list);
```

11.2.3 Sort images on timestamp

```
function [ sortedList ] = sortImagesTime()
%This function sorts the input image based on the timestamp
% The images were required in a certain predefined order.
% In order to know which image should undergo which operation,
% the images are sorted based on the time stamp in the dicom header.
% Output: List of images sorted on time stamp

global path;
global imageFiles;
global imageIndex;
global totalImages;

listPath = cell(totalImages,1);
listTime = [];
listSorted = cell(totalImages,1);

if totalImages ~= 19
    msgbox('There are not exactly 19 images in the directory');
    error('There are not exactly 19 images in the directory');
end

%Make a list of all the imagefile names and create a time list
for i = 1 : totalImages

    name = imageFiles(imageIndex(i)).name; %get name of image file
    imagePath = fullfile(path, name); %initialise the path of the image
    f = dicomread(imagePath); %Load the dicom file
    metadata = dicominfo(imagePath); %Load dicom header
    time = str2num(metadata.(dicomlookup('0008', '0033')));
```

```

listPath{i} = imagePath;
listTime = [listTime ; time];

end

j = 1;
while size(listTime, 1) >= 1
    [m, i] = min(listTime);%get value and index of lowest timestamp
    lowest = listPath{i};%get the name of that file
    listSorted{j} = lowest;%place in sorted list
    listTime(i,:) = [];%delete it from time and name list
    listPath(i) = [];
    j = j + 1;
end

sortedList = listSorted;

end

```

11.2.4 Sort images on properties

```

function [ sortedList ] = sortImages()
%This function sorts the input image based on the machine parameters
% The images were required in a certain predefined order.
% In order to know which image should undergo which operation,
% the images are sorted based on the collimator/gantry and table angles.
% output: sorted list of images

global path;
global imageFiles;
global imageIndex;
global totalImages;
g0c0t0 = 0;
g0c90t0 = 0;
g0c165t0 = 0;
g0c270t0 = 0;
g0c0t90 = 0;
g0c0t270 = 0;
g180c0t0 = 0;
g90c90t0 = 0;
g90c0t0 = 0;
g90c270t0 = 0;
g270c270t0 = 0;
g270c0t0 = 0;
g270c90t0 = 0;
vertversch = 0;
longversch = 0;
latversch = 0;
fs5x5 = 0;
fs10x10 = 0;
fs18x18 = 0;
gantry270 = cell(1,1);
gantry0 = cell(1,1);

%Sort images wheter on RTimage label (Clinac) or G/C/T angles (Truebeam)
for i = 1 : totalImages

    name = imageFiles(imageIndex(i)).name; %get name of image file

```



```

imagePath = fullfile(path, name); %initialise the path of the image
f = dicomread(imagePath); %Load the dicom file
metadata = dicominfo(imagePath); %Load dicom header
header = metadata.(dicomlookup('3002','0002')); %Get RTimageLabel
machine = metadata.(dicomlookup('0008', '1010')); %Get machine name
c = metadata.(dicomlookup('300a','0120')); %Collimator angle
g = metadata.(dicomlookup('300a','011e')); %Gantry angle
t = metadata.(dicomlookup('300a','0122')); %table angle

%Voor CLINAC
if contains(machine, 'CLINAC') == 1 || contains(machine, 'clinac') == 1

    %The three field sizes
    if contains(header, '5x5 VELD-') == 1
        fs5x5 = imagePath;
    elseif contains(header, '10x10 VELD-') == 1
        fs10x10 = imagePath;
    elseif contains(header, '18x18 VELD-') == 1
        fs18x18 = imagePath;

    %Difference mech/rad isocenter (13 images)
    elseif contains(header, 'G0 C0-') == 1
        g0c0t0 = imagePath;
    elseif contains(header, 'G0 C90-') == 1
        g0c90t0 = imagePath;
    elseif contains(header, 'G0 C165-') == 1
        g0c165t0 = imagePath;
    elseif contains(header, 'G0 C270-') == 1
        g0c270t0 = imagePath;
    elseif contains(header, 'G0 T90-') == 1
        g0c0t90 = imagePath;
    elseif contains(header, 'G0 T270-') == 1
        g0c0t270 = imagePath;
    elseif contains(header, 'G180-') == 1
        g180c0t0 = imagePath;
    elseif contains(header, 'G90 C90-') == 1
        g90c90t0 = imagePath;
    elseif contains(header, 'G90-') == 1
        g90c0t0 = imagePath;
    elseif contains(header, 'G90 C270-') == 1
        g90c270t0 = imagePath;
    elseif contains(header, 'G270 C270-') == 1
        g270c270t0 = imagePath;
    elseif contains(header, 'G270-') == 1
        g270c0t0 = imagePath;
    elseif contains(header, 'G270 C90-') == 1
        g270c90t0 = imagePath;

    %lateral/vertical/longitudinal table motion
    elseif contains(header, 'G0 LAT 15-') == 1
        latversch = imagePath;
    elseif contains(header, 'G0 LNG 15-') == 1
        longversch = imagePath;
    elseif contains(header, 'G0 VRT 15-') == 1 || contains(header, 'G270 VRT 15-') == 1
        vertversch = imagePath;
    end

elseif contains(machine, 'TRUEBEAM') == 1 || contains(machine, 'truebeam') == 1

```

```

%G=0; C=165; T=0
if c <= 166 && c >= 164
    if (g < 1 || g >= 359) && (t <= 1 || t >= 359)
        g0c165t0 = imagePath;
    elseif (g >= 1 || g <= 359) || (t >= 1 || t <= 359)
        % Construct a questdlg with two options
        choice = questdlg(['The gantry and/or table angles are not equal to 0.
Instead the gantry angle is ' num2str(g) ' & the table angle is ' num2str(t) '. Would you
like to go through with this values?'], ...
        'Image G0C165T0', ...
        'Yes','No','No');
        % Handle response
        switch choice
            case 'Yes'
                g0c165t0 = imagePath;
            case 'No'
                error('Image G0C165T0 is not correct');
        end
    end
end

%Table angles 90 and 270 (G and C are 0)
if t <= 91 && t >= 89
    if (g < 1 || g >= 359) && (c <= 1 || c >= 359)
        g0c0t90 = imagePath;

        elseif (g >= 1 || g <= 359) || (c >= 1 || c <= 359)
            % Construct a questdlg with two options
            choice = questdlg(['The gantry and collimator angles are not equal to 0.
Instead the gantry angle is ' num2str(g) ' & the collimator angle is ' num2str(c) '. Would
you like to go through with this value?'], ...
            'Image G0C0T90', ...
            'Yes','No','No');
            % Handle response
            switch choice
                case 'Yes'
                    g0c0t90 = imagePath;
                case 'No'
                    error('Image G0C0T90 is not correct');
            end
        end

    elseif t <= 271 && t >= 269
        if (g < 1 || g >= 359) && (c <= 1 || c >= 359)
            g0c0t270 = imagePath;
        elseif (g >= 1 || g <= 359) || (c >= 1 || c <= 359)
            % Construct a questdlg with two options
            choice = questdlg(['The gantry and collimator angles are not equal to 0.
Instead the gantry angle is ' num2str(g) ' & the collimator angle is ' num2str(c) '. Would
you like to go through with this value?'], ...
            'Image G0C0T270', ...
            'Yes','No','No');
            % Handle response
            switch choice
                case 'Yes'
                    g0c0t270 = imagePath;
                case 'No'
                    error('Image G0C0T270 is not correct');
            end
        end
    end
end

```

```

end

%Collimator 90 (G0C90T0; G90C90T0 and G270C90T0)
if c <= 91 && c >= 89
    %G0C90T0
    if (g < 1 || g >= 359) && (t <= 1 || t >= 359)
        g0c90t0 = imagePath;
    %G90C90T0
    elseif g <= 91 && g >= 89 && t <= 0.5 && t >= -0.5
        g90c90t0 = imagePath;
    %G270C90T0
    elseif g <= 271 && g >= 269 && t <= 0.5 && t >= -0.5
        g270c90t0 = imagePath;
    else
        % Construct a questdlg with two options
        choice = questdlg([ 'The gantry angle is ' num2str(g) ' The collimator angle
is' num2str(c) 'and the table angle is ' num2str(t) '. Please select the right image
assignment.' ] , ...
            'There went something wrong.', ...
            'G0C90T0','G90C90T0', 'G270C90T0' ,'G0C90T0');
        % Handle response
        switch choice
            case 'G0C90T0'
                g0c90t0 = imagePath;
            case 'G90C90T0'
                g90c90t0 = imagePath;
            case 'G270C90T0'
                g270c90t0 = imagePath;
        end
    end
end

%Collimator 270 (G0C270T0; G90C270T0 and G270C270T0)
if c <= 271 && c >= 269
    %G0C270T0
    if (g <= 1 || g >= 359) && (t <= 1 || t >= 359)
        g0c270t0 = imagePath;

    %G90C270T0
    elseif (g <= 91 && g >= 89) && (t <= 1 || t >= 359)
        g90c270t0 = imagePath;

    %G270C270T0
    elseif (g <= 271 && g >= 269) && (t <= 1 || t >= 359)
        g270c270t0 = imagePath;
    else
        % Construct a questdlg with two options
        choice = questdlg([ 'The gantry angle is ' num2str(g) ' The collimator angle
is ' num2str(c) 'and the table angle is ' num2str(t) '. Please select the right image
assignment.' ] , ...
            'There went something wrong.', ...
            'G0C270T0','G90C270T0', 'G270C270T0' ,'G0C270T0');
        % Handle response
        switch choice
            case 'G0C270T0'
                g0c270t0 = imagePath;
            case 'G90C270T0'
                g90c270t0 = imagePath;

```

```

        case 'G270C270T0'
            g270c270t0 = imagePath;
        end
    end
end

%Gantry 90 and 180 (both with C and T zero)
if (c <= 1 || c >= 359) && not(g <= 1 || g >= 359)
    %G90C0T0
    if (g <= 91 && g >= 89) && (t <= 1 || t >= 359)
        g90c0t0 = imagePath;
    %G180C0T0
    elseif (g <= 181 && g >= 179) && (t <= 1 || t >= 359)
        g180c0t0 = imagePath;

    %G270C0T0 and G270 VRT 15
    elseif (g <= 271 && g >= 269) && (t <= 1 || t >= 359)
        gantry270 = [gantry270 ; imagePath];

    else
        % Construct a questdlg with two options
        choice = questdlg([ 'The gantry angle is ' num2str(g) ' The collimator
angle is ' num2str(c) 'and the table angle is ' num2str(t) '. Please select the right image
assignment.' ] , ...
            'There went something wrong.', ...
            'G90C0T0', 'G180C0T0', 'G270C0T0' , 'G90C0T0');
        % Handle response
        switch choice
            case 'G90C0T0'
                g90c0t0 = imagePath;
            case 'G180C0T0'
                g180c0t0 = imagePath;
            case 'G270C0T0'
                gantry270 = [gantry270 ; imagePath];
        end
    end
end

%Gantry angle 0 and lateral/longitudinal motion and Fieldsizes
%(all on G0C0T0)

if (g <= 1 || g >= 359) && (c <= 1 || c >= 359) && (t <= 1 || t >= 359)
    gantry0 = [gantry0 ; imagePath];
end

end
end

if contains(machine, 'TRUEBEAM') == 1 || contains(machine, 'truebeam') == 1
    gantry0(1) = [];
    gantry270(1) = [];

%Sort the six images with G = 0 on timestamp
[rows, cols] = size(gantry0);
%Check if there are exactly six images in array
if rows ~= 6
    error('There are not exactly six images with G = 0°; C = 0°; T = 0°');
end

gantry0_time = [];

```

```

gantry0_sort = cell(1,1);

for i = 1 : rows
    metadata_g0 = dicominfo(gantry0{i});
    time = str2num(metadata_g0.(dicomlookup('0008','0033')));
    gantry0_time = [gantry0_time ; time];
end

while size(gantry0_time,1) >= 1
    [m, i] = min(gantry0_time);
    mini = gantry0{i};
    gantry0_sort = [gantry0_sort ; mini];
    gantry0_time(i,:) = [];
    gantry0(i) = [];
end
gantry0_sort(1) = [];
j = 1;
while j <= rows
    switch j
        case 1
            g0c0t0 = gantry0_sort{j};
        case 2
            longversch = gantry0_sort{j};
        case 3
            latversch = gantry0_sort{j};
        case 4
            fs5x5 = gantry0_sort{j};
        case 5
            fs10x10 = gantry0_sort{j};
        case 6
            fs18x18 = gantry0_sort{j};
        otherwise
            %do nothing
    end
    j = j + 1;
end

%Sort the two images with G = 270 on timestamp
[row, col] = size(gantry270);
%Check if there are exactly two images in array
if row ~= 2
    error('There are not exactly two images with G = 270°; C = 0° and T = 0°.');
end
%Sort them on timestamp
metadata_t1 = dicominfo(gantry270{1});
g270_t1 = metadata_t1.(dicomlookup('0008','0033'));
g270_t1 = str2num(g270_t1);
metadata_t2 = dicominfo(gantry270{2});
g270_t2 = metadata_t2.(dicomlookup('0008','0033'));
g270_t2 = str2num(g270_t2);

if g270_t1 < g270_t2
    g270c0t0 = gantry270{1};
    vertversch = gantry270{2};
else
    g270c0t0 = gantry270{2};
    vertversch = gantry270{1};
end
end

```

```

listSorted = cell(19,1);
listSorted{1} = g0c0t0;
listSorted{2} = g0c90t0;
listSorted{3} = g0c165t0;
listSorted{4} = g0c270t0;
listSorted{5} = g0c0t90;
listSorted{6} = g0c0t270;
listSorted{7} = g180c0t0;
listSorted{8} = g90c90t0;
listSorted{9} = g90c0t0;
listSorted{10} = g90c270t0;
listSorted{11} = g270c270t0;
listSorted{12} = g270c0t0;
listSorted{13} = g270c90t0;
listSorted{14} = vertversch;
listSorted{15} = longversch;
listSorted{16} = latversch;
listSorted{17} = fs5x5;
listSorted{18} = fs10x10;
listSorted{19} = fs18x18;

sortedList = listSorted;

end

```

11.2.5 Perform analysis

```

global path;
global imageFiles;
global imageIndex;
global totalImages;
global location;
global list;
global exportLoc;

files = cell(1,totalImages);
%First divide the list of images into 5 groups
%1: distance between mech/rad isocenter
dist = list(1:13);
%2: vertical table motion
vert = list{14};
%3: Longitudinal table motion
long = list{15};
%4: Lateral table motion
lat = list{16};
%5 field sizes;
fs = list(17:19);

%% Distance between mech/rad isocenter

w = waitbar(0, 'Calculating distances between mech/rad isocenter');
dist_results = struct('i',0,'Gantry',0, 'Collimator', 0, 'Table', 0, 'deltaIso', 0);
x_iso = [];
y_iso = [];
for i = 1 : 13
    metadata = dicominfo(dist{i}); %get dicom header

```

```

c = metadata.(dicomlookup('300a','0120'));%Collimator angle
g = metadata.(dicomlookup('300a','011e'));%Gantry angle
t = metadata.(dicomlookup('300a','0122'));%table angle
try
if i == 3

    [deltaIso, xRad, yRad, xMech, yMech, rad, veld] = deltaxiso165(dist{i}); %computations

else

    [deltaIso, xRad, yRad, xMech, yMech, rad, veld] = deltaxiso(dist{i}); %computations

end

%create figure without displaying and save a handle
fhandle = figure('visible','off');
imshow(dicomread(dist{i}), 'DisplayRange', []);

%plotting
set(0,'CurrentFigure',fhandle), hold on, plot(xRad, yRad, '*'); %radiation iso
%plot field edges
set(0,'CurrentFigure',fhandle), hold on, line([veld(1) veld(3)], [veld(2) veld(4)],
'Color', 'y');
set(0,'CurrentFigure',fhandle), hold on, line([veld(5) veld(7)], [veld(6) veld(8)],
'Color', 'y');
set(0,'CurrentFigure',fhandle), hold on, line([veld(3) veld(7)], [veld(4) veld(8)],
'Color', 'y');
set(0,'CurrentFigure',fhandle), hold on, line([veld(5) veld(1)], [veld(6) veld(2)],
'Color', 'y');
set(0,'CurrentFigure',fhandle), hold on, viscircles([xMech yMech], rad); %mechanical iso
set(0,'CurrentFigure',fhandle), hold on, plot(xMech, yMech, '+');
%diagonals
set(0,'CurrentFigure',fhandle), hold on, line([veld(1) veld(7)], [veld(2) veld(8)],
'Color', 'y');
set(0,'CurrentFigure',fhandle), hold on, line([veld(5) veld(3)], [veld(6) veld(4)],
'Color', 'y');

%Setting up results
in1 = sprintf('%d: G: %.2f°, C: %.2f° en T: %.2f°',i,g,c,t);
res = sprintf('; distance between mechanical and radiation isocenter = %.3fmm.',deltaIso);
result = title({in1;res}, 'FontSize', 8);

%Save results
dist_results(i).i = i;
dist_results(i).Gantry = round(g,0);
dist_results(i).Collimator = round(c,0);
dist_results(i).Table = round(t,0);
dist_results(i).deltaIso = round(deltaIso, 3);

catch
disp('Error occurred during the analysis of one of the pictures. Code will continue
running. ');
end
%Save radiation iso's for Colli = 0,165,90,270 for averaging
switch i
case 1
x_iso = [x_iso xRad];
y_iso = [y_iso yRad];

```

```

        k = dicomread(dist{i})
        [Rr, Cc] = size(k);
    case 2
        x_iso = [x_iso xRad];
        y_iso = [y_iso yRad];
    case 3
        x_iso = [x_iso xRad];
        y_iso = [y_iso yRad];
    case 4
        x_iso = [x_iso xRad];
        y_iso = [y_iso yRad];
    otherwise
        %do nothing
end

%write to pdf
try
    %Zoom to dot
    pos = get(fhandle, 'CurrentPoint');
    zoom_pos = [xMech yMech];
    set(fhandle, 'CurrentPoint', zoom_pos);
    field = zoom(fhandle);
    field.Enable = 'on';

    %Make sure one page in pdf is one image
    fhandle.PaperPositionMode = 'auto';
    fig_pos = fhandle.PaperPosition;
    fhandle.PaperSize = [fig_pos(3) fig_pos(4)];

    %write to pdf
    pathOut = path;
    fileOut = sprintf('out%d_v1.pdf', i);
    loc = fullfile(pathOut, fileOut);

    %Check if file already exists
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf'); %save as pdf
    fileattrib(loc, '+h -w', '', 's'); %hidden&read-only
    files{end+1} = loc;

    %Zoomed in image to pdf
    set(0, 'CurrentFigure', fhandle), hold on, zoom(20);
    fileOut = sprintf('out%d_v2.pdf', i);
    loc = fullfile(pathOut, fileOut);
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf');
    fileattrib(loc, '+h -w', '', 's');
    files{end+1} = loc;

catch ME
    msgbox(ME.message);
    msgbox('Error during writing to pdf');
    %delete files????
    close(w);
    close(fhandle);
end
waitbar(i/13);
close(fhandle);

```



```

end

table_dist = struct2table(dist_results); %results to table

%write table to excel
waitbar(1, w, 'Writing results to excel');
fileOut = 'table.xls';
loc = fullfile(pathOut,fileOut);
deleteifexist(loc);
writetable(table_dist, loc);
fileattrib(loc,'+h ',' ','s'); %hidden maken

%% Vertical table motion

waitbar(1, w, 'Calculating lateral table motion. ');
vert_results = struct('VRT15', 0, 'VRT_dev', 0, 'Long_dev', 0);

try
    dcm1 = dist{12};
    dcm2 = vert;
    [vertAfw, longver2, vert_x1, vert_y1, vert_x2, vert_y2, vert_r1, vert_r2] =
    Verticaal(dcm1, dcm2);
    vertImage = imfuse(dicomread(dcm1), dicomread(dcm2), 'blend', 'Scaling', 'joint');
    %Create figure without displaying and save its handle
    fhandle = figure('visible','off');
    imshow(vertImage,'DisplayRange',[]);
    [R, C] = size(vertImage);

    %plotting
    set(0,'CurrentFigure',fhandle), hold on,  viscircles([vert_x1 vert_y1], vert_r1); %first
circle
    set(0,'CurrentFigure',fhandle), hold on,  viscircles([vert_x2 vert_y2], vert_r2); %second
circle
    set(0,'CurrentFigure',fhandle), hold on,  line([vert_x1 vert_x2], [vert_y1 vert_y2]);%join
with line

    %Setting up results
    in1 = sprintf('The vertical deviation is %.2fmm', vertAfw);
    res = sprintf('The longitudinal displacement is %.2fmm.',longver2);
    result = title({in1;res}, 'FontSize', 8);

    %Save results
    lat_results(1).VRT15 = 'done';
    lat_results(1).VRT_dev = round(vertAfw,3);
    lat_results(1).Long_dev = round(longVer2,3);

catch
    disp('Error occured during the analysis of one of the pictures. Code will continue
running. ');
end

%write to pdf
try
    %Full picture
    pos = get(fhandle, 'CurrentPoint');
    zoom_pos = [vert_x1 vert_y1];
    set(fhandle, 'CurrentPoint', zoom_pos);
    field = zoom(fhandle);
    field.Enable = 'on';

```

```

%Make sure one page in pdf is one image
fhandle.PaperPositionMode = 'auto';
fig_pos = fhandle.PaperPosition;
fhandle.PaperSize = [fig_pos(3) fig_pos(4)];

%write to pdf
pathOut = path;
fileOut = sprintf('out%d_v1.pdf', 14);
loc = fullfile(pathOut, fileOut);

%Check if file already exists
deleteifexist(loc);
saveas(fhandle, loc, 'pdf'); %save as pdf
fileattrib(loc, '+h -w', '', 's'); %hidden&read-only
files{end+1} = loc;

%First zoomed in image to pdf
set(fhandle, 'CurrentPoint', zoom_pos);
set(0, 'CurrentFigure', fhandle), hold on,
axis([vert_x1-20 vert_x1+20 vert_y1-20 vert_y2+20]), zoom(4);
fileOut = sprintf('out%d_v2.pdf', 14);
loc = fullfile(pathOut, fileOut);
deleteifexist(loc);
saveas(fhandle, loc, 'pdf');
fileattrib(loc, '+h -w', '', 's');
files{end+1} = loc;

%Second zoomed in image to pdf
set(fhandle, 'CurrentPoint', zoom_pos);
set(0, 'CurrentFigure', fhandle), hold on,
axis([vert_x2-20 vert_x2+20 vert_y1-20 vert_y2+20]), zoom(4);
fileOut = sprintf('out%d_v3.pdf', 14);
loc = fullfile(pathOut, fileOut);
deleteifexist(loc);
saveas(fhandle, loc, 'pdf');
fileattrib(loc, '+h -w', '', 's');
files{end+1} = loc;
close(fhandle);
catch ME
    msgbox(ME.message);
    msgbox('Error during writing to pdf');
    close(fhandle);
end

table_vert = struct2table(lat_results); %results to table

%Table to excel
waitbar(1, w, 'Writing results to excel');
fileOut = 'table.xls';
loc = fullfile(pathOut, fileOut);
writetable(table_vert, loc, 'Sheet', 1, 'Range', 'A16')
fileattrib(loc, '+h ', '', 's'); %hidden maken

%% Longitudinal table motion

waitbar(1, w, 'Calculating longitudinal table motion. ');
long_results = struct('LNG15', 0, 'LNG_dev', 0, 'Lat_dev', 0);

try

```

```

dcm1 = dist{1};
dcm2 = long;
[longAfw, latVer, long_x1, long_y1, long_x2, long_y2, long_r1, long_r2] =
Longitudinal(dcm1, dcm2); %calculate
longImage = imfuse(dicomread(dcm1),dicomread(dcm2),'blend','scaling','joint');
%create figure without displaying and save a handle
fhandle = figure('visible','off');
imshow(longImage,'DisplayRange',[]);
[R, C] = size(longImage);

%plotting
set(0,'CurrentFigure',fhandle), hold on, viscircles([long_x1 long_y1], long_r1); %first
circle
set(0,'CurrentFigure',fhandle), hold on, viscircles([long_x2 long_y2], long_r2); %second
circle
set(0,'CurrentFigure',fhandle), hold on, line([long_x1 long_x2], [long_y1 long_y2]);%join
with line

%Setting up results
in1 = sprintf('The longitudinal deviation is %.2fmm', longAfw);
res = sprintf('The lateral displacement is %.2fmm.',latVer);
result = title({in1;res}, 'FontSize', 8);
%Save results
long_results(1).LNG15 = 'done';
long_results(1).LNG_dev = round(longAfw,3);
long_results(1).Lat_dev = round(latVer,3);

catch
disp('Error occurred during the analysis of one of the pictures. Code will continue
running. ');
end

%write to pdf
try
%Full picture
pos = get(fhandle, 'CurrentPoint');
zoom_pos = [C/2 R/2];
set(fhandle, 'CurrentPoint', zoom_pos);
field = zoom(fhandle);
field.Enable = 'on';

%Make sure one page in pdf is one image
fhandle.PaperPositionMode = 'auto';
fig_pos = fhandle.PaperPosition;
fhandle.PaperSize = [fig_pos(3) fig_pos(4)];

%write to pdf
pathOut = path;
fileOut = sprintf('out%d_v1.pdf', 15);
loc = fullfile(pathOut, fileOut);

%Check if file already exists
deleteifexist(loc);
saveas(fhandle, loc, 'pdf'); %save as pdf
fileattrib(loc, '+h -w', '', 's'); %hidden&read-only
files{end+1} = loc;

%First zoomed in image to pdf

```

```

set(fhandle, 'CurrentPoint', zoom_pos);
set(0, 'CurrentFigure', fhandle), hold on,
axis([long_x1-20 long_x1+20 long_y1-20 long_y1+20]), zoom(4);
fileOut = sprintf('out%d_v2.pdf', 15);
loc = fullfile(pathOut, fileOut);
deleteifexist(loc);
saveas(fhandle, loc, 'pdf');
fileattrib(loc, '+h -w', '', 's');
files{end+1} = loc;

%Second zoomed in image to pdf
set(fhandle, 'CurrentPoint', zoom_pos);
set(0, 'CurrentFigure', fhandle), hold on,
axis([long_x2-20 long_x2+20 long_y2-20 long_y2+20]), zoom(4);
fileOut = sprintf('out%d_v3.pdf', 15);
loc = fullfile(pathOut, fileOut);
deleteifexist(loc);
saveas(fhandle, loc, 'pdf');
fileattrib(loc, '+h -w', '', 's');
files{end+1} = loc;
close(fhandle);
catch ME
    msgbox(ME.message);
    msgbox('Error during writing to pdf');
    close(fhandle);
end

```

```
table_long = struct2table(long_results); %results to table
```

```

%Table to excel
waitbar(1, w, 'Writing results to excel');
fileOut = 'table.xls';
loc = fullfile(pathOut, fileOut);
writetable(table_long, loc, 'Sheet', 1, 'Range', 'A19')
fileattrib(loc, '+h ', '', 's'); %hidden maken

```

```
%% Lateral table motion
```

```

waitbar(1, w, 'Calculating lateral table motion. ');
lat_results = struct('LAT15', 0, 'LAT_dev', 0, 'Long_dev', 0);

try
    dcm1 = dist{1};
    dcm2 = lat;
    [latAfw, longVer, lat_x1, lat_y1, lat_x2, lat_y2, lat_r1, lat_r2] = Lateral(dcm1, dcm2);
    latImage = imfuse(dicomread(dcm1), dicomread(dcm2), 'blend', 'scaling', 'joint');
    %Create figure without displaying and save its handle
    fhandle = figure('visible','off');
    imshow(latImage, 'DisplayRange', []);
    [R, C] = size(latImage);

    %plotting
    set(0, 'CurrentFigure', fhandle), hold on, viscircles([lat_x1 lat_y1], lat_r1); %first
circle
    set(0, 'CurrentFigure', fhandle), hold on, viscircles([lat_x2 lat_y2], lat_r2); %second
circle
    set(0, 'CurrentFigure', fhandle), hold on, line([lat_x1 lat_x2], [lat_y1 lat_y2]); %join
with line

    %Setting up results

```

```

in1 = sprintf('The lateral deviation is %.2fmm', latAfw);
res = sprintf('The longitudinal displacement is %.2fmm.', longVer);
result = title({in1;res}, 'FontSize', 8);

%Save results
lat_results(1).LAT15 = 'done';
lat_results(1).LAT_dev = round(latAfw,3);
lat_results(1).Long_dev = round(longVer,3);

catch
    disp('Error ocured during the analysis of one of the pictures. Code will continue
running.');
```

```

end

%write to pdf
try
    %Full picture
    pos = get(fhandle, 'CurrentPoint');
    zoom_pos = [C/2 R/2];
    set(fhandle, 'CurrentPoint', zoom_pos);
    field = zoom(fhandle);
    field.Enable = 'on';

    %Make sure one page in pdf is one image
    fhandle.PaperPositionMode = 'auto';
    fig_pos = fhandle.PaperPosition;
    fhandle.PaperSize = [fig_pos(3) fig_pos(4)];

    %write to pdf
    pathOut = path;
    fileOut = sprintf('out%d_v1.pdf', 16);
    loc = fullfile(pathOut, fileOut);

    %Check if file already exists
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf'); %save as pdf
    fileattrib(loc, '+h -w', '', 's'); %hidden&read-only
    files{end+1} = loc;

    %First zoomed in image to pdf
    set(fhandle, 'CurrentPoint', zoom_pos);
    set(0, 'CurrentFigure', fhandle), hold on,
    axis([lat_x1-20 lat_x1+20 lat_y1-20 lat_y1+20]), zoom(4);
    fileOut = sprintf('out%d_v2.pdf', 16);
    loc = fullfile(pathOut, fileOut);
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf');
    fileattrib(loc, '+h -w', '', 's');
    files{end+1} = loc;

    %Second zoomed in image to pdf
    set(fhandle, 'CurrentPoint', zoom_pos);
    set(0, 'CurrentFigure', fhandle), hold on,
    axis([lat_x2-20 lat_x2+20 lat_y2-20 lat_y2+20]), zoom(4);
    fileOut = sprintf('out%d_v3.pdf', 16);
    loc = fullfile(pathOut, fileOut);
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf');
```

```

        fileattrib(loc, '+h -w', '', 's');
        files{end+1} = loc;
        close(fhandle);
    catch ME
        msgbox(ME.message);
        msgbox('Error during writing to pdf');
        close(fhandle);
    end

    table_lat = struct2table(lat_results); %results to table

    %Table to excel
    waitbar(1, w, 'Writing results to excel');
    fileOut = 'table.xls';
    loc = fullfile(pathOut, fileOut);
    writetable(table_lat, loc, 'Sheet', 1, 'Range', 'A22');
    fileattrib(loc, '+h ', '', 's'); %hidden maken

%% Assymetric fieldsizes

if Rr ~= 384
    x_iso = double(mean(x_iso));
    y_iso = double(mean(y_iso));
else
    x_iso = double(mean(x_iso))*2;
    y_iso = double(mean(y_iso))*2;
end
w = waitbar(0, 'Calculating asymmetrical fieldsizes');
field_results = struct('field', 0, 'x1_dev', 0, 'x2_dev', 0, 'y1_dev', 0, 'y2_dev', 0,
    'x_total_dev', 0, 'y_total_dev', 0);
for i = 1 : 3
    try
        dcm = fs{i};
        [fs_x1, fs_x2, fs_xtot, fs_y1, fs_y2, fs_ytot, fs_veld] = veldgrootte(dcm, x_iso, y_iso);
        image = dicomread(dcm);
        [R, C] = size(image);
        %Create figure without displaying and save its handle
        fhandle = figure('visible','off');
        imshow(image, 'DisplayRange', []);

        %plotting
        set(0, 'CurrentFigure', fhandle), hold on, plot(x_iso, y_iso, '*');
        set(0, 'CurrentFigure', fhandle), hold on, line([fs_veld(1) fs_veld(3)], [fs_veld(2)
fs_veld(4)], 'Color', 'y');
        set(0, 'CurrentFigure', fhandle), hold on, line([fs_veld(5) fs_veld(7)], [fs_veld(6)
fs_veld(8)], 'Color', 'y');
        set(0, 'CurrentFigure', fhandle), hold on, line([fs_veld(3) fs_veld(7)], [fs_veld(4)
fs_veld(8)], 'Color', 'y');
        set(0, 'CurrentFigure', fhandle), hold on, line([fs_veld(5) fs_veld(1)], [fs_veld(6)
fs_veld(2)], 'Color', 'y');

        switch i
            case 1
                name = '5x5: ';
                fsize_a = 25;
                fsize = 50;
            case 2
                name = '10x10: ';
                fsize_a = 50;
                fsize = 100;
        end
    catch
    end
end

```

```

case 3
    name = '18x18: ';
    fsize_a = 90;
    fsize = 180;
end

%Setting up results
in1 = sprintf('%s x1 is %.2fmm and x2 is %.2fmm', name, fs_x1, fs_x2);
res = sprintf('y1 is %.2fmm and y2 is %.2fmm', fs_y1, fs_y2);
res2 = sprintf('Total field size x: %.2fmm and total field size y: %.2fmm', fs_xtot,
fs_ytot);
result = title({in1;res;res2}, 'FontSize', 5);

%Compute deviations
fs_x1 = fsize_a - fs_x1;
fs_x2 = fsize_a - fs_x2;
fs_y1 = fsize_a - fs_y1;
fs_y2 = fsize_a - fs_y2;
fs_xtot = fsize - fs_xtot;
fs_ytot = fsize - fs_ytot;

%Save results
field_results(i).field = name;
field_results(i).x1_dev = round(fs_x1,3);
field_results(i).x2_dev = round(fs_x2,3);
field_results(i).y1_dev = round(fs_y1,3);
field_results(i).y2_dev = round(fs_y2,3);
field_results(i).x_total_dev = round(fs_xtot,3);
field_results(i).y_total_dev = round(fs_ytot,3);

catch
    disp('Error occurred during the analysis of one of the pictures. Code will continue
running. ');
end

%write to pdf
try
    %Zoom to dot
    pos = get(fhandle, 'CurrentPoint');
    zoom_pos = [x_iso y_iso];
    set(fhandle, 'CurrentPoint', zoom_pos);
    field = zoom(fhandle);
    field.Enable = 'on';

    %Make sure one page in pdf is one image
    fhandle.PaperPositionMode = 'auto';
    fig_pos = fhandle.PaperPosition;
    fhandle.PaperSize = [fig_pos(3) fig_pos(4)];

    %write to pdf
    pathOut = path;
    fileOut = sprintf('out%d_v1.pdf', i+16);
    loc = fullfile(pathOut, fileOut);

    %Check if file already exists
    deleteifexist(loc);
    saveas(fhandle, loc, 'pdf'); %save as pdf

```

```

        fileattrib(loc, '+h -w', '', 's'); %hidden&read-only
        files{end+1} = loc;

    catch ME
        msgbox(ME.message);
        msgbox('Error during writing to pdf');
        close(w);
        close(fhandle);
    end
    waitbar(i/3);
    close(fhandle);
end

table_field = struct2table(field_results); %results to table

%Table to excel
waitbar(1, w, 'Writing results to excel');
fileOut = 'table.xls';
loc = fullfile(pathOut, fileOut);
writetable(table_field, loc, 'Sheet', 1, 'Range', 'A25')
fileattrib(loc, '+h ', '', 's'); %hidden maken

%% Convert individual PDFs to one PDF (gohstscript needed)

Outputfile = 'Mechanische_QA_6w.pdf'; %Create output file
exportloc = fullfile(pathOut, Outputfile);
try
    hExcel = actxserver('Excel.Application'); %Start excel
    hworkbook = hExcel.Workbooks.Open(loc); %Open excel table
    hworksheet = hworkbook.Sheets.Item(1);
    deleteifexist(exportloc);
    hworksheet.ExportAsFixedFormat('xlTypePDF', exportloc);

    append_pdfs(exportloc, files{:}); %Fuse all the PDFs in one file

    %Delete all other pdf's & excel files
    for k = 1 : length(files)
        delete(files{k});
    end
    %Close excel
    hworkbook.Save;
    Quit(hExcel);
    delete(hExcel);
    delete(fullfile(pathOut, 'table.xls')); %Excel tabel verwijderen
catch ME
    msgbox(ME.message);
    msgbox('Error: writing to PDF');
    %Close excel
    hworkbook.Save;
    Quit(hExcel);
    delete(hExcel);
    %Delete Files
    delete(fullfile(pathOut, 'table.xls'));
    for k=1:length(files)
        delete(files{k});
    end
    close(w);
    error('Error: writing to PDF');
end

```



```

%Close waitbar
close(w);

%Messagebox when analysis is done
bericht = sprintf('Analysis done');

msgbox(bericht);

```

11.2.6 Distance between mechanical- and radiation isocentre

```

function [ deltaiso, x_straliso, y_straliso, x_mechiso, y_mechiso, radius, veld ] = deltaiso(
dcm )

%This function detects the radiation and mechanical isocenter
%Afterwards it computes the distance between both
%input: the dicom image
%output: distance between both iso's, the radiation and mechanical iso,
%      radius of detected circle and field parameters for plotting
dicom = dcm;
g = dicomread(dicom);
f = medfilt2(g, 'symmetric');%noise removal
g_med = imadjust(f,stretchlim(f),[]);%contrast stretching

%Size of the image (resolution)
[R, C] = size(f);

%Convert image to class uint16
o_u16 = uint16(g_med);

%Conversionfactor for distance computation (mm per pixel)
metadata = dicominfo(dicom);
conversionfactor = metadata.ImagePlanePixelSpacing(1) * (metadata.RadiationMachineSAD /
metadata.RTImageSID);

%Find left field edge (splitted up in two to avoid crossing one of the
%dots in the image, which could disturb the computation)
[ x_links1, y_links1 ] = veldrand_recht(o_u16, [C/3.5 C/6], R/3.5, R/2.1, 1);

[ x_links2, y_links2 ] = veldrand_recht(o_u16, [C/3.5 C/6], R/1.9, 2.5*R/3.5, 1);
%Fuse datasets and remove outliers
x_links3 = [x_links1 x_links2];
    x_mu1 = mean(x_links3);
    stdev1 = std(x_links3);
    ind1 = find(abs(x_links3)<(x_mu1 - 3*stdev1));
    x_links3(ind1) = [];
    x_links4 = x_links3;
    ind21 = find(abs(x_links4)>(x_mu1 + 3*stdev1));
    x_links4(ind21) = [];
    x_links = x_links4;
y_links3 = [y_links1 y_links2];
    y_links3(ind1) = [];
    y_links3(ind21) = [];
    y_links = y_links3;
p_links = polyfit(y_links,x_links,1);%Fit function through points
syms X ;
yfit_links(X) = p_links(1)*X + p_links(2);
yfit_linksinv(X) = finverse(yfit_links(X));

```

```

%Find right field edge (splitted up in two to avoid crossing one of the
%dots in the image, which could disturb the computation)
[ x_rechts1, y_rechts1 ] = velrand_recht(o_u16, [2.5*C/3.5 5*C/6], R/3.5, R/2.1, 0);

[ x_rechts2, y_rechts2 ] = velrand_recht(o_u16, [2.5*C/3.5 5*C/6], R/1.9, 2.5*R/3.5, 0);
%Fuse datasets and remove outliers
x_rechts3 = [x_rechts1 x_rechts2];
    x_mur = mean(x_rechts3);
    stdevr = std(x_rechts3);
    indr = find(abs(x_rechts3)<(x_mur - 3*stdevr));
    x_rechts3(indr) = [];
    x_rechts4 = x_rechts3;
    ind2r = find(abs(x_rechts4)>(x_mur + 3*stdevr));
    x_rechts4(ind2r) = [];
    x_rechts = x_rechts4;
y_rechts3 = [y_rechts1 y_rechts2];
    y_rechts3(indr) = [];
    y_rechts3(ind2r) = [];
    y_rechts = y_rechts3;
p_rechts = polyfit(y_rechts,x_rechts,1); %fit function through points
yfit_rechts(X) = p_rechts(1)*X + p_rechts(2);
yfit_rechtsinv(X) = finverse(yfit_rechts(X));

%Find top field edge (splitted up in two to avoid crossing one of the
%dots in the image, which could disturb the computation)
[x_boven1, y_boven1] = velrand_liggend(o_u16, [R/10 R/6], C/3, C/2.1, 1);

[x_boven2, y_boven2] = velrand_liggend(o_u16, [R/10 R/6], C/1.9, 2*C/3, 1);

y_boven3 = [y_boven1 y_boven2];
    y_mub = mean(y_boven3);
    stdevb = std(y_boven3);
    indb = find(abs(y_boven3)<(y_mub - 3*stdevb));
    y_boven3(indb) = [];
    y_boven4 = y_boven3;
    ind2b = find(abs(y_boven4)>(y_mub + 3*stdevb));
    y_boven4(ind2b) = [];
    y_boven = y_boven4;
x_boven3 = [x_boven1 x_boven2];
    x_boven3(indb) = [];
    x_boven3(ind2b) = [];
    x_boven = x_boven3;
p_boven = polyfit(x_boven,y_boven,1);
yfit_boven(X) = p_boven(1)*X + p_boven(2);

%Find bottom field edge (splitted up in two to avoid crossing one of the
%dots in the image, which could disturb the computation)
[x_onder1, y_onder1] = velrand_liggend(o_u16, [9*R/10 5*R/6], C/3, C/2.1, 0);

[x_onder2, y_onder2] = velrand_liggend(o_u16, [9*R/10 5*R/6], C/1.9, 2*C/3, 0);

y_onder3 = [y_onder1 y_onder2];
    y_mu0 = mean(y_onder3);
    stdevo = std(y_onder3);
    indo = find(abs(y_onder3)<(y_mu0 - 3*stdevo));
    y_onder3(indo) = [];
    y_onder4 = y_onder3;
    ind2o = find(abs(y_onder4)>(y_mu0 + 3*stdevo));
    y_onder4(ind2o) = [];

```

```

y_onder = y_onder4;
x_onder3 = [x_onder1 x_onder2];
x_onder3(indo) = [];
x_onder3(ind2o) = [];
x_onder = x_onder3;
p_onder = polyfit(x_onder,y_onder,1);
yfit_onder(x) = p_onder(1)*x + p_onder(2);

%Find intersection of field edges
[x1,y1] = vindSnijpunt(yfit_linksinv, yfit_boven);
[x2,y2] = vindSnijpunt(yfit_rechtsinv, yfit_boven);
[x3,y3] = vindSnijpunt(yfit_linksinv, yfit_onder);
[x4,y4] = vindSnijpunt(yfit_rechtsinv, yfit_onder);

veld = [x1 y1 x2 y2 x3 y3 x4 y4];%save field parameters for plotting

%Compute radiation iso (intersection of diagonals)
x_straliso = vindSnijpiso_x(x1, y1, x2, y2, x3, y3, x4, y4);
y_straliso = vindSnijpiso_y(x1, y1, x2, y2, x3, y3, x4, y4)

%edge detection
[~, threshold] = edge(o_u16, 'sobel');
fudgeFactor = .5;
Bws = edge(o_u16,'sobel', threshold * fudgeFactor);

%morphological closing
Bwdfill = imclose(Bws, strel('disk', 1));

se = strel('disk', 1);%set up structure element
erode = imerode(Bwdfill, se);%erosion
dilate = imdilate(erode, se);%dilation

%Detect all circles and their centroids and radii
stats = regionprops('table',dilate,'Centroid',...
'MajorAxisLength','MinorAxisLength');
centers = stats.Centroid;%get centroids
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);%diameters
radii = diameters/2;%radii
[maxRad, ind] = max(radii);%Since there is always one circle detected which
%enveloppes the whole image -> discard this one
radii(ind) = [];
centers(ind,:) = [];

mechiso = [];
zoekcrit = 0.5;
done = 0;
it = 0;
%Iterative process which expands the ROI untill one circle is detected
while done == 0
    %Set up ROI
    r_roi = double(y_straliso);
    c_roi = double(x_straliso);
    a = zoekcrit/conversiefactor;

    r_roi_array = [(r_roi)-a ; (r_roi)-a; (r_roi)+a; (r_roi)+a];
    c_roi_array = [(c_roi)-a ; (c_roi)+a; (c_roi)+a; (c_roi)-a];
    xq = centers(:,1);
    yq = centers(:,2);
    in = inpolygon(xq, yq, c_roi_array, r_roi_array);
    %Find centers in ROI
    x_mechiso = xq(in);

```

```

y_mechiso = yq(in);
mechiso = [x_mechiso y_mechiso];
[ro, co] = size(mechiso);
if ro == 1
    done = 1;
end
if it > 10
    done = 1;
end
zoekcrit = zoekcrit*2;
it = it + 1;
end

%Find radius of detected circle
ind1 = find(xq == x_mechiso);
ind2 = find(yq == y_mechiso);
ind = intersect(ind1,ind2);
radius = radii(ind);

%Compute distance between rad and mech iso (in mm)
delta_iso = afstand2p(x_mechiso, x_straliso, y_mechiso, y_straliso);
deltaiso = delta_iso*conversiefactor;
end

```

```

function [ deltaiso, x_straliso, y_straliso, x_mechiso, y_mechiso, radius, veld ] =
deltaiso165( dcm )

%This function detects the radiation and mechanical isocenter (coll = 165)
%Afterwards it computes the distance between both
%input: the dicom image
%output: distance between both iso's, the radiation and mechanical iso,
%        radius of detected circle and field parameters for plotting
dicom = dcm;
g = dicomread(dicom);
f = medfilt2(g, 'symmetric');%noise removal
g_med = imadjust(f,stretchlim(f, []));%contrast stretching

%Size of the image (resolution)
[R, C] = size(f);

%Convert image to class uint16
o_u16 = uint16(g_med);

%Conversion factor in mm per pixel
metadata = dicominfo(dicom);
conversiefactor = metadata.ImagePlanePixelSpacing(1) * (metadata.RadiationMachineSAD /
metadata.RTImageSID);

%Find left field edge
[ x_links1, y_links1 ] = veldrand_recht(o_u16, [C/3 C/5], R/5, R/2.2, 1);

x_links3 = x_links1;
x_mu1 = mean(x_links3);
stdev1 = std(x_links3);
ind1 = find(abs(x_links3)<(x_mu1 - 3*stdev1));
x_links3(ind1) = [];
x_links4 = x_links3;

```

```

ind2l = find(abs(x_links4)>(x_mu1 + 3*stdevl));
x_links4(ind2l) = [];
x_links = x_links4;
y_links3 = y_links1;
y_links3(indl) = [];
y_links3(ind2l) = [];
y_links = y_links3;
p_links = polyfit(x_links,y_links,1);
syms x ;
yfit_links(x) = p_links(1)*x + p_links(2);

%Find right field edge
[ x_rechts1, y_rechts1 ] = veldrand_recht(o_u16, [2.3*C/3.5 5*C/6], R/1.8, 3.3*R/4, 0);

x_rechts3 = x_rechts1;
x_mur = mean(x_rechts3);
stdevr = std(x_rechts3);
indr = find(abs(x_rechts3)<(x_mur - 3*stdevr));
x_rechts3(indr) = [];
x_rechts4 = x_rechts3;
ind2r = find(abs(x_rechts4)>(x_mur + 3*stdevr));
x_rechts4(ind2r) = [];
x_rechts = x_rechts4;
y_rechts3 = y_rechts1;
y_rechts3(indr) = [];
y_rechts3(ind2r) = [];
y_rechts = y_rechts3;
p_rechts = polyfit(x_rechts,y_rechts,1);
yfit_rechts(x) = p_rechts(1)*x + p_rechts(2);

%Find top field edge
[x_boven1, y_boven1] = veldrand_liggend(o_u16, [R/30 R/6], C/3, C/2.1, 1);
[x_boven2, y_boven2] = veldrand_liggend(o_u16, [R/10 R/4], 1.8*C/3, C/1.32, 1);
y_boven3 = [y_boven1 y_boven2];
y_mu3 = mean(y_boven3);
stdevb = std(y_boven3);
indb = find(abs(y_boven3)<(y_mu3 - 3*stdevb));
y_boven3(indb) = [];
y_boven4 = y_boven3;
ind2b = find(abs(y_boven4)>(y_mu3 + 3*stdevb));
y_boven4(ind2b) = [];
y_boven = y_boven4;
x_boven3 = [x_boven1 x_boven2];
x_boven3(indb) = [];
x_boven3(ind2b) = [];
x_boven = x_boven3;
p_boven = polyfit(x_boven,y_boven,1);
yfit_boven(x) = p_boven(1)*x + p_boven(2);

%Find bottom field edge
[x_onder1, y_onder1] = veldrand_liggend(o_u16, [9*R/10 3.7*R/5], C/4, C/2.5, 0);
[x_onder2, y_onder2] = veldrand_liggend(o_u16, [29*R/30 2.5*R/3], C/1.9, 2*C/3, 0);
y_onder3 = [y_onder1 y_onder2];
y_mu3 = mean(y_onder3);
stdevo = std(y_onder3);
indo = find(abs(y_onder3)<(y_mu3 - 3*stdevo));
y_onder3(indo) = [];
y_onder4 = y_onder3;
ind2o = find(abs(y_onder4)>(y_mu3 + 3*stdevo));
y_onder4(ind2o) = [];

```

```

y_onder = y_onder4;
x_onder3 = [x_onder1 x_onder2];
x_onder3(indo) = [];
x_onder3(ind2o) = [];
x_onder = x_onder3;
p_onder = polyfit(x_onder,y_onder,1);
yfit_onder(x) = p_onder(1)*x + p_onder(2);

%Find intersections of field edges
[x1,y1] = vindSnijpunt(yfit_links, yfit_boven);
[x2,y2] = vindSnijpunt(yfit_rechts, yfit_boven);
[x3,y3] = vindSnijpunt(yfit_links, yfit_onder);
[x4,y4] = vindSnijpunt(yfit_rechts, yfit_onder);

veld = [x1 y1 x2 y2 x3 y3 x4 y4];%save field parameters for plotting

%Find intersection of diagonals (= radiation isocenter)
x_straliso = vindSnijpiso_x(x1, y1, x2, y2, x3, y3, x4, y4);
y_straliso = vindSnijpiso_y(x1, y1, x2, y2, x3, y3, x4, y4);

%edge detection
[~, threshold] = edge(f, 'sobel');
fudgeFactor = .5;
BWS = edge(f,'sobel', threshold * fudgeFactor);

%Morphological closing
Bwdfill = imclose(BWS, strel('disk', 1));

se = strel('disk', 1);%set up structuring element
erode = imerode(Bwdfill, se);%erosion
dilate = imdilate(erode, se);%dilation

%Detect all circles and their centroids and radii
stats = regionprops('table',dilate,'Centroid',...
'MajorAxisLength','MinorAxisLength');
centers = stats.Centroid;%get centroids
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);%diameters
radii = diameters/2;%radii
[maxRad, ind] = max(radii);%discard the biggest circle
radii(ind) = [];
centers(ind,:) = [];

mechiso = [];
zoekcrit = 0.5;
done = 0;
it = 0;
%Iterative process which expands the ROI untill one circle is detected
while done == 0
    %Set up ROI
    r_roi = double(y_straliso);
    c_roi = double(x_straliso);
    a = zoekcrit/conversiefactor;

    r_roi_array = [(r_roi)-a ; (r_roi)-a; (r_roi)+a; (r_roi)+a];
    c_roi_array = [(c_roi)-a ; (c_roi)+a; (c_roi)+a; (c_roi)-a];
    xq = centers(:,1);
    yq = centers(:,2);
    in = inpolygon(xq, yq, c_roi_array, r_roi_array);
    %Find centers in ROI
    x_mechiso = xq(in);
    y_mechiso = yq(in);

```

```

mechiso = [x_mechiso y_mechiso];
[ro, co] = size(mechiso);
if ro == 1
    done = 1;
end
if it > 10
    done = 1;
end
zoekcrit = zoekcrit*2;
it = it + 1;
end

%Find radius of detected circle
ind1 = find(xq == x_mechiso);
ind2 = find(yq == y_mechiso);
ind = intersect(ind1,ind2);
radius = radii(ind);

%Compute distance between rad/mech isocenter in mm
delta_iso = afstand2p(x_mechiso, x_straliso, y_mechiso, y_straliso);
deltaiso = delta_iso*conversiefactor;

end

```

11.2.7 Field edge algorithm (horizontal and vertical)

```

function [x_waarden, y_waarden] = veldrand_liggend(I, yv, xk1, xk2,reverse)
%Function to search horizontal field-edges
%Inputs: the input image, yv = length of profiles
%        xk1 & xk2 = begin en end of the moving profiles
%        reverse = 1 for top edge and 0 for bottom edge
%Output: The points that are defining the field edge

f = I;
y_prof = yv;
[R,C] = size(f);
minimum = double(min(min(f)));%get min & max intensity value
maximum = double(max(max(f)));
range = [minimum:maximum];
val = double(median(range)); %get intensity value of fwhm
y_values = [];%initialize the output arrays
x_values = [];

%Iterative process
for k = xk1 : xk2
    [cx,cy,c] = improfile(f, [k k], y_prof, 100); %create an intensity profile
    if reverse == 1 %flip arrays if reverse = 1
        cx = flip(cx);
        cy = flip(cy);
        c = flip(c);
    end
    x_values = [x_values cx(1)];%add x-value to output array
    C = c.'; %transpose c

    [Verschil index] = min(abs(C-val)); %find smallest difference from VAL and it's index.
    i = index;
    c_wl = C(i); %Get the intensity value

```

```

teller = 0;
indices = [0];

%Check if this value appears more than one time in a row and how many
while C(index) == C(index+1)
    teller = teller + 1;
    indices = [indices teller];
    index = index + 1;
end

%Get all the corresponding y-values and compute mean value
y_tot = 0;
for k = 1 : teller+1
    y_tot = y_tot + cy(i+indices(k));
end
y_gem1 = y_tot/(teller+1);

%Find a partner to interpolate to VAL
if c_w1 ~= val && c_w1 < val %if the found value is greater than VAL
    %we have to find a partner smaller than VAL

    index2 = i - 1;
    t = 0;
    indic = [0];
    i2 = index2;

    %Check if partner appears more than one time in a row and how many
    while C(index2) == C(index2-1)
        t = t+1;
        indic = [indic t];
        index2 = index2-1;
    end

    y_tot2 = 0;

    %Find corresponding y-values and compute mean value
    for k = 1 : t + 1
        y_tot2 = y_tot2 + cy(i2 - indic(k));
    end
    y_gem2 = y_tot2/(t+1);
    c_w2 = C(i2);

elseif c_w1 ~= val && c_w1 > val %if the found value is smaller than VAL
    %we have to find a partner greater than VAL

    index2 = i;
    t = 0;
    indic = [0];
    %Check if partner appears more than one time in a row and how many
    while C(index2) == C(index2+1)
        index2 = index2+1;
    end
    i2 = index2 + 1;
    i_i2 = i2;
    while C(i2) == C(i2+1)
        t = t+1;
        indic = [indic t];
        i2 = i2+1;
    end
    y_tot2 = 0;
    %Get corresponding y-values and compute mean

```



```

    for k = 1 : t+1
        y_tot2 = y_tot2 + cy(i_i2 + indic(k));
    end
    y_gem2 = y_tot2/(t+1);
    c_w2 = C(i_i2);

    elseif c_w1 == val %if found value = VAL interpolation is not needed
        y_gem2 = 0;
    end

    x_fin = 0;
    %Interpolate to VAL if needed
    if c_w1 == val;
        y_fin = y_gem1;
    elseif c_w1 ~= val && c_w1 < c_w2
        x1 = c_w1;
        x2 = c_w2;
        y1 = y_gem1;
        y2 = y_gem2;
        x = val;
        y_fin = interpoleer(x1,x2,y1,y2,x);
    elseif c_w1 ~= val && c_w1 > c_w2
        x1 = c_w2;
        x2 = c_w1;
        y1 = y_gem2;
        y2 = y_gem1;
        x = val;
        y_fin = interpoleer(x1,x2,y1,y2,x);
    end
    y_values = [y_values y_fin];
end

%Assign values to output arrays
y_waarden = y_values;
x_waarden = x_values;
end

```

```

function [ x_waarden, y_waarden ] = veldrand_recht(I, xv, yk1, yk2, reverse)
%Function to search vertical field-edges
%Inputs: the input image, xv = length of profile
%        yk1 & yk2 = begin en end of the moving profiles
%        reverse = 1 for left edge and 0 for right hand edge
%Output: The points that are defining the field edge

f = I;
x_prof = xv;
[R,C] = size(f);
minimum = double(min(min(f))); %get min & max intensity value
maximum = double(max(max(f)));
range = [minimum:maximum];
val = double(median(range)); %get intensity value of fwhm
x_values = []; %initialize the output arrays
y_values = [];

%Iterative process
for k = yk1 : yk2
    [cx,cy,c] = improfile(f, x_prof, [k k], 100); %create an intensity profile

```

```

if reverse == 1 %flip arrays if reverse = 1
    cx = flip(cx);
    cy = flip(cy);
    c = flip(c);
end
C = c.'; %transpose c
y_values = [y_values cy(1)]; %add y-value to output array

[Verschil index] = min(abs(C-val)); %find smallest difference from VAL and it's index.
i = index;
c_w1 = C(i); %Get the intensity value

teller = 0;
indices = [0];

%Check if this value appears more than one time in a row and how many
while C(index) == C(index+1)
    teller = teller + 1;
    indices = [indices teller];
    index = index + 1;
end

%Get al the corresponding x-values and compute mean value
x_tot = 0;
for k = 1 : teller+1
    x_tot = x_tot + cx(i+indices(k));
end
x_gem1 = x_tot/(teller+1);

%Find a partner to interpolate to VAL
if c_w1 ~= val && c_w1 > val %if the found value is greater than VAL
    %we have to find a partner smaller than VAL

    index2 = i - 1;
    t = 0;
    indic = [0];
    i2 = index2;
    %Check if partner appears more than one time in a row and how many
    while C(index2) == C(index2-1)
        t = t+1;
        indic = [indic t];
        index2 = index2-1;
    end

    x_tot2 = 0;

    %Find corresponding x-values and compute mean value
    for k = 1 : t + 1
        x_tot2 = x_tot2 + cx(i2 - indic(k));
    end
    x_gem2 = x_tot2/(t+1);
    c_w2 = C(i2);

elseif c_w1 ~= val && c_w1 < val %if the found value is smaller than VAL
    %we have to find a partner greater than VAL

    index2 = i;
    t = 0;
    indic = [0];
    %Check if partner appears more than one time in a row and how many

```

```

while C(index2) == C(index2+1)
    index2 = index2+1;
end
i2 = index2 + 1;
i_i2 = i2;

while C(i2) == C(i2+1)
    t = t+1;
    indic = [indic t];
    i2 = i2+1;
end
x_tot2 = 0;
%Get corresponding x-values and compute mean
for k = 1 : t+1
    x_tot2 = x_tot2 + cx(i_i2 + indic(k));
end
x_gem2 = x_tot2/(t+1);
c_w2 = C(i_i2);
%If the found value = VAL we don't have to interpolate
elseif c_w1 == val
    x_gem2 = 0;
end

%Interpolate to VAL
x_fin = 0;
if c_w1 == val;
    x_fin = x_gem1;
elseif c_w1 ~= val && c_w1 < c_w2
    x1 = c_w1;
    x2 = c_w2;
    y1 = x_gem1;
    y2 = x_gem2;
    x = val;
    x_fin = interpoleer(x1,x2,y1,y2,x);
elseif c_w1 ~= val && c_w1 > c_w2
    x1 = c_w2;
    x2 = c_w1;
    y1 = x_gem2;
    y2 = x_gem1;
    x = val;
    x_fin = interpoleer(x1,x2,y1,y2,x);
end
x_values = [x_values x_fin]; %add found x-value to output array
end

%Assign values to the output arrays
x_waarden = x_values;
y_waarden = y_values;

end

```

11.5.8 Deviations on table position indicators (3 dimensions)

```
function [ vertAfw, longVer, x_p1, y_p1, x_p2, y_p2, radius1, radius2 ] = Vertical( dcm1, dcm2 )
```

```
%Function to calculate the vertical table motion
%Inputs: original image at G270 and image after 15 cm vertical displacement
%Outputs: vertical deviation of the motion, the longitudinal displacement
%of the table, centers and radius of the two detected dots.
dicom = dcm1;
dicom2 = dcm2;
f = dicomread(dicom);
g = dicomread(dicom2);

metadata1 = dicominfo(dcm1);
metadata2 = dicominfo(dcm2);

%Get size (resolution) of images
[R, C] = size(f);
[R2, C2] = size(g);

%Initialize conversionfactor
conversionfactor = metadata1.ImagePlanePixelSpacing(1) * (metadata1.RadiationMachineSID / metadata1.RTImageSID); %amount of mm's per pixel

%Fuse both images
c = imfuse(f,g,'blend','scaling','joint');

%Edge detection
[~, threshold] = edge(c, 'sobel');
fudgeFactor = .5;
BWS = edge(c,'sobel', threshold * fudgeFactor);

%Morphological closing/erosion/dilation
Bwdfill = imclose(BWS, strel('disk', 1));

Bwdfill = imerode(Bwdfill, strel('disk', 1));

Bwdfill = imdilate(Bwdfill, strel('disk', 1));

%Detect all circles/ellipses together with their centroids and radii
stats = regionprops('table',Bwdfill,'Centroid',...
    'MajorAxisLength','MinorAxisLength');
centers = stats.Centroid; %get center of circles
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2); %calculate diameters
radii = diameters/2; %calc. radii
radii_boven10 = radii > 10; %Circles with radii > 10 are neglected
k = find(radii_boven10);
radii(k) = [];
centers(k,:) = [];

%Find 1st circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopVerticalPosition < metadata2.TableTopVerticalPosition %(+)
if C == 512
    r_roi = 330;
    c_roi = 147;
    a = 1;
elseif C == 1024
    r_roi = 661;
```

```

    c_roi = 298;
    a = 1; %
end
elseif metadata1.TableTopVerticalPosition > metadata2.TableTopVerticalPosition %(-)
if C == 512
    r_roi = 329;
    c_roi = 76;
    a = 1;
elseif C == 1024
    r_roi = 329*2;
    c_roi = 76*2;
    a = 1;
end
end

done = 0;
it = 0;
%Iterative process which expands ROI untill one circle is detected
while done == 0
    r_roi_array = [(r_roi)-a ; (r_roi)-a; (r_roi)+a; (r_roi)+a];
    c_roi_array = [(c_roi)-a ; (c_roi)+a; (c_roi)+a; (c_roi)-a];

    xq = centers(:,1);
    yq = centers(:,2);
    in = inpolygon(xq, yq, c_roi_array, r_roi_array);
    %Find centers in ROI
    x_p1 = xq(in);
    y_p1 = yq(in);
    center1 = [x_p1 y_p1];
    [ro, co] = size(center1);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a = a*2;
    it = it + 1;
end

%Find radius of detected circle (only needed for visualisation purposes)
ind1 = find(xq == x_p1);
ind2 = find(yq == y_p1);
ind = intersect(ind1,ind2);
radius1 = radii(ind);

%Find 2nd circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopVerticalPosition < metadata2.TableTopVerticalPosition %(+)
if C == 512
    r_roi2 = 331;
    c_roi2 = 433;
    a2 = 1;
elseif C == 1024
    r_roi2 = 660;
    c_roi2 = 882;
    a2 = 1;
end
elseif metadata1.TableTopVerticalPosition > metadata2.TableTopVerticalPosition %(-)
if C == 512

```

```

    r_roi2 = 328;
    c_roi2 = 364;
    a2 = 1;
elseif C == 1024
    r_roi2 = 328*2;
    c_roi2 = 364*2;
    a2 = 1;
end
end

done = 0;
it = 0;
%Iterative proces which expands ROI untill one circle is detected
while done == 0
    r_roi_array2 = [(r_roi2)-a2 ; (r_roi2)-a2; (r_roi2)+a2; (r_roi2)+a2];
    c_roi_array2 = [(c_roi2)-a2 ; (c_roi2)+a2; (c_roi2)+a2; (c_roi2)-a2];

    xq2 = centers(:,1);
    yq2 = centers(:,2);
    in = inpolygon(xq2, yq2, c_roi_array2, r_roi_array2);
    %Find centers in ROI
    x_p2 = xq2(in);
    y_p2 = yq2(in);
    center2 = [x_p2 y_p2];
    [ro, co] = size(center2);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a2 = a2*2;
    it = it + 1;
end

%Find radius of detected circle
ind3 = find(xq2 == x_p2);
ind4 = find(yq2 == y_p2);
ind_2 = intersect(ind3,ind4);
radius2 = radii(ind_2);

%Compute vertical and longitudinal distance between both centroids of both circles
if x_p1 > x_p2
    verschVert = abs(x_p1 - x_p2)*conversiefactor;
else
    verschVert = abs(x_p2 - x_p1)*conversiefactor;
end
if y_p1 > y_p2
    longVer = abs(y_p1 - y_p2)*conversiefactor;
else
    longVer = abs(y_p2 - y_p1)*conversiefactor;
end
vertAfw = 150 - verschVert; %Compute deviation
end

```

```
function [ AfwLong, verschLat, x_p1, y_p1, x_p2, y_p2, radius1, radius2 ] = Longitudinaal(
dcm1, dcm2 )
```

```
%Function to calculate the longitudinal table motion
%Inputs: original image at G0 and image after 15 cm LNG displacement
%Outputs: LNG deviation of the motion & the lateral displacement
%of the table, centers and radius of the two detected dots.
```

```
dicom = dcm1;
dicom2 = dcm2;
f = dicomread(dicom);
g = dicomread(dicom2);
metadata1 = dicominfo(dcm1);
metadata2 = dicominfo(dcm2);
```

```
%size of the images
[R, C] = size(f);
[R2, C2] = size(g);
```

```
%Conversionfactor in mm per pixel
conversionfactor = metadata1.ImagePlanePixelSpacing(1) * (metadata1.RadiationMachineSAD /
metadata1.RTImageSID);
```

```
%Fuse both images
c = imfuse(f,g,'blend','scaling','joint');
```

```
%Edge detection
[~, threshold] = edge(c, 'sobel');
fudgeFactor = .5;
Bws = edge(c,'sobel', threshold * fudgeFactor);
```

```
%Morphological closing/erosion/dilation
Bwdfill = imclose(Bws, strel('disk', 1));

Bwdfill = imerode(Bwdfill, strel('disk', 1));

Bwdfill = imdilate(Bwdfill, strel('disk', 1));
```

```
%Detect all circles/ellipses together with their centroids and radii
stats = regionprops('table',Bwdfill,'Centroid',...
'MajorAxisLength','MinorAxisLength');
centers = stats.Centroid;%get centers
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);%diameters
radii = diameters/2;%radii
radii_boven10 = radii > 10;%discard radii greater than 10
k = find(radii_boven10);
radii(k) = [];
centers(k,:) = [];
```

```
%Find 1st circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopLongitudinalPosition < metadata2.TableTopLongitudinalPosition %(+)
if C == 512
r_roi = 300;
c_roi = 392;
a = 1;
elseif C == 1024
r_roi = 14;
c_roi = 784;
a = 1;
```

```

end
elseif metadata1.TableTopLongitudinalPosition > metadata2.TableTopLongitudinalPosition %(-)
if C == 512
    r_roi = 85;
    c_roi = 393;
    a = 1;
elseif C == 1024
    r_roi = 85*2;
    c_roi = 393*2;
    a = 1;
end
end

done = 0;
it = 0;
%Iterative process which expands ROI untill one circle is detected
while done == 0
    r_roi_array = [(r_roi)-a ; (r_roi)-a; (r_roi)+a; (r_roi)+a];
    c_roi_array = [(c_roi)-a ; (c_roi)+a; (c_roi)+a; (c_roi)-a];

    xq = centers(:,1);
    yq = centers(:,2);
    in = inpolygon(xq, yq, c_roi_array, r_roi_array);
    %Find centers in ROI
    x_p1 = xq(in);
    y_p1 = yq(in);
    center1 = [x_p1 y_p1];
    [ro co] = size(center1);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a = a*2;
    it = it+1;
end

%Find radius of detected circle (only needed for visualisation purposes)
ind1 = find(xq == x_p1)
ind2 = find(yq == y_p1)
ind = intersect(ind1,ind2)
radius1 = radii(ind)

%Find 2nd circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopLongitudinalPosition < metadata2.TableTopLongitudinalPosition %(+)
if C == 512
    r_roi2 = 15;
    c_roi2 = 392;
    a2 = 1;
elseif C == 1024
    r_roi2 = 596;
    c_roi2 = 789;
    a2 = 1;
end
elseif metadata1.TableTopLongitudinalPosition > metadata2.TableTopLongitudinalPosition %(-)
if C == 512
    r_roi2 = 371;
    c_roi2 = 393;

```



```

a2 = 1;
elseif C == 1024
    r_roi2 = 371*2;
    c_roi2 = 393*2;
    a2 = 1;
end
end

done = 0;
it = 0;
%Iterative proces which expands ROI untill one circle is detected
while done == 0
    r_roi_array2 = [(r_roi2)-a2 ; (r_roi2)-a2; (r_roi2)+a2; (r_roi2)+a2];
    c_roi_array2 = [(c_roi2)-a2 ; (c_roi2)+a2; (c_roi2)+a2; (c_roi2)-a2];

    xq2 = centers(:,1);
    yq2 = centers(:,2);
    in = inpolygon(xq2, yq2, c_roi_array2, r_roi_array2);
    %Find centers in ROI
    x_p2 = xq2(in);
    y_p2 = yq2(in);
    center2 = [x_p2 y_p2];
    [ro, co] = size(center2);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a2 = a2*2;
    it = it+1;
end

%Find radius of detected circle
ind3 = find(xq2 == x_p2);
ind4 = find(yq2 == y_p2);
ind_2 = intersect(ind3,ind4);
radius2 = radii(ind_2);

%Compute vertical and longitudinal distance between both centroids of both circles
if y_p1 > y_p2
    verschLong = abs(y_p1 - y_p2)*conversiefactor;
else
    verschLong = abs(y_p2 - y_p1)*conversiefactor;
end
if x_p1 > x_p2
    verschLat = abs(x_p1 - x_p2)*conversiefactor;
else
    verschLat = abs(x_p2 - x_p1)*conversiefactor;
end
AfwLong = 150 - verschLong; %Compute deviation
end

```

```
function [ afwLat, verschLong, x_p1, y_p1, x_p2, y_p2, radius1, radius2 ] = Lateraal( dcm1,
dcm2 )
```

```
%Function to calculate the lateral table motion
%Inputs: original image at G0 and image after 15 cm LAT displacement
%Outputs: LAT deviation of the motion & the longitudinal displacement
%of the table, centers and radius of the two detected dots.
dicom = dcm1;
dicom2 = dcm2;
f = dicomread(dicom);
g = dicomread(dicom2);

metadata1 = dicominfo(dcm1);
metadata2 = dicominfo(dcm2);

%size of the images
[R, C] = size(f);
[R2, C2] = size(g);

%Conversionfactor in mm per pixel
conversionfactor = metadata1.ImagePlanePixelSpacing(1) * (metadata1.RadiationMachineSAD /
metadata1.RTImageSID);

%Fuse both images
c = imfuse(f,g,'blend','scaling','joint');

%Edge detection
[~, threshold] = edge(c, 'sobel');
fudgeFactor = .5;
BWS = edge(c,'sobel', threshold * fudgeFactor);

%Morphological closing/erosion/dilation
Bwdfill = imclose(BWS, strel('disk', 1));

Bwdfill = imerode(Bwdfill, strel('disk', 1));

Bwdfill = imdilate(Bwdfill, strel('disk', 1));

%Detect all circles/ellipses together with their centroids and radii
stats = regionprops('table',Bwdfill,'Centroid',...
'MajorAxisLength','MinorAxisLength');
centers = stats.Centroid;
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);%diameters
radii = diameters/2;%radii
radii_boven10 = radii > 10;%discard radii greater than 10
k = find(radii_boven10);
radii(k) = [];
centers(k,:) = [];

%Find 1st circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopLateralPosition < metadata2.TableTopLateralPosition %(+)
if C == 512
r_roi = 329;
c_roi = 147;
a = 1;
elseif C == 1024
r_roi = 604;
c_roi = 198;
```

```

    a = 1;
end
elseif metadata1.TableTopLateralPosition > metadata2.TableTopLateralPosition %(-)
if C == 512
    r_roi = 330;
    c_roi = 73;
    a = 1;
elseif C == 1024
    r_roi = 330*2;
    c_roi = 73*2;
    a = 1;
end
end

done = 0;
it = 0;
%Iterative process which expands ROI untill one circle is detected
while done == 0
    r_roi_array = [(r_roi)-a ; (r_roi)-a; (r_roi)+a; (r_roi)+a];
    c_roi_array = [(c_roi)-a ; (c_roi)+a; (c_roi)+a; (c_roi)-a];

    xq = centers(:,1);
    yq = centers(:,2);
    in = inpolygon(xq, yq, c_roi_array, r_roi_array);
    %Find centers in ROI
    x_p1 = xq(in);
    y_p1 = yq(in);
    center1 = [x_p1 y_p1];
    [ro co] = size(center1);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a = a*2;
    it = it + 1;
end

%Find radius of detected circle
ind1 = find(xq == x_p1);
ind2 = find(yq == y_p1);
ind = intersect(ind1,ind2);
radius1 = radii(ind);

%Find 2nd circle
%ROI differs for positive vs. negative table motion
if metadata1.TableTopLateralPosition < metadata2.TableTopLateralPosition %(+)
if C == 512
    r_roi2 = 329;
    c_roi2 = 434;
    a2 = 1;
elseif C == 1024
    r_roi2 = 596;
    c_roi2 = 789;
    a2 = 1;
end
elseif metadata1.TableTopLateralPosition > metadata2.TableTopLateralPosition %(-)
if C == 512
    r_roi2 = 331;

```

```

    c_roi2 = 361;
    a2 = 1;
elseif C == 1024
    r_roi2 = 331*2;
    c_roi2 = 361*2;
    a2 = 1;
end
end

done = 0;
it = 0;
%Iterative proces which expands ROI untill one circle is detected
while done == 0
    r_roi_array2 = [(r_roi2)-a2 ; (r_roi2)-a2; (r_roi2)+a2; (r_roi2)+a2];
    c_roi_array2 = [(c_roi2)-a2 ; (c_roi2)+a2; (c_roi2)+a2; (c_roi2)-a2];

    xq2 = centers(:,1);
    yq2 = centers(:,2);
    in = inpolygon(xq2, yq2, c_roi_array2, r_roi_array2);
    %Find centers in ROI
    x_p2 = xq2(in);
    y_p2 = yq2(in);
    center2 = [x_p2 y_p2];
    [ro, co] = size(center2);
    if ro == 1
        done = 1;
    end
    if it > 10
        done = 1;
    end
    a2 = a2*2;
    it = it + 1
end

%Find radius of detected circle
ind3 = find(xq2 == x_p2);
ind4 = find(yq2 == y_p2);
ind_2 = intersect(ind3,ind4);
radius2 = radii(ind_2);

%Compute vertical and longitudinal distance between both centroids of both circles
if y_p1 > y_p2
    verschLong = abs(y_p1 - y_p2)*conversiefactor;
else y_p2 > y_p1
    verschLong = abs(y_p2 - y_p1)*conversiefactor;
end
if x_p1 > x_p2
    verschLat = abs(x_p1 - x_p2)*conversiefactor;
else
    verschLat = abs(x_p2 - x_p1)*conversiefactor;
end
afwLat = 150 - verschLat;
end

```

11.2.9 Deviation on asymmetrical field size parameters

```
function [ fsx1, fsx2, fsxtot, fsy1, fsy2, fsytot, veld ] = veldgrootte(dcm, x_stral, y_stral)
%This function calculates the assymetrical fieldsize
%Inputs: The dicom image and the radiation isocenter
%Outputs: Assymetrical fieldsizes x1,x2,y1 and y2
%       Total fieldsize in x and y direction
%       veld = nessecary parameters to plot the field
x_straliso = x_stral;
y_straliso = y_stral;
dicom = dcm;
metadata1 = dicominfo(dcm);
g = dicomread(dicom);
f = medfilt2(g, 'symmetric'); %noise removal
g_med = imadjust(f,stretchlim(f),[]); %Contrast stretching

%Get size of image (resolution)
[R, C] = size(f);

%Convert class to uint16
o_u16 = uint16(g_med);

%Conversionfactor for distance measurements (in mm per pixel)
conversiefactor = metadata1.ImagePlanePixelSpacing(1) * (metadata1.RadiationMachinesAD /
metadata1.RTImageSID);

%Find left field edge
[ x_links1, y_links1 ] = veldrand_recht(o_u16, [0 C/2], 115*R/300, 185*R/300, 1);

%Following lines removes outliers from dataset when they are situated
%further than 3 times a stdev from the mean
x_links3 = x_links1;
    x_mu1 = mean(x_links3);%mean
    stdev1 = std(x_links3);%stdev
    ind1 = find(abs(x_links3)<(x_mu1 - 3*stdev1));%detect outliers
    x_links3(ind1) = [];%remove outliers
x_links4 = x_links3;
    ind21 = find(abs(x_links4)>(x_mu1 + 3*stdev1));
    x_links4(ind21) = [];
x_links = x_links4;
%Also remove the corresponding y-values
y_links3 = y_links1;
    y_links3(ind1) = [];
    y_links3(ind21) = [];
y_links = y_links3;
%Fit a linear function through the remaining points (X & Y are reversed)
p_links = polyfit(y_links,x_links,1);
syms x ;
yfit_links(x) = p_links(1)*x + p_links(2);%Horizontal fit
yfit_linksinv(x) = finverse(yfit_links(x));%inverse function yields the
                                %required vertical fit

%Find right field edge
[ x_rechts1, y_rechts1 ] = veldrand_recht(o_u16, [C/2 C], 115*R/300, 185*R/300, 0);
x_rechts3 = x_rechts1;
    x_mur = mean(x_rechts3);
    stdevr = std(x_rechts3);
    indr = find(abs(x_rechts3)<(x_mur - 3*stdevr));
    x_rechts3(indr) = [];
```

```

x_rechts4 = x_rechts3;
ind2r = find(abs(x_rechts4)>(x_mur + 3*stdevr));
x_rechts4(ind2r) = [];
x_rechts = x_rechts4;
y_rechts3 = y_rechts1;
y_rechts3(indr) = [];
y_rechts3(ind2r) = [];
y_rechts = y_rechts3;
p_rechts = polyfit(y_rechts,x_rechts,1);
yfit_rechts(X) = p_rechts(1)*X + p_rechts(2);
yfit_rechtsinv(X) = finverse(yfit_rechts(X));

%Find top field edge
[x_boven1, y_boven1] = velrand_liggend(o_u16, [0 R/2], 123*C/300, 177*C/300, 1);
y_boven3 = y_boven1;
y_mu_b = mean(y_boven3);
stdevb = std(y_boven3);
indb = find(abs(y_boven3)<(y_mu_b - 3*stdevb));
y_boven3(indb) = [];
y_boven4 = y_boven3;
ind2b = find(abs(y_boven4)>(y_mu_b + 3*stdevb));
y_boven4(ind2b) = [];
y_boven = y_boven4;
x_boven3 = x_boven1;
x_boven3(indb) = [];
x_boven3(ind2b) = [];
x_boven = x_boven3;
p_boven = polyfit(x_boven,y_boven,1);
yfit_boven(X) = p_boven(1)*X + p_boven(2);

%Find bottom field edge
[x_onder1, y_onder1] = velrand_liggend(o_u16, [R/2 R], 123*C/300, 177*C/300, 0);
y_onder3 = y_onder1;
y_mu_o = mean(y_onder3);
stdevo = std(y_onder3);
indo = find(abs(y_onder3)<(y_mu_o - 3*stdevo));
y_onder3(indo) = [];
y_onder4 = y_onder3;
ind2o = find(abs(y_onder4)>(y_mu_o + 3*stdevo));
y_onder4(ind2o) = [];
y_onder = y_onder4;
x_onder3 = x_onder1;
x_onder3(indo) = [];
x_onder3(ind2o) = [];
x_onder = x_onder3;
p_onder = polyfit(x_onder,y_onder,1);
yfit_onder(X) = p_onder(1)*X + p_onder(2);

%Calculate the intersections (corners) of the field edges
[x1,y1] = vindSnijpunt(yfit_linksinv, yfit_boven);
[x2,y2] = vindSnijpunt(yfit_rechtsinv, yfit_boven);
[x3,y3] = vindSnijpunt(yfit_linksinv, yfit_onder);
[x4,y4] = vindSnijpunt(yfit_rechtsinv, yfit_onder);

veld = [x1 y1 x2 y2 x3 y3 x4 y4]; %Save field corners for visualisation

%Find assymetric fieldsizes in y direction
[grootte_boven, grootte_onder] = AssymVeldgrootte_y(yfit_boven, yfit_onder, x_straliso,
y_straliso);
fsy1 = double(grootte_boven*conversiefactor);

```

```

fsy2 = double(grootte_onder*conversiefactor);
fsytot = fsy1 + fsy2;

%Find assymmetric fieldsizes in x direction
[grootte_links ,grootte_rechts] = AssymVeldgrootte_x(yfit_linksinv, yfit_rechtsinv,
x_straliso, y_straliso);
fsx1 = grootte_links*conversiefactor;
fsx2 = grootte_rechts*conversiefactor;
fsxtot = fsx1 + fsx2;

end

```

```

function [ x_links, x_rechts ] = AssymVeldgrootte_x(vg11, vg12, x, y)
%This function computes the assymetrical field size in y direction
%Input: the two function which define both horizontal field edges
%       the reference point (averaged radiation isocenter)
%Output: both assymetrical fieldsizes
afstand_l = [];
afstand_r = [];

syms f(x)
f(x) = vg11;
x1 = solve(f(x) == y, x);
y1 = y;

x2 = x;
y2 = y;

syms g(x)
g(x) = vg12;
x3 = solve(g(x) == y, x);
y3 = y;

afst_l = afstand2p(x1, x2, y1, y2);
afst_r = afstand2p(x3, x2, y3, y2);

afstand_l = [afstand_l afst_l];
afstand_r = [afstand_r afst_r];

x_links = mean(afstand_l);
x_rechts = mean(afstand_r);
end

```

```

function [ y_boven, y_onder ] = AssymVeldgrootte_y(vg11, vg12, x, y)
%This function computes the assymetrical field size in y direction
%Input: the two function which define both horizontal field edges
%       the reference point (averaged radiation isocenter)
%Output: both assymetrical fieldsizes
afstand_b = [];
afstand_o = [];

syms f(x)
f(x) = vg11;
x1 = x;
y1 = f(x);

x2 = x;
y2 = y;

syms g(x)
g(x) = vg12;
x3 = x;
y3 = g(x);

afst_b = afstand2p(x1, x2, y1, y2);
afst_o = afstand2p(x3, x2, y3, y2);

afstand_b = [afstand_b afst_b];
afstand_o = [afstand_o afst_o];

y_boven = mean(afstand_b);
y_onder = mean(afstand_o);
end

```

11.2.10 Concatenating PDF files¹³

```

%APPEND_PDFS Appends/concatenates multiple PDF files
%
% Example:
%   append_pdfs(output, input1, input2, ...)
%   append_pdfs(output, input_list{:})
%   append_pdfs test.pdf temp1.pdf temp2.pdf
%
% This function appends multiple PDF files to an existing PDF file, or
% concatenates them into a PDF file if the output file doesn't yet exist.
%
% This function requires that you have ghostscript installed on your
% system. Ghostscript can be downloaded from: http://www.ghostscript.com
%
% IN:
%   output - string of output file name (including the extension, .pdf).
%           If it exists it is appended to; if not, it is created.
%   input1 - string of an input file name (including the extension, .pdf).
%           All input files are appended in order.
%   input_list - cell array list of input file name strings. All input
%               files are appended in order.

```

¹³ These functions are original courtesy of Oliver Woodford (2011).


```

% Copyright: Oliver Woodford, 2011

% Thanks to Reinhard Knoll for pointing out that appending multiple pdfs in
% one go is much faster than appending them one at a time.

% Thanks to Michael Teo for reporting the issue of a too long command line.
% Issue resolved on 5/5/2011, by passing gs a command file.

% Thanks to Martin Wittmann for pointing out the quality issue when
% appending multiple bitmaps.
% Issue resolved (to best of my ability) 1/6/2011, using the prepress
% setting

% 26/02/15: If temp dir is not writable, use the output folder for temp
%           files when appending (Javier Paredes); sanity check of inputs

function append_pdfs(varargin)

if nargin < 2, return; end % sanity check

% Are we appending or creating a new file
append = exist(varargin{1}, 'file') == 2;
output = [tempname '.pdf'];
try
    % Ensure that the temp dir is writable (Javier Paredes 26/2/15)
    fid = fopen(output, 'w');
    fwrite(fid, 1);
    fclose(fid);
    delete(output);
    isTempDirOk = true;
catch
    % Temp dir is not writable, so use the output folder
    [dummy, fname, fext] = fileparts(output); %#ok<ASGLU>
    fpath = fileparts(varargin{1});
    output = fullfile(fpath, [fname fext]);
    isTempDirOk = false;
end
if ~append
    output = varargin{1};
    varargin = varargin(2:end);
end
% Create the command file
if isTempDirOk
    cmdfile = [tempname '.txt'];
else
    cmdfile = fullfile(fpath, [fname '.txt']);
end
fh = fopen(cmdfile, 'w');
fprintf(fh, '-q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dPDFSETTINGS=/prepress -sOutputFile="%s"
-f', output);
fprintf(fh, ' "%s"', varargin{:});
fclose(fh);
% Call ghostscript
ghostscript(['@"' cmdfile '"']);
% Delete the command file
delete(cmdfile);
% Rename the file if needed
if append
    movefile(output, varargin{1});

```

```
end
end
```

```
function varargout = ghostscript(cmd)
%GHOSTSCRIPT Calls a local GhostScript executable with the input command
%
% Example:
% [status result] = ghostscript(cmd)
%
% Attempts to locate a ghostscript executable, finally asking the user to
% specify the directory ghostscript was installed into. The resulting path
% is stored for future reference.
%
% Once found, the executable is called with the input command string.
%
% This function requires that you have Ghostscript installed on your
% system. You can download this from: http://www.ghostscript.com
%
% IN:
% cmd - Command string to be passed into ghostscript.
%
% OUT:
% status - 0 iff command ran without problem.
% result - Output from ghostscript.

% Copyright: Oliver Woodford, 2009-2015, Yair Altman 2015-
%{
% Thanks to Jonas Dorn for the fix for the title of the uigetdir window on Mac OS.
% Thanks to Nathan Childress for the fix to default location on 64-bit windows systems.
% 27/04/11 - Find 64-bit Ghostscript on windows. Thanks to Paul Durack and
%         Shaun Kline for pointing out the issue
% 04/05/11 - Thanks to David Chorlian for pointing out an alternative
%         location for gs on linux.
% 12/12/12 - Add extra executable name on windows. Thanks to Ratish
%         Punnoose for highlighting the issue.
% 28/06/13 - Fix error using GS 9.07 in Linux. Many thanks to Jannick
%         Steinbring for proposing the fix.
% 24/10/13 - Fix error using GS 9.07 in Linux. Many thanks to Johannes
%         for the fix.
% 23/01/14 - Add full path to ghostscript.txt in warning. Thanks to Koen
%         Vermeer for raising the issue.
% 27/02/15 - If Ghostscript croaks, display suggested workarounds
% 30/03/15 - Improved performance by caching status of GS path check, if ok
% 14/05/15 - Clarified warning message in case GS path could not be saved
% 29/05/15 - Avoid cryptic error in case the ghostscript path cannot be saved (issue #74)
% 10/11/15 - Custom GS installation webpage for MacOS. Thanks to Andy Hueni via FEX
%}

    try
        % Call ghostscript
        [varargout{1:nargout}] = system([gs_command(gs_path()) cmd]);
    catch err
        % Display possible workarounds for Ghostscript croaks
        url1 = 'https://github.com/altmany/export_fig/issues/12#issuecomment-61467998'; %
        issue #12
        url2 = 'https://github.com/altmany/export_fig/issues/20#issuecomment-63826270'; %
        issue #20
    end
end
```

```

hg2_str = ''; if using_hg2, hg2_str = ' or Matlab R2014a'; end
fprintf(2, 'Ghostscript error. Rolling back to GS 9.10%s may possibly solve this:\n *
<a href="%s">%s</a> ',hg2_str,url1,url1);
if using_hg2
    fprintf(2, '(GS 9.10)\n * <a href="%s">%s</a> (R2014a)',url2,url2);
end
fprintf('\n\n');
if ismac || isunix
    url3 = 'https://github.com/altmany/export_fig/issues/27'; % issue #27
    fprintf(2, 'Alternatively, this may possibly be due to a font path issue:\n * <a
href="%s">%s</a>\n\n',url3,url3);
    % issue #20
    fpath = which(mfilename);
    if isempty(fpath), fpath = [mfilename('fullpath') '.m']; end
    fprintf(2, 'Alternatively, if you are using csh, modify shell_cmd from "export..."
to "setenv ..."\nat the bottom of <a
href="matlab:opentoline('%s',174)">%s</a>\n\n',fpath,fpath);
end
rethrow(err);
end
end

function path_ = gs_path
% Return a valid path
% Start with the currently set path
path_ = user_string('ghostscript');
% Check the path works
if check_gs_path(path_)
    return
end
% Check whether the binary is on the path
if ispc
    bin = {'gswin32c.exe', 'gswin64c.exe', 'gs'};
else
    bin = {'gs'};
end
for a = 1:numel(bin)
    path_ = bin{a};
    if check_store_gs_path(path_)
        return
    end
end
% Search the obvious places
if ispc
    default_location = 'C:\Program Files\gs\';
    dir_list = dir(default_location);
    if isempty(dir_list)
        default_location = 'C:\Program Files (x86)\gs\'; % Possible location on 64-bit
systems
        dir_list = dir(default_location);
    end
    executable = {'\bin\gswin32c.exe', '\bin\gswin64c.exe'};
    ver_num = 0;
    % If there are multiple versions, use the newest
    for a = 1:numel(dir_list)
        ver_num2 = sscanf(dir_list(a).name, 'gs%g');
        if ~isempty(ver_num2) && ver_num2 > ver_num
            for b = 1:numel(executable)
                path2 = [default_location dir_list(a).name executable{b}];
                if exist(path2, 'file') == 2

```

```

        path_ = path2;
        ver_num = ver_num2;
    end
end
end
end
if check_store_gs_path(path_)
    return
end
else
    executable = {'/usr/bin/gs', '/usr/local/bin/gs'};
    for a = 1:numel(executable)
        path_ = executable{a};
        if check_store_gs_path(path_)
            return
        end
    end
end
% Ask the user to enter the path
while true
    if strcmp(computer, 'MAC', 3) % Is a Mac
        % Give separate warning as the uigetdir dialogue box doesn't have a
        % title
        uiwait(warndlg('Ghostscript not found. Please locate the program.'))
    end
    base = uigetdir('/', 'Ghostscript not found. Please locate the program. ');
    if isequal(base, 0)
        % User hit cancel or closed window
        break;
    end
    base = [base filesep]; %#ok<AGROW>
    bin_dir = {'', ['bin' filesep], ['lib' filesep]};
    for a = 1:numel(bin_dir)
        for b = 1:numel(bin)
            path_ = [base bin_dir{a} bin{b}];
            if exist(path_, 'file') == 2
                if check_store_gs_path(path_)
                    return
                end
            end
        end
    end
end
end
if ismac
    error('Ghostscript not found. Have you installed it
(http://pages.uoregon.edu/koch?');
else
    error('Ghostscript not found. Have you installed it from www.ghostscript.com?');
end
end

function good = check_store_gs_path(path_)
% Check the path is valid
good = check_gs_path(path_);
if ~good
    return
end
% Update the current default path to the path found
if ~user_string('ghostscript', path_)
    filename = fullfile(fileparts(which('user_string.m')), '.ignore', 'ghostscript.txt');

```

```

        warning('Path to ghostscript installation could not be saved in %s (perhaps a
permissions issue). You can manually create this file and set its contents to %s, to improve
performance in future invocations (this warning is safe to ignore).', filename, path_);
        return
    end
end

function good = check_gs_path(path_)
    persistent isOk
    if isempty(path_)
        isOk = false;
    elseif ~isequal(isOk,true)
        % Check whether the path is valid
        [status, message] = system([gs_command(path_) '-h']); %#ok<ASGLU>
        isOk = status == 0;
    end
    good = isOk;
end

function cmd = gs_command(path_)
    % Initialize any required system calls before calling ghostscript
    % TODO: in Unix/Mac, find a way to determine whether to use "export" (bash) or "setenv"
(csh/tcsh)
    shell_cmd = '';
    if isunix
        shell_cmd = 'export LD_LIBRARY_PATH=""; '; % Avoids an error on Linux with GS 9.07
    end
    if ismac
        shell_cmd = 'export DYLD_LIBRARY_PATH=""; '; % Avoids an error on Mac with GS 9.07
    end
    % Construct the command string
    cmd = sprintf('%s"%s" ', shell_cmd, path_);
end

```

```

function string = user_string(string_name, string)
%USER_STRING Get/set a user specific string
%
% Examples:
% string = user_string(string_name)
% isSaved = user_string(string_name, new_string)
%
% Function to get and set a string in a system or user specific file. This
% enables, for example, system specific paths to binaries to be saved.
%
% The specified string will be saved in a file named <string_name>.txt,
% either in a subfolder named .ignore under this file's folder, or in the
% user's prefdir folder (in case this file's folder is non-writable).
%
% IN:
% string_name - String containing the name of the string required, which
%               sets the filename storing the string: <string_name>.txt
% new_string - The new string to be saved in the <string_name>.txt file
%
% OUT:
% string - The currently saved string. Default: ''
% isSaved - Boolean indicating whether the save was succesful

```

```

% Copyright (C) Oliver Woodford 2011-2014, Yair Altman 2015-

% This method of saving paths avoids changing .m files which might be in a
% version control system. Instead it saves the user dependent paths in
% separate files with a .txt extension, which need not be checked in to
% the version control system. Thank you to Jonas Dorn for suggesting this
% approach.

% 10/01/2013 - Access files in text, not binary mode, as latter can cause
% errors. Thanks to Christian for pointing this out.
% 29/05/2015 - Save file in prefdir if current folder is non-writable (issue #74)

if ~ischar(string_name)
    error('string_name must be a string.');
```

```

end
% Create the full filename
fname = [string_name '.txt'];
dname = fullfile(fileparts(mfilename('fullpath')), '.ignore');
file_name = fullfile(dname, fname);
if nargin > 1
    % Set string
    if ~ischar(string)
        error('new_string must be a string.');
```

```

end
% Make sure the save directory exists
%dname = fileparts(file_name);
if ~exist(dname, 'dir')
    % Create the directory
    try
        if ~mkdir(dname)
            string = false;
            return
        end
    catch
        string = false;
        return
    end
    % Make it hidden
    try
        fileattrib(dname, '+h');
```

```

    catch
    end
end
% write the file
fid = fopen(file_name, 'wt');
if fid == -1
    % file cannot be created/updated - use prefdir if file does not already exist
    % (if file exists but is simply not writable, don't create a duplicate in prefdir)
    if ~exist(file_name, 'file')
        file_name = fullfile(prefdir, fname);
        fid = fopen(file_name, 'wt');
```

```

    end
    if fid == -1
        string = false;
        return;
    end
end
try
    fprintf(fid, '%s', string);
catch
```

```

        fclose(fid);
        string = false;
        return
    end
    fclose(fid);
    string = true;
else
    % Get string
    fid = fopen(file_name, 'rt');
    if fid == -1
        % file cannot be read, try to read the file in prefdir
        file_name = fullfile(prefdir, fname);
        fid = fopen(file_name, 'rt');
        if fid == -1
            string = '';
            return
        end
    end
    string = fgetl(fid);
    fclose(fid);
end
end
end

```

```

%USING_HG2 Determine if the HG2 graphics engine is used
%
%   tf = using_hg2(fig)
%
%IN:
%   fig - handle to the figure in question.
%
%OUT:
%   tf - boolean indicating whether the HG2 graphics engine is being used
%        (true) or not (false).

% 19/06/2015 - Suppress warning in R2015b; cache result for improved performance
% 06/06/2016 - Fixed issue #156 (bad return value in R2016b)
%
%This function is original courtesy of Oliver Woodford, 2011

function tf = using_hg2(fig)
    persistent tf_cached
    if isempty(tf_cached)
        try
            if nargin < 1, fig = figure('visible','off'); end
            oldwarn = warning('off','MATLAB:graphicsversion:GraphicsVersionRemoval');
            try
                % This generates a [supressed] warning in R2015b:
                tf = ~graphicsversion(fig, 'handlegraphics');
            catch
                tf = ~verLessThan('matlab','8.4'); % =R2014b
            end
            warning(oldwarn);
        catch
            tf = false;
        end
        if nargin < 1, delete(fig); end
        tf_cached = tf;
    end
end

```

```

else
    tf = tf_cached;
end
end

```

11.2.11 Other functions

```

function [ afstand ] = afstand2p( x1, x2, y1, y2 )
%This function computes the distance between two points
%Input: the two points
%Output: the distance between both points

afstand = double(sqrt((((x2-x1)^2)+((y2-y1)^2))));

end

```

```

function [] = deleteifexist(fileloc)
% Deletes file if it already exists
% Input: fileloc = relative path to file

%Set up warning (e.g. if the file is opened MATLAB will deny permission)
w = warning('error','MATLAB:DELETE:Permission');
try
    if (exist(fileloc, 'file') == 2)
        delete(fileloc);
    end
catch ME
    error(ME.message);
end
warning(w);

end

```

```

function [ interp ] = interpoleer( x1, x2, y1, y2, x )
%This function linearly interpolates to a target x value
%input: 2 points to define the function y(x), a target x value
%output: The interpolated y-value
interp = y1 + ((y2-y1)/(x2-x1))*(x-x1);
end

```



```

function [ xsnijpunt ] = vindSnijpiso_x(P1x, P1y, P2x, P2y, P3x, P3y, P4x, P4y)
%Finds the y-coordinate of the intersection point of the diagonals that
%define the radiation isocenter
%inputs: the four points that define the two diagonals
%outputs: the y-coordinate of intersection
syms x;
y1 = (((P4y-P1y)/(P4x-P1x))*(x-P1x)) + P1y;
y2 = (((P2y-P3y)/(P2x-P3x))*(x-P3x)) + P3y;
eqn = y1 == y2;
xsnijpunt = solve(eqn, x);
end

```

```

function [ ysnijpunt ] = vindSnijpiso_y(P1x, P1y, P2x, P2y, P3x, P3y, P4x, P4y)
%Finds the y-coordinate of the intersection point of the diagonals that
%define the radiation isocenter
%inputs: the four points that define the two diagonals
%outputs: the y-coordinate of intersection
syms y;
x1 = (((P4x-P1x)/(P4y-P1y))*(y-P1y)) + P1x;
x2 = (((P2x-P3x)/(P2y-P3y))*(y-P3y)) + P3x;
eqn = x1 == x2;
ysnijpunt = solve(eqn, y);
end

```

```

function [ x_snijpunt, y_snijpunt ] = vindSnijpunt(vg1, vg2)
%Finds intersection (x,y) of two equations
%inputs : 2 equations
%outputs : x&y coordinate of intersection
syms X
eqn = vg1 == vg2;
x_snijpunt = solve(eqn, X);
y_snijpunt = vg1(x_snijpunt);

end

```

11.3 Appendix C: Example of output file

In this appendix a few pages of the output file are given. An output file always starts with a table on the first page which contains a summarize of the results. The following pages contain all the processed output images together with a detailed image. The example starts below. It contains one example of each operation.

i	Gantry	Collimat	Table	deltalso
1	0	360	360	0,358
2	0	90	360	0,719
3	0	165	360	0,714
4	0	270	360	0,244
5	0	360	90	0,533
6	0	360	270	1,332
7	180	360	0	1,364
8	90	90	0	0,939
9	90	360	0	1,161
10	90	270	0	1,053
11	270	270	0	0,952
12	270	0	0	1,13
13	270	90	0	1,3

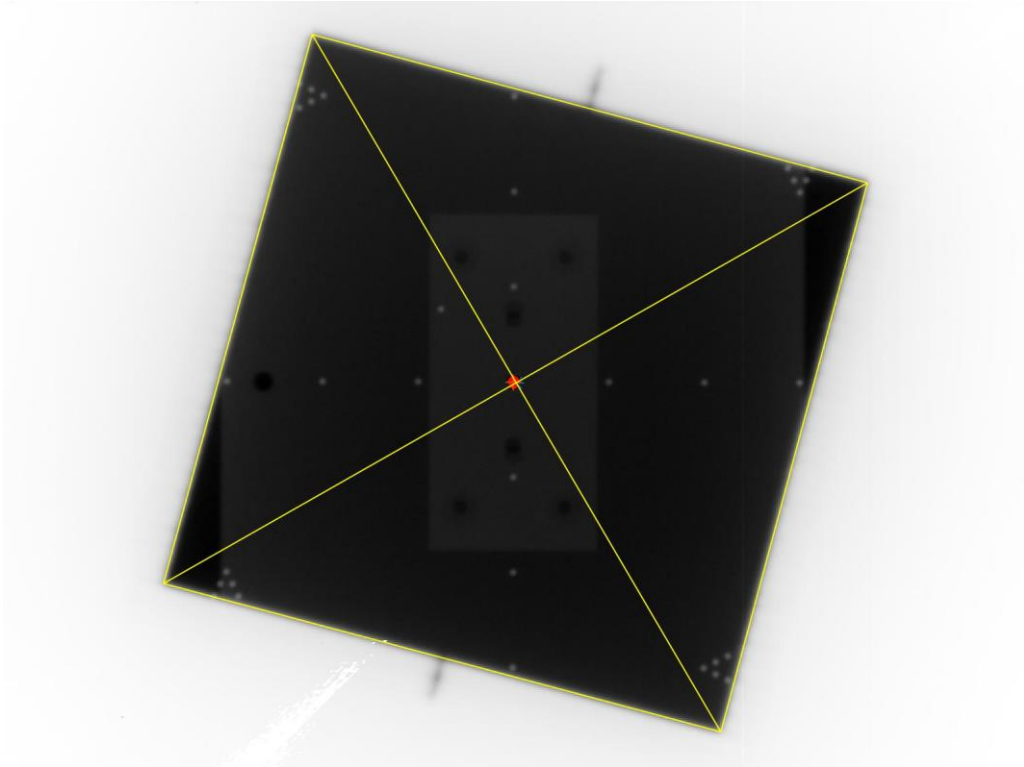
VRT15 VRT_dev Long_dev
done -0,425 0,105

LNG15 LNG_dev Lat_dev
done 0,41 0,627

LAT15 LAT_dev Long_dev
done 0,254 0,261

field	x1_dev	x2_dev	y1_dev	y2_dev	x_total_dev	y_total_dev
5x5:	0,367	0,257	0,556	0,047	0,623	0,603
10x10:	-0,469	-0,382	0,709	0,488	-0,852	1,197
18x18:	0,108	0,637	0,645	0,546	0,744	1,191

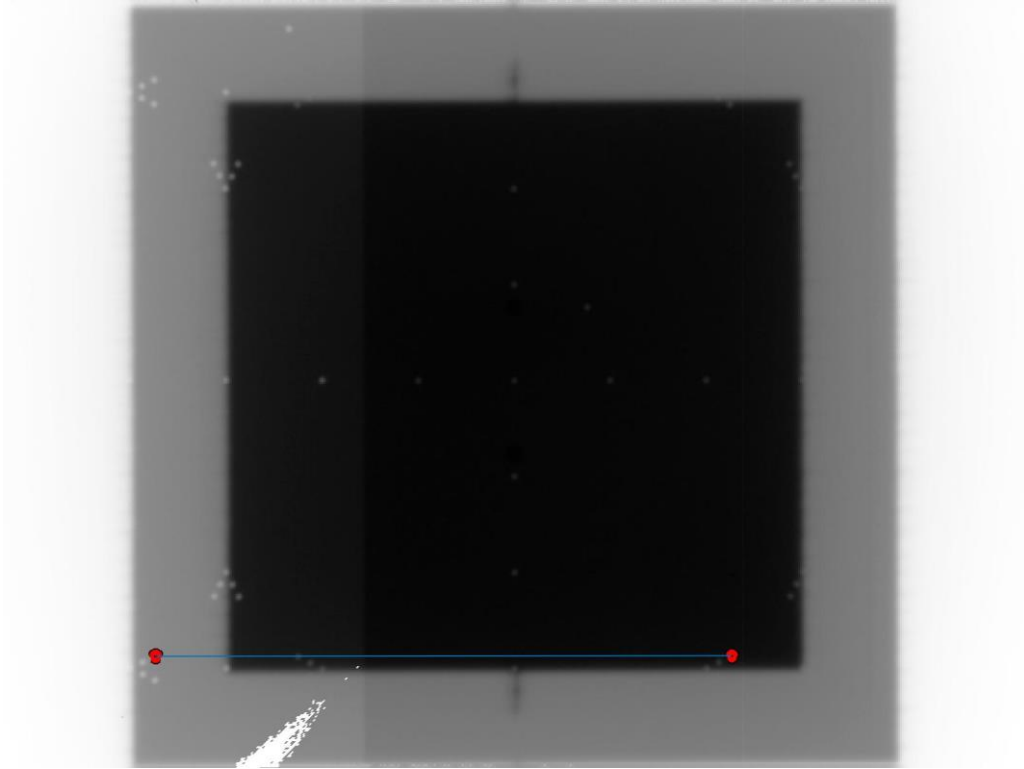
3: G: 0.00°, C: 165.00° en T: 359.96°
; distance between mechanical and radiation isocenter = 0.714mm.



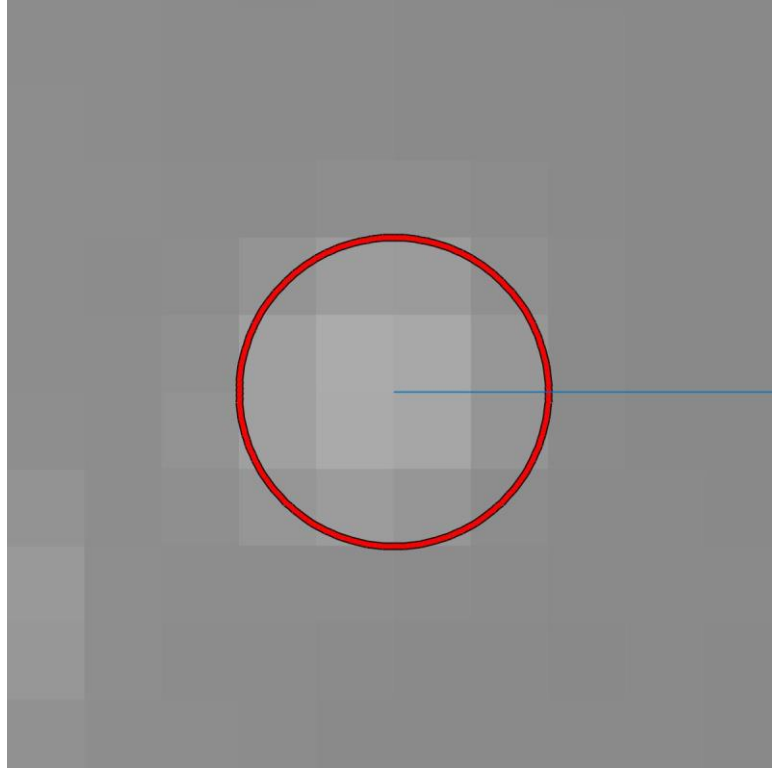
3: G: 0.00°, C: 165.00° en T: 359.96°
; distance between mechanical and radiation isocenter = 0.714mm.



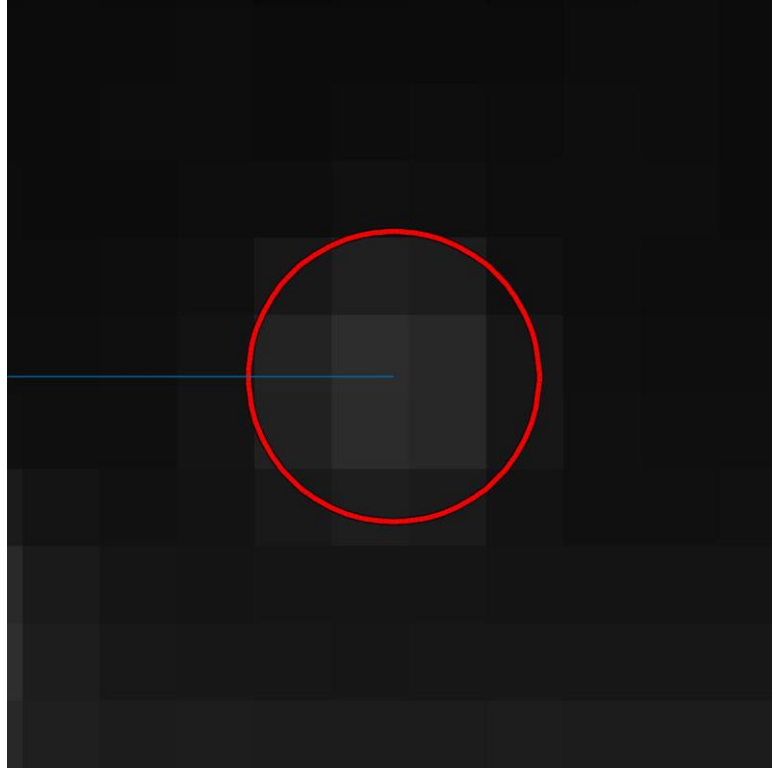
The vertical deviation is -0.42mm The longitudinal displacement is 0.10mm.



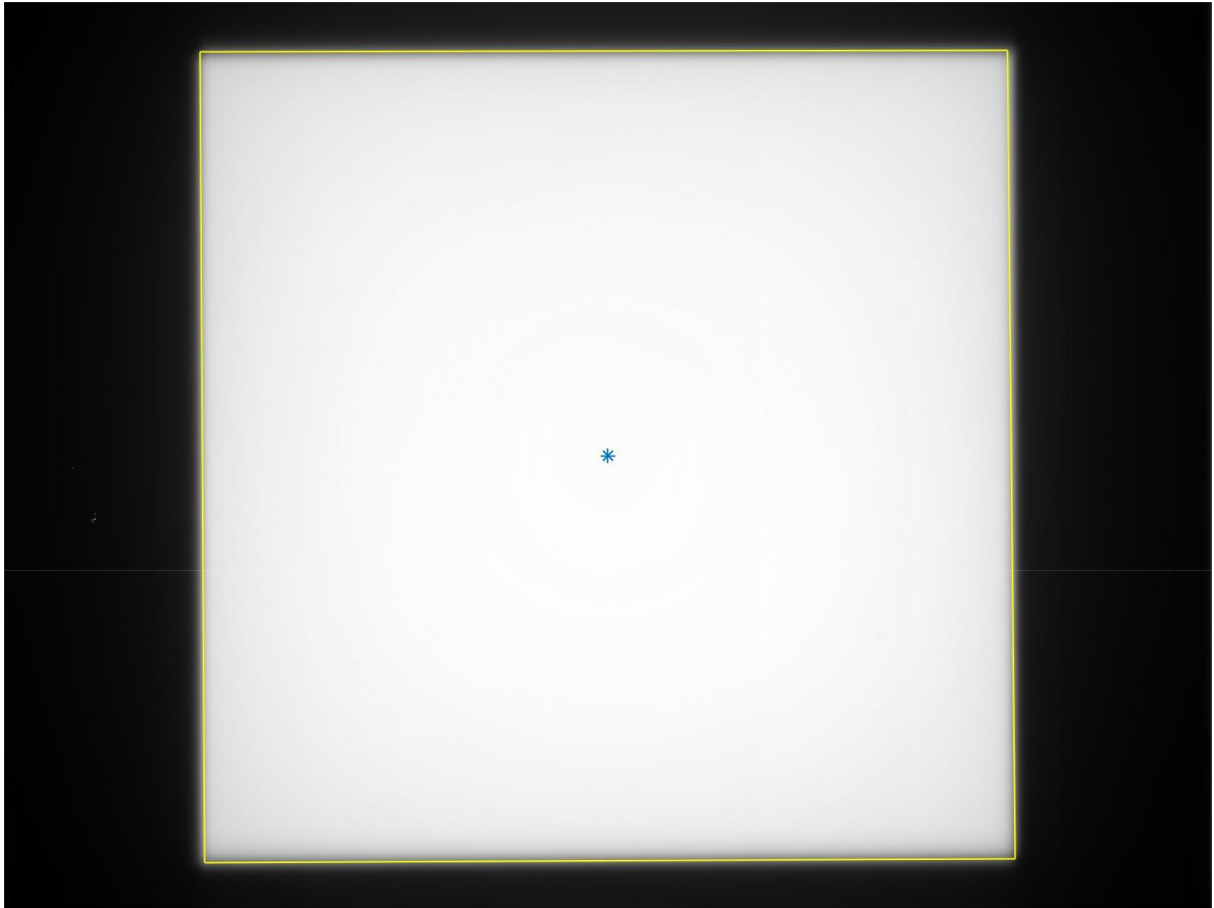
The vertical deviation is -0.42mm The longitudinal displacement is 0.10mm.



The vertical deviation is -0.42mm The longitudinal displacement is 0.10mm.



18x18: x1 is 89.89mm and x2 is 89.36mm y1
is 89.36mm and y2 is 89.45mm
Total field size x: 179.26mm and total field size y: 178.81mm



Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Automatization of mechanical quality assurance of a linac using MATLAB

Richting: **master in de industriële wetenschappen: nucleaire technologie-nucleaire technieken / medisch nucleaire technieken**
Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Smeulders, Jelle

Datum: **6/06/2017**