Development of a workflow for generating clinically deliverable spatio-temporally fractionated radiotherapy plans using the eclipse scripting API

Florian Dietsche

Master thesis

Department of Physics, University of Zürich

Department of Radiation Oncology, University Hospital Zürich and University of Zürich

Supervised by

Nathan Torelli

Prof. Jan Unkelbach

March 21, 2023

Contents

1. Introduction

Radiotherapy is one of the main pillars of cancer treatment. After a patient is diagnosed with cancer and referred for radiotherapy, a treatment plan is generated, which aims at delivering a high radiation dose to the tumour whilst sparing the surrounding healthy tissue as good as possible [1]. In conventional clinical practice, the entire treatment planning process is performed using commercial treatment planning systems (TPS). Such commercial TPS offer certified medical approaches, but do not support latest research.

Research software, on the other hand, are often open-source and can thus be more easily extended to address research questions. The in-house treatment planning software "opt4D" used at the University Hospital of Zurich offers features to investigate new, non-standard planning approaches. As an example, spatio-temporal fractionation schemes involve the delivery of different dose distributions in different fractions in order to optimally exploit the fractionation effects. An ideal spatio-temporally fractionated treatment should hypo-fractionate the tumour as well as deliver a near-uniform fractionation to the healthy tissue. Such a plan can be obtained by simultaneously optimizing multiple dose distributions based on their cumulative biologically effective dose (BED). As previous in-silico studies [2-4] yielded promising results, it is of interest to generate spatio-temporally fractionated treatment plans which can be delivered in the clinics. This may allow to perform a clinical trial for testing the feasibility of spatio-temporally fractionated treatments.

To deliver a plan with a commercial linac, xml-files, containing the plan information, are needed [5]. Thus, it is of interest to import treatment plans made in research software to a commercial TPS, which are capable of directly generating the needed files. Importing plans calculated in opt4D thus allows delivering them, which is a necessity for a clinical trial.

In this thesis, a workflow has been developed to generate clinically deliverable plans, starting from treatment plans generated in the research TPS. This workflow was implemented using the commercial Eclipse TPS, alongside the corresponding Eclipse scripting API (ESAPI). The workflow was validated for a patient with prostate cancer using conventional uniformly fractionated treatments and also used to generate clinically deliverable STF plans for a patient with a large H&N tumour.

2. Workflow

A workflow has been implemented which allows to generate clinically deliverable plans in Eclipse starting from plans obtained in the in-house TPS opt4D. The workflow, whose main steps are schematically illustrated on the flowchart in Figure 1, starts with the creation of a plan based on CT and structure set files exported from Eclipse, and incorporates calculating all individual apertures in Eclipse, exporting the resulting dose and additional quantities from Eclipse, performing data preparation to generate files for a weight re-optimization, and uploading the resulting weight re-optimized plan into Eclipse.

Both technical aspects of the implementation and the intended use of the workflow will be highlighted in the following sections.

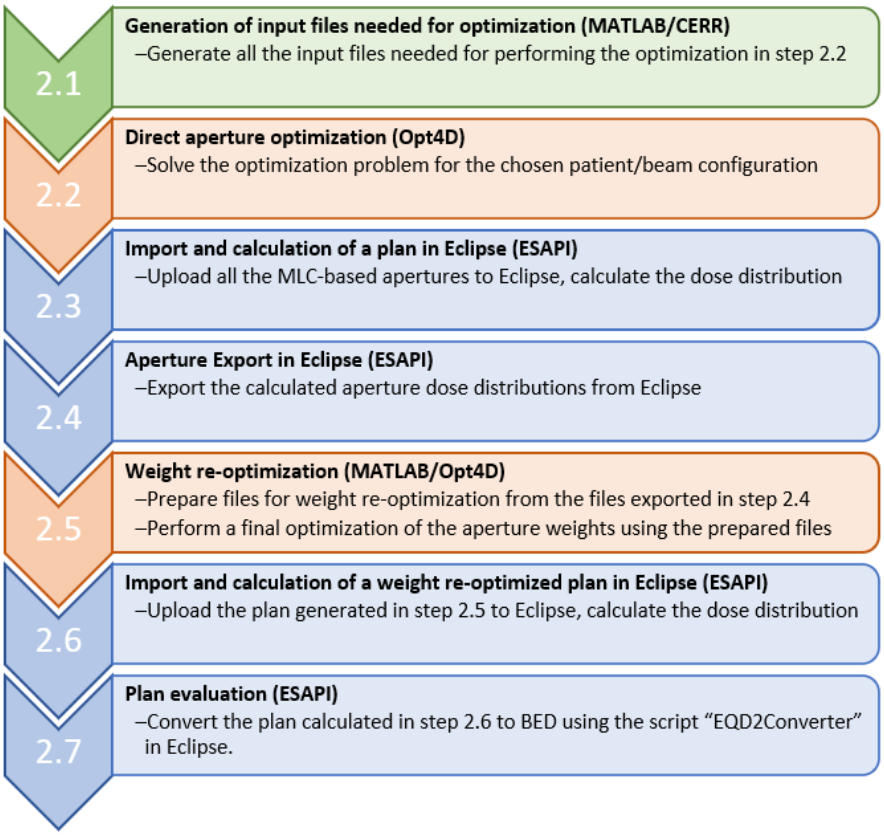


Figure 1: Flowchart of the workflow

2.1. Generation of input files needed for optimization

First of all, DICOM files for the CT and structure set have to be exported from Eclipse. These files can be imported into the open-source research software CERR, which is in-

terfaced with MATLAB, where the imported files are also visualised. CERR can then be used to calculate the dose-influence matrices, structure VOIs, patient geometry information files and distance files, which are needed as input for performing an optimization in opt4D. The resulting patient geometry files can be stored as .dif files, whilst the .dij files store the beamlet dose distributions.

The original implementation for writing .dif files was modified in this thesis to also store the information about the isocentre position of the treatment based on the CERR coordinate system. This information is necessary later on in the workflow to set the treatment isocenter position in Eclipse.

The structure VOIs store information about the shape of the structures present in the patient in 3D matrix format. These files are used in opt4D to determine which structure a voxel belongs to, and thus which objective function applies to said voxel. The pre-existing MATLAB implementation to generate these files was not modified for this thesis. The files containing this information will also be referred to as vvfiles during this thesis.

An additional function necessary for the workflow is the command *find_distance_file*, which calculates the distance of each voxel from the target. This version of the function was based on an existing MATLAB function, and thus its implementation will be discussed later on when discussing the weight re-optimization step. These distance files are important for normal tissue objectives (NTO) objectives, which penalize doses above a value which depends on the distance of a voxel from the PTV edge.

2.1.1. Workflow description

First, the RTSTRUCT and CT DICOM files need to be exported from Eclipse. This can be achieved by opening the patient, right-clicking the CT and selecting the option Export -> PATIENT File Export Filter. This step is illustrated in Figure 2.

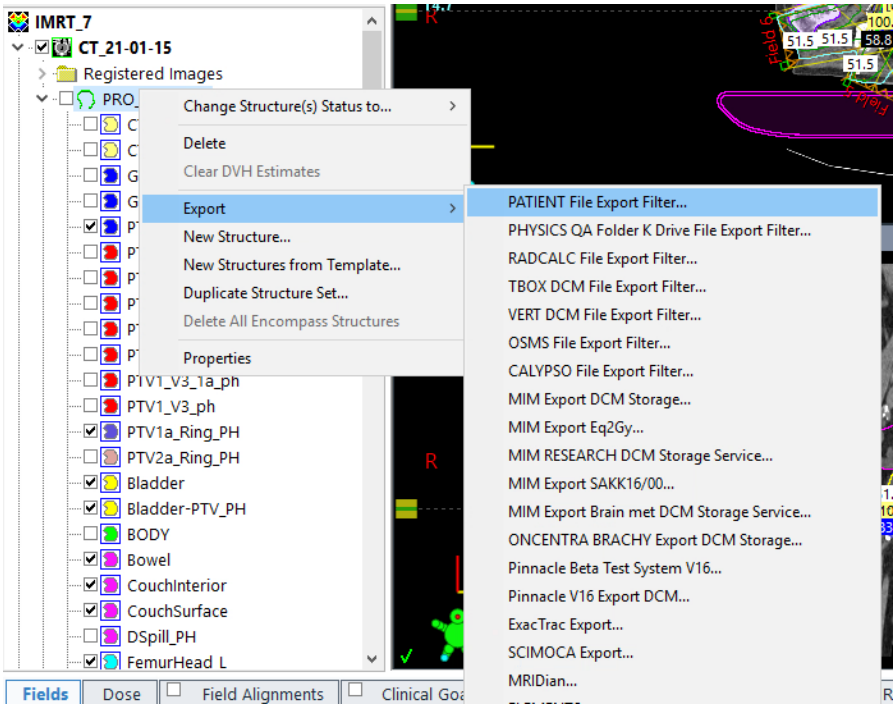


Figure 2: CT and structure set export in Eclipse

A window will be opened by Eclipse. Here, the directory for the output can be chosen, see Figure 3 for reference.

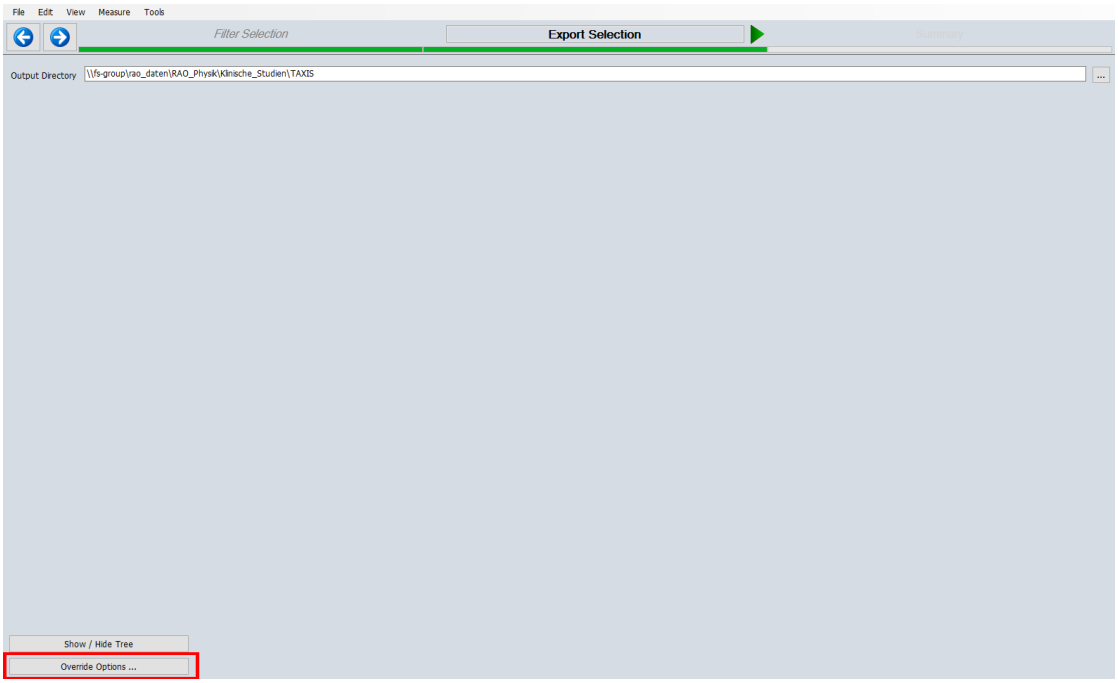


Figure 3: Export window in Eclipse

Before the output can be generated, the options have to be adjusted. To do so, one can press the "Override Options ..." button on the lower left corner. An additional window will open upon pressing this button.

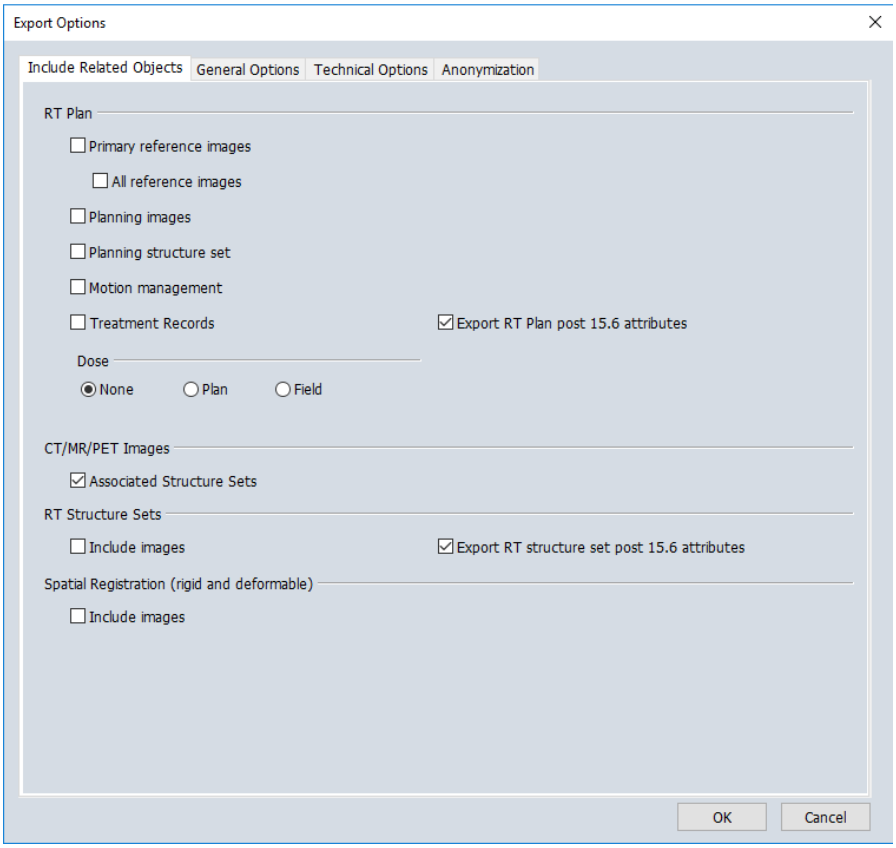


Figure 4: Override the "Include Related Objects" options in Eclipse

In the tab "Include Related Object" (Figure 4), the option "None" should be chosen for dose, while the box below "CT/MT/PET Images" should be checked to include associated structures. Then, one should switch to the "Anonymization" tab (Figure 5).

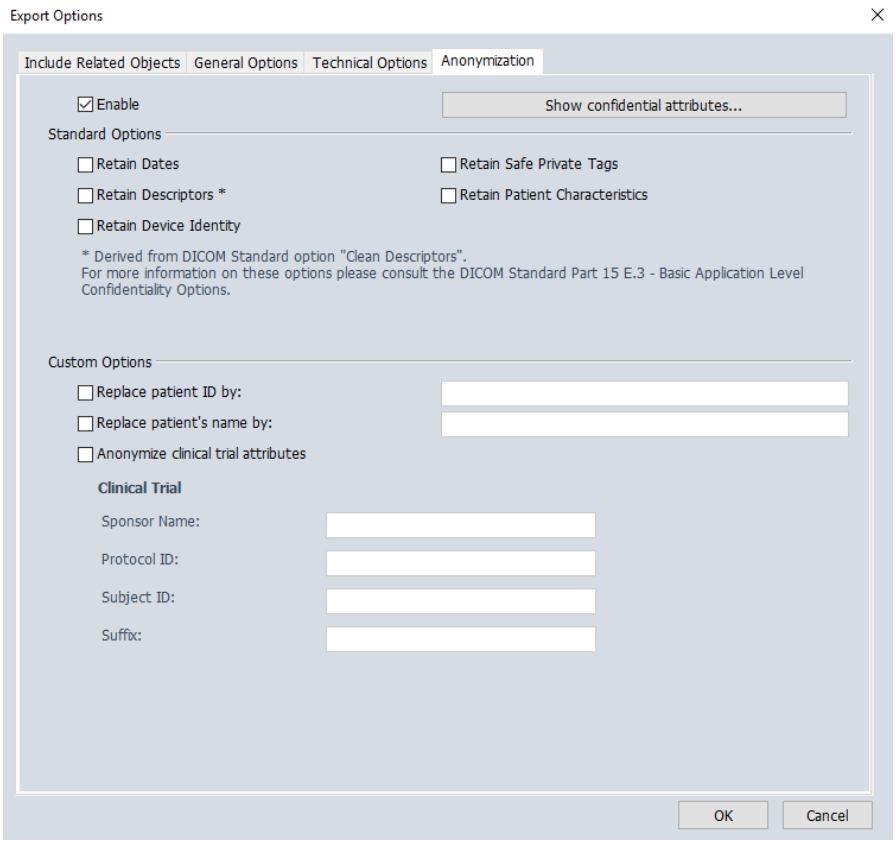


Figure 5: Enable anonymization for the export

In this tab, the "Enable" option should be checked by clicking it. In the custom options, the field "Replace patient ID by:" needs to be checked, with the naming convention "zzz_ChosenName". The same steps should be applied to the field "Replace patient's name by:". Once this is done, the "OK" button can be pressed to export the files to the chosen location.

Afterwards, CERR has to be opened by typing *CERR* into MATLAB's command line. A GUI should appear (Figure 6), where the DICOM batch import format needs to be chosen from the dropdown menu in the Import section. Now, the folder containing the RTSTRUCT and CT DICOM files can be selected.

Figure 6: CERR GUI after being started from the command line

CERR will now import the DICOM files and convert them to a patient file, which can then be saved as a .mat file. Once this is done, CERR can be re-opened. This time, the study option should be chosen, where one can now import the previously generated patient file, as seen in Figure 7.

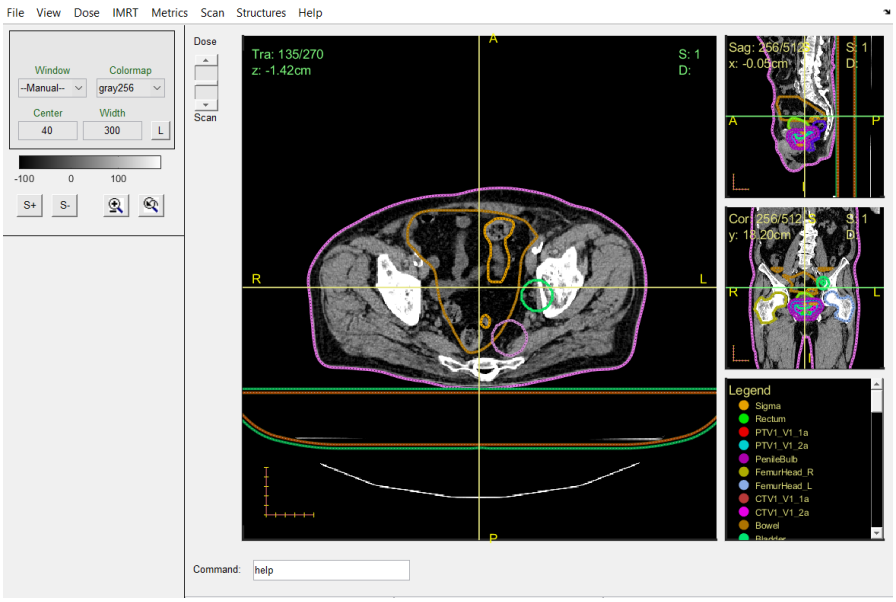


Figure 7: CERR GUI after the created .mat file has been opened

In a next step, the IMRTP tab should to be opened, as seen in Figure 8, where one can define a new IMRTPlan. Here, single fields with specific angles can be created, as well as multiple fields with equispaced distances. Each field contains multiple parameters (e.g. gantry angle, couch angle, etc.).

If the Dij's are created to be used in a VMAT plan, it is important to keep Eclipse's limitations in mind. In Eclipse, gantry angles are given in a range of [0,360), and full rotations are not possible. As an example, if a plan should be made with 181 control points between the angles -180° - 180° in CW direction, the beam angles should instead be defined between the angles 180.1° - 179.9°.

For this workflow, it is important that the isocentre position is set to be the centre of mass (COM) as seen in Figure 8, and that both the couch and collimator angle are set to the respective values. The gantry angle should correspond to one of the desired beam angles for the optimisation step in opt4D. Also important to note is that one should choose $\triangle$X and $\triangle$Y to be 0.5 cm, which is usually done automatically in the case of equispaced beams. These values denote the width of the beamlets in both X and Y direction.

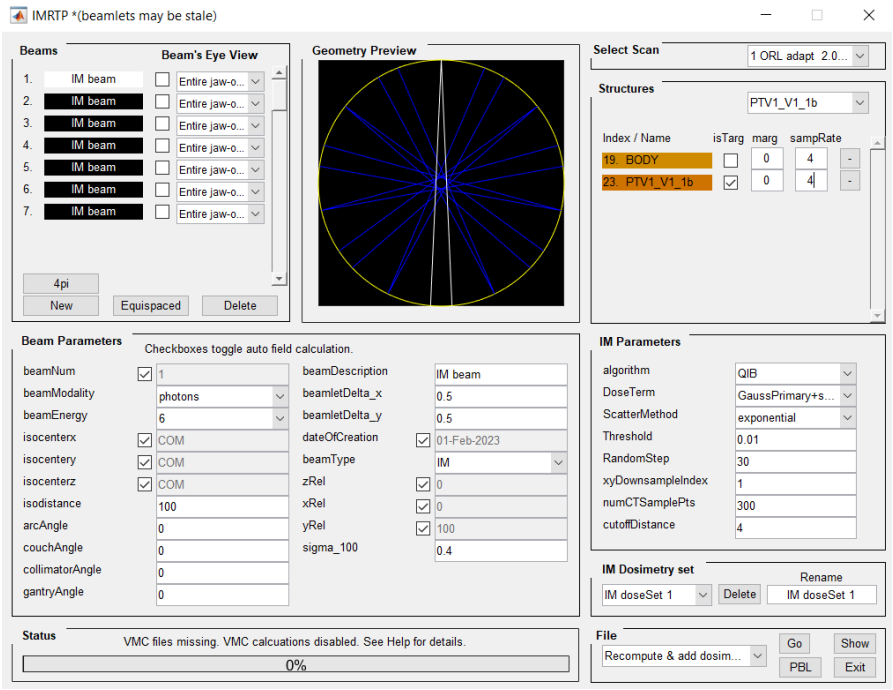


Figure 8: IMRTP Creation Menu within CERR

Then, target and body structures have to be selected in the dropdown menu on the top right corner. For the target structure, the box isTarg must be checked as well. This step is important as the isocentre position is calculated as the centre of mass of the selected target structure. The sampRate, visible to the right of the structure name, defines the downsampling applied to the dose grid. A sample rate of 4 corresponds to merging 4 voxels together in X and Y direction, e.g. producing a matrix of size 128×128 from an initial matrix of size 512×512. The downsampling is not applied to the z-direction of the matrix.

The calculation is then started by pressing the Go button on the bottom right corner.

Generation of dij/dif-files Once the calculation has finished, the MATLAB command prompt has to be opened. To store the dij and dif files, the following command is used:

Command 2.1: Create Dij/Dif files

write_konrad_dij_from_cerr(my_dij_name, structureNumber)

- my_dij_name is the name that you want to give to your files.
- structureNumber is the name associated to the structure in CERR for which the voxel dose values should be stored (often BODY).

The resulting *dij* files contain the dose contribution of each beamlet to the voxels in the selected VOI, whereas the *dif* files store the information about the patient geometry.

Generation of falloff-files Often, it is desirable to also penalize the dose falloff. Penalizing the dose falloff based on the distance to the target volume helps to reduce dose within healthy tissue which does not contain organs at risk or structures with objective functions. This is called a normal tissue objective (NTO), where the maximum dose value depends on the distance to the target structure. Thus, to use such an objective function, one needs a file storing these distance values for each voxel outside of the target structure. Using the following command, this file can be generated.

Command 2.2: Dose Distance

generate_distance(structNum, dif_file, sampleRate, output_file)

- structNum can be seen by looking it up under the Structures tab in CERR.
- dif_file is the name of the .dif file produced with *write_konrad_dij_from_cerr* using command 2.1.
- sampleRate has to correspond to the sampling rate chosen in the IMRTP menu.
- output_file defines the name for the file to be created.

Generation of VOI-files In a last step, information about the VOIs has to be stored

VOI is an acronym for "volume of interest" and can be stored as a matrix with the same size as the CT matrix. Such a matrix exists for every structure, where voxels belonging to the structure are assigned a value of one, and voxels outside of the structure are assigned a value of zero. This information is used during the optimization process in opt4D to determine which objectives are to be applied to which voxels, as different objectives can be defined for different structures. When downsampled resolutions are used, the VOI matrix also needs to be downsampled. For the adjustment of the VOI matrix shape, CERR needs information about the shape of the matrix to be produced. By using the following command, one can upload the previously generated distance file, which corresponds to the correct downsampled matrix size:

Command 2.3: Dose Upload

ju_load_downsampled_dose(outFile, dif, sampleRate, dose_name)

- outFile refers to the name you have chosen in command 2.2.
- dif is the name of the .dif file produced with *write_konrad_dij_from_cerr* using command 2.1.
- sampleRate has to correspond to the sampling rate chosen in the IMRTP menu.
- dose_name can be arbitrarily chosen.

Then, the VOIs can be downsampled and stored using

Command 2.4: Generate VOI files

make_vv_and_voi_from_cerr(1, filename, [firstStrNum:lastStrNum])

- The 1 denotes the dose number (uploaded with *ju_load_downsampled_dose*, command 2.3)
- filename refers to the name of the produced files (often, 'structures' is chosen)
- firstStrNum denotes the structure number of the first structure (usually set to 1)
- lastStrNum can be found by visiting the Structures tab in CERR and counting the total amount of structures present

2.2. Direct aperture optimization

This part of the workflow aims at optimizing a dose distribution and the corresponding set of MLC apertures and MU weights based on a set of defined objective functions for different structures of the patient, and generating a file containing the aperture information for the upload to Eclipse.

To connect section 2.2 of the workflow to sections 2.1 and 2.3, output and input modalities had to be modified to accommodate for the needed files of each step.

Starting with the input modification, the *dif* files created in section 2.1 were adjusted to include the isocentre positions for the plans calculated in CERR. Thus, the input section of opt4D had to be modified to store this information for the data export. To achieve this, the constructor of the *Aperture* class in opt4D was modified to store the parsed isocentre position read from the *.dif* files.

The implementation of the aperture information file started with the consideration of many technical aspects. Chosen output modality had to be easily writeable and accessible. Furthermore, the implemented functions should be written such that they could be easily modified to accommodate changes to the capabilities of opt4D, e.g., for beam angle selection.

Ultimately, the *.csv* file-type was chosen. *.csv* files can be written in an editor file, and thus can be produced by writing correctly formatted strings to an editor. This format follows the convention that rows should be separated by a semicolon (;).

To ensure a correct data upload into Eclipse, it was further important to find and export all variables needed for treatment creation in Eclipse and add this information to the generated *.csv* file. Table 1 references the chosen variables and describes what the corresponding values mean

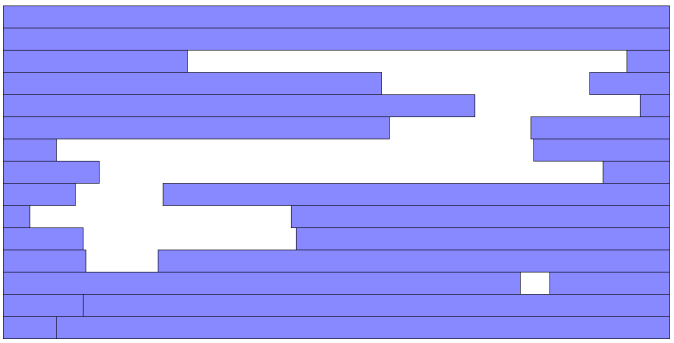
Variable	Description
Aperture Nr.	Indexes the aperture number
Field Nr.	Denotes the field the aperture belongs to
leaf Positions	x and y positions of the MLC pairs
Weight	The non-normalized weight of the aperture
Gantry Angle	The gantry angle of the field
Collimator Angle	The collimator angle of the field
Table Angle	The table angle of the field
Isocenter	x,y,z positions of the isocenter
Instance Nr.	Denotes the instance number the aperture belongs to

Table 1: Information stored in the csv file generated by opt4D

Thus, each row contains information about the positions of the left and right MLC leaf for each leaf pair, alongside the corresponding aperture weight, and the instance, field and aperture it is assigned to. For each combination of instance number, field number and aperture number, there exists one row storing further information, i.e. the gantry, collimator and table angle and the isocentre position. An example for this is shown in Figure 9.

	Aperture	Field	LeftLeaf_x	RightLeaf_x	Leaf_y	Weight	GantryAngle	CollimatorAn	TableAngle	Iso_x	Iso_y	Iso_z	Instance_No
2	4	0	-25.496416	57.27396	-27.5	0.115553	0	0	0	0.1684	-18.719601	-5.4801	0
3	4	0	11.440916	50	-22.5								
4	4	0	28.528728	59.849731	-17.5								
5	4	0	12.506719	39.213562	-12.5								
6	4	0	-49.93996	39.662895	-7.5								
7	4	0	-41.63208	52.652721	-2.5								
8	4	0	-46.265724	-30	2.5								
9	4	0	-54.935673	-5.816827	7.5								
10	4	0	-45	-5	12.5								
11	4	0	-44.385178	-30.754919	17.5								
12	4	0	37.055454	42.8372	22.5								
13	4	0	-45	-45	27.5								
14	4	0	-50	-50	32.5								

(a)



(b)

Figure 9: a) displays how an aperture is stored in the *.csv* file. b) shows a visualization of the corresponding leaf positions of the aperture.

2.2.1. Workflow description

Once all the files needed for the optimization have been prepared, they can be grouped into separate folders similar to a structure like the following:

- Dij
- Dif
- Output
- Plan
- vvfiles

The *dij* and *dif* files should be placed into the Dij/Dif folder and the *structure* files should be placed in the vvfiles folder. This includes the files *structures.dif*, *structures.floatvoi*, *structures.voi* and *structures.vv*.

Next, the contents of the Plan folder are generated. This folder needs two files, the *plan.pln* file and the *runopt4D* file. The *plan.pln* file is discussed first.

Setting up the plan.pln file This file contains information about the planning objectives to be used for the optimization process, the number of fields, the dose delivery model, etc. Based on the VOI numbers contained in the *structures.vv* file, objectives on different structures can be defined. *plan.pln* can be initialised as an empty *.txt* file. Then, the following statements should be added at the beginning.

```
title ...
    # Denotes the title of the plan
plan_file_root Dij/yourDijFilename
    # .dij files used for the optimization, .dij must not be added
vv_file_name vvfiles/structures.vv
    # .vv file used for the optimization
dif_file_name Dif/yourDifFilename.dif
    # .dif file used for the optimization

nBeams ...
    # Total amount of beams to be used, corresponds to amount of dij
  files
nInstances ...
    # Total amount of instances to be used for STF.
    # If plan is uniformly fractionated, nInstances = 1

parameter_type ...
    # squarerootbixel or dao depending on the optimization method
read_dijs_only_for_first_instance yes/no
    # Reads only the apertures for the first instance if set to "yes"
Dose_Delivery_Model standard/bed
    # Optimization based on physical dose models or BED
BED_nFractions ...
    # Amount of fractions used for BED calculation
BED_nFractions_in_instance
    # = BED_nFractions if uniformly fractionated
    # Otherwise apply the rules from the description below
```

For *BED_nFractions_in_instance* in the STF case, the amount of fractions per instance needs to be defined. This is done by typing the amount of fractions for each instance, separated by a white space. As an example, if one wants to use 8 fractions, both

1 1 1 1 1 1 1 1

and

2 2 2 2

are eligible, with the upper notation meaning that each fraction will be optimized on its own, whereas the bottom notation means that the dose distribution to be delivered in each 2 fractions will be identical.

To define objectives and constraints on the different VOIs, one needs to know their VOI number. These numbers are found within the structures.vv file, where they are listed in the form

0000x StructureName

However, it is important to note that there is a numbering difference between the .vv file and opt4D. VOIs in the plan.pln file should be addressed using the number found in structures.vv + 1. As an example, if one wants to select VOI 00001, then the corresponding number in plan.pln would be 2.

These VOI numbers can also be added to the *plan.pln* file as help, but one must put a # in front of them, such that opt4D understands that this is a comment. Not doing this results in a crash.

Many different objective functions are implemented in opt4D. The most commonly used ones are quadratic penality functions and functions penalizing mean dose values. Quadratic penalty functions can be implemented by specifying d^{min} and d^{max} for a specific VOI.

```
OBJ  objNum  type meansquarederror
OBJ  objNum  VOI voiNumber
OBJ  objNum  min_dose ...
OBJ  objNum  max_dose ...
OBJ  objNum  weight_under ...
OBJ  objNum  weight_over ...
```

Functions penalizing mean dose values instead are implemented as follows

```
OBJ  objNum  type minimizemean
OBJ  objNum  VOI voiNumber
OBJ  objNum  weight ...
```

These objectives are based on physical dose or other dose models. If the plan has to be optimized based on the BED, then another set of commands has to be added to the *Plan.pln* file to specify that the objectives must be evaluated for the BED

```
METAOBJ metaobjNum type bed
METAOBJ metaobjNum objectives objNum
METAOBJ metaobjNum alphabeta ...
```

where the α/β value of the corresponding structure should be specified.

If the plan is based on the objectives of an existing plan in Eclipse, the weights can be converted using the following formula

$$w_{\text{opt4D, structure}} = \left(\frac{w_{\text{Eclipse, structure}}}{\min(w_{\text{Eclipse}})} \right)^5 \tag{2.2.1}$$

where w_{Eclipse} denotes the set of all weights of the plan in Eclipse.

Setting up the runopt4d file The second file to be generated in the Plan folder stores information about the optimization algorithms and parameters to be used. This file is called *runopt4D*, which is a text file.

The first entry of such a file refers to the directory on the machine where opt4D is found:

youropt4Dlocation/opt4D

The second statement should correspond to the output folder created previously.

-o Output

In the following section, parameters with an underscore can be used for DAO, whilst the others are used for fluence map optimisation only.

Statements following the output-folder declaration give specifications about the optimization algorithm to be used. A few examples are listed here

- −−use_new_dao Initialises direct aperture optimisation (DAO)
- −−imrt_dao produces an IMRT plan by using DAO

If these statements are absent, fluence map optimisation will be carried out instead.

The parameters of the chosen algorithm are also declared in this file. Examples for this include the following:

- −−lbfgs_m x Amount of saved vectors for approximation
- −−linesearch_steps x linesearch is allowed to have at most x steps
- −−max_apertures maximum amount of apertures to be produced
- −−max_steps maximum amount of iterations to be executed
- −−min_steps minimum amount of iterations to be executed
- −−max_removable_apertures maximum amount of apertures to be removed

It is further needed to specify how the fluence map should be initialised. Often, one of the two following options is chosen:

- −−zero_initial_fluence fluence map is initialised as zeros
- −−unit_initial_fluence fluence map is initialised as ones

Alongside this initialisation, the noise added on top of the initialised fluence map has to be chosen by setting the following values:

- −−initial_fluence_noise noise added to the initial fluence map
- −−random_seed seed for the random number generation

The last line of opt4D references the location of the plan.pln file

Plan/plan.pln

On a side-note, for windows users on an Ubuntu WSL, the prefix /mnt/c can be used to access the windows folders within the WSL.

To generate a csv file used for the plan upload step later on, the following flag must be added to *runopt4D*:

−−generate_CSV

This task can now be performed by implemented GUIs, so the plan.pln and runopt4d file do not have to be set up manually any longer.

2.3. Import and calculation of a plan in Eclipse

The plan import and calculation is implemented in the form of a C# script called opt4D2Eclipse utilising the Eclipse Scripting API (ESAPI). Besides the csv file generated during the optimisation step, it takes information about the fractionation scheme as an input, as well as information about the desired structure set and target structure in Eclipse (not to be confused with what was done in section 2.1). The input is implemented as part of a GUI.

This section will also concern itself with motivating the underlying software architecture MVVM and explain how data from different sources, Eclipse and opt4D, is processed and accessed within the script.

2.3.1. Plan information in C#

As aforementioned, *.csv* files are used to store plan data. To keep the algorithm tidy and reduce possible errors, pre-processing was integrated into the ESAPI script. This pre-processing step consists of creating a hierarchical grouping of different class instances to mirror ESAPIs intrinsic class structure for beam objects (as shown in Figure 10).

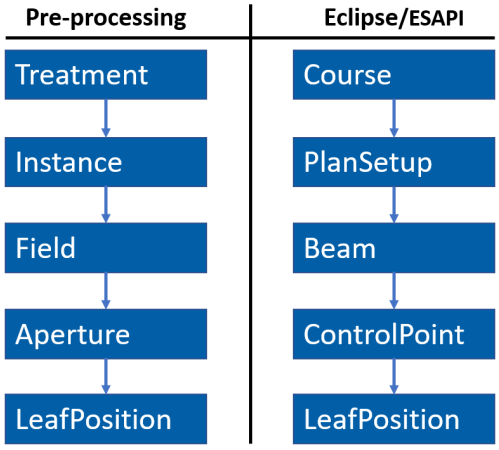


Figure 10: Comparison between class hierarchy between pre-processing in Opt4D2Eclipse and ESAPI

The *treatment* class is the parent node of the structure. It contains all other class objects and is only initialised once. It contains only one class attribute, which is a list of *Instance* objects. The total amount of *Instance* objects corresponds to the amount of different

instances (i.e. dose distributions) within the plan. Each instance object in turn contains a list of *Field* objects.

Field objects store the attributes of the beam to be used, namely the gantry, table and collimator angle, as well as the total field weight. Additionally, a list of *Aperture* objects is stored for each beam, which corresponds to all apertures generated in opt4D belonging to this specific beam number and instance number.

Aperture objects contain information about the aperture weight, and a list of *leafPosition* objects. *LeafPosition* objects store the positions of the left and right MLC leaves for a specific leaf row and the y value of the corresponding leaf row. Given this y value, the corresponding leaf number for a machine can then be calculated by using information about the width of the outer and inner MLC leaves. This is currently implemented based on information about Varian's TrueBeam 3232, equipped with a Millenium MLC 120 (Varian Medical Systems, Palo Alto, CA), containing a total of 60 MLC leaf pairs.

The leaves of the Millenium MLC 120 have a width of 10 mm for the 10 outermost MLC leaf pairs in each direction (leaf numbers 1-10, 51-60) and a width of 5 mm for the 40 inner MLC leaf pairs (leaf numbers 11-50). The list of *LeafPosition* objects stored within an *Aperture* corresponds to all necessary leaf positions to replicate the aperture given in opt4D in Eclipse.

Figure 11 displays the class hierarchy alongside the layers at which certain parameters are stored.

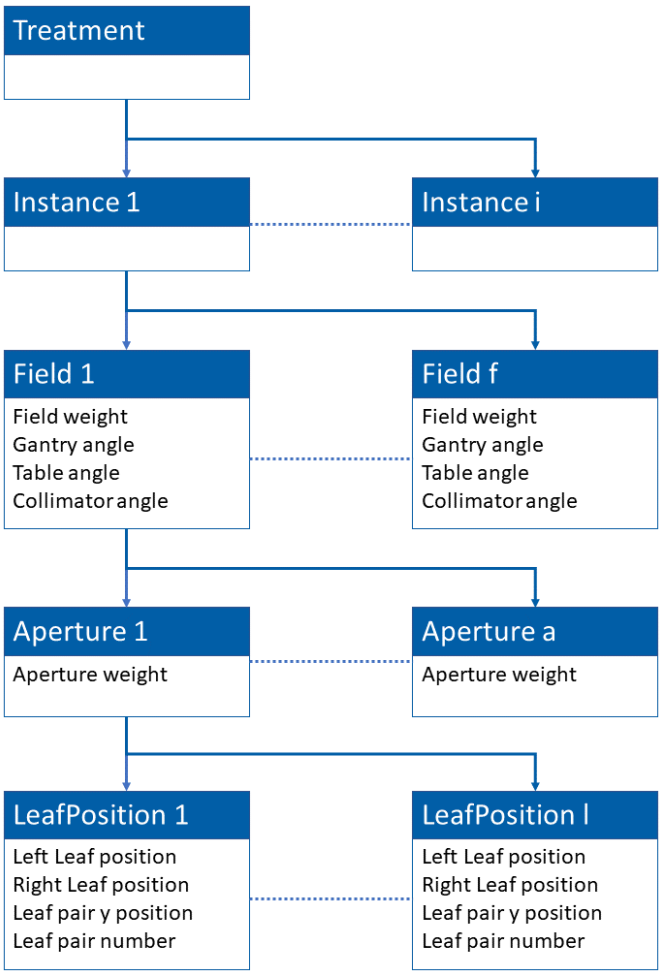


Figure 11: Diagram of the class hierarchy used in the data processing step.

2.3.2. MVVM architecture

The code for this section was written following the Model-View-ViewModel (MVVM) architecture, which is the accepted standard for Windows presentation foundation (WPF) applications, which is the application type of binary plug-ins for Eclipse. Binary plug-ins are one of the three plug-in types available for Eclipse, with the other ones being single-file plug-ins and stand-alone executables.

For this project, a binary-plug in was chosen as it allows for user interaction through a GUI, whereas single-file plug-ins have no interaction capabilities and stand-alone executables can only be interacted with via console commands, which reduces the simplicity of the interaction.

In this setting, different components interact with one another to connect the back-end to the front-end as shown in Figure 12:

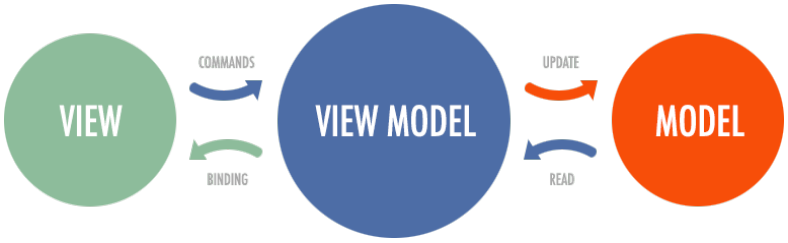


Figure 12: MVVM architecture [10]

- **Model**
  - The Model concerns itself with accessing and manipulating data
- **View**
  - The View contains the GUI, i.e., all components that a user directly interacts with
- **ViewModel**
  - ViewModel handles the interactions between a view and the corresponding model. Furthermore, it is also in charge of calling methods depending on the user input

In the most simple case encountered in this thesis, MVVM components will only interact with each other to fetch and represent data from Eclipse, whereas in other sections, retrieval and processing of user input is required. Processing takes place mainly in the model component, whilst the view gathers the user input and communicates the task at hand to the model via the ViewModel.

Such an architecture is quite convenient for tools under development. Adding new features amounts to minor editing of the GUI and data submission, whilst the data manipulation and processing step are implemented in the model component. This makes it simpler to keep the code tidy and improves readability.

2.3.3. Patient Model

The *patient model* corresponds to one of the simplest versions of MVVM. The data of the active patient is stored in a *patient model*, whose components are communicated through the corresponding *patient ViewModel* to the *patient view*, which is embedded in the main view and accessed via a tabular. The gathered information is then displayed on the GUI. This step was mainly introduced as a simple means to verify that the correct patient has been opened and to conform to the often used practice of including patient information on the GUI.

This section does not allow for any user interaction or data manipulation.

2.3.4. Calculation Model

The key component of the code is the *calculation model* alongside the *PlanView* and *Calculation Viewmodel* counterparts. This part of the program realizes the conversion from an opt4D plan to an Eclipse plan.

User-Interaction happens in the view, which is embedded in the main GUI and accessible through a tabular. The GUI allows the following fields as input:

- CSV-Path - Filepath of the csv table from opt4D
- Nr of Fractions - Planned number of fractions
- Dose per Fraction - Planned dose per fraction
- Structure Set - Dropdown menu of all available structure sets for the patient in Eclipse
- Target Structure - Dropdown menu of all structures in the structure set in Eclipse
- Energy Mode - Dropdown menu of available energy mode settings for commercial linacs
- Machine ID - Dropdown menu of all available linacs
- Treatment Type - Radio-button, either IMRT or ARC

These entries are necessary for every type of calculation. If a weight-optimized plan is imported, as discussed in section 2.5, the following additional fields need to be filled as well:

- MU - Filepath of the exported MU file
- Weight File - Filepath of the weight file from opt4D

The GUI is displayed in Figure 15.

In the following paragraphs, some possible pitfalls and intricacies of ESAPI are highlighted.

Leaf Position manipulation leaf positions in ESAPI are stored as a float array of size [2,x], where the x denotes the total amount of leaf pairs in the MLC. The first index addresses the left/right leaf (0 for left leaf, 1 for right leaf), whereas the second index yields the y position of the leaf (i.e. the leaf number). For example, if one wants to inspect the position of the 2nd left leaf, one would choose [0,1].

In ESAPI, leaf positions are set in two different ways depending on the situation. In case a beam with a single aperture is created, the leaf positions are handed to the function directly as an input as a 2D float array. In the other setting, where a beam with multiple apertures is generated, the workflow changes slightly.

One first creates a beam with the desired amount of apertures. This leads to ESAPI initialising a *beam* class object with a list of control points, where each control point contains its own list of leaf positions, which are all set to 0. Iterating through all control points allows to access the editable parameters of the beam, including the leaf positions. After the leaf positions were adjusted according to the obtained plan from opt4D, these changes must then be applied to the *beam* object to save the configuration.

In general, the leaf positioning algorithm corresponds to the following structure:

```
1  BeamParameters parameters = Beam.GetEditableBeamParameters()
2  foreach (ControlPoint cp in parameters.ControlPoints)
3  {
4    float[,] leafPositions = cp.LeafPositions;
5
6    Aperture aperture = instance._Fields.Where( x => x._FieldNr ==
     counter).First()._Apertures.First();
7
8    foreach (LeafPositions LeafPos in aperture)
9    {
10     leafPositions[0, LeafPos._LeafNr] = (float)LeafPos._LeftLeafX;
11     leafPositions[1, LeafPos._LeafNr] = (float)LeafPos._RightLeafX;
12   }
13 }
14 Beam.ApplyParameters(parameters);
```

Limitations and adjustments of leaf positions in opt4D A pitfall arose when implementing the leaf position algorithm in ESAPI, because opt4D generated leaf positions unfeasible for Eclipse to carry out due to physical limitations. This leads to a crash of the program during the modification of the leaf positions.

Further testing revealed that this crash was produced by combinations of MLC leaf and jaw positions. If a leaf tail enters a margin around the jaw position, it would contribute to the dose distribution via leakage. As this is not possible in Eclipse, an error is produced.

For the prostate patient, no problems occurred, due to the simple shape of the tumour. For the complex head-and-neck tumour shape on the other hand, a threshold had to be set for maximum leaf overtravel to mitigate the issue. This threshold was set to 42.9 mm, obtained by trial-and-error testing.

To solve this issue whilst allowing the maximum range for the MLC leaves, the relation between the jaw positions and the maximum leaf overtravel distance has to be found and implemented into the opt4D code as a dynamic boundary value. As covered in the next section, right and left jaw positions are determined by the leftmost and rightmost MLC position. Thus, this boundary depends on the leftmost and rightmost MLC position of an aperture.

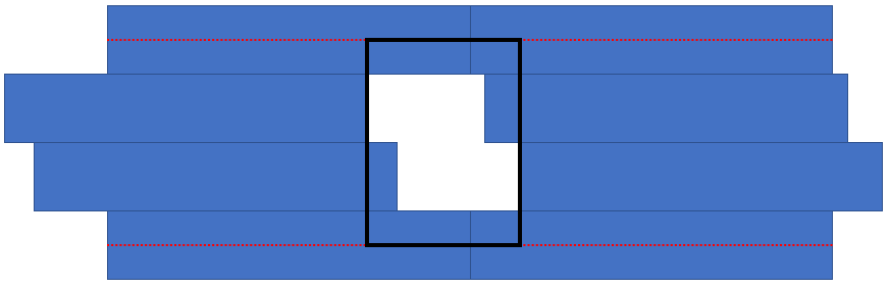
One could also apply such a dynamic boundary to the leaf position algorithm. However, one would need to either reduce the jaw opening, possibly closing parts of open MLC leaves, or would need to set a threshold on the maximum leaf overtravel, as done for the

head-and-neck tumour patient. This results in a change of the aperture shapes, which comes at the cost of not producing the calculated optimal aperture shapes, possibly worsening the result.

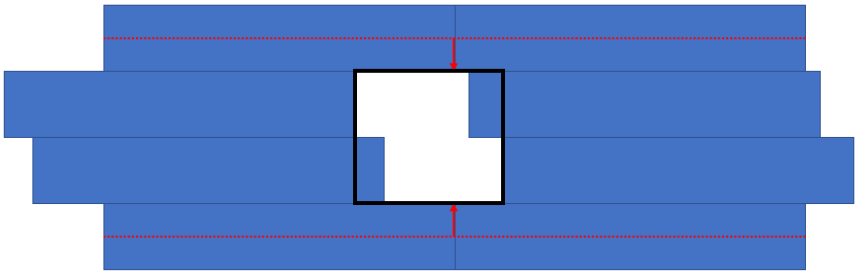
Another issue arose regarding the fully closed Leafs. As aforementioned, leaf positions are initialised at 0, meaning that closed Leafs still contribute to the dose calculation, e.g. through leakage. The following section provides more insight into this issue.

Tracking Jaw positions Besides the MLC positions, one also has to adjust the jaw positions, which are stored as ESAPIs *VRect* objects. A VRect object is a rectangle whose x1, x2, y1 and y2 position can be altered depending on the input given to the constructor.

When going through the MLC positions for each control point, the algorithm stores the maximum and minimum position of all open leaf pairs in each dimension. This means that the minimum value in x direction is defined based on the left-most left leaf position for an open leaf pair, and vice versa for the maximum x value. In y direction, the same concept is applied, however, as the y-values are defined with respect to their middle, half of the leaf height has to be added or subtracted from the value depending on whether it is a maximum or a minimum value, as illustrated in Figure 13.



(a) Jaw position without adjustment on the y value of the leaf positions



(b) Jaw position after correcting the y value of the leaf positions

Figure 13: Comparison between jaw positions (a) without and (b) with adjustment of the y coordinates

Isocenter-Positions in Eclipse In Eclipse, two different coordinate systems are used (for more information about this, consult Appendix A). When altering the isocentre positions through ESAPI, one works in the DICOM coordinate system with its corresponding DICOM origin. The GUI of Eclipse uses the standard coordinate system with the corresponding user origin. Figure 14 illustrates the differences in origin points.

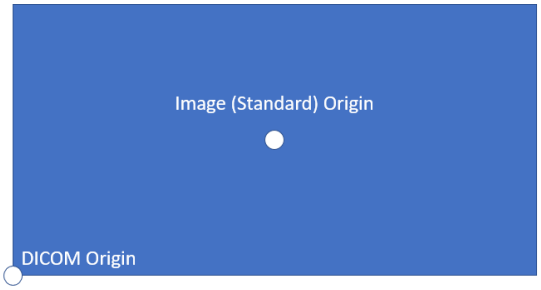


Figure 14: Comparison of standard and DICOM origins in a 2D toy example

When a plan with a fixed isocentre is imported into Eclipse, the plan will thus be set to this isocentre with respect to the DICOM origin point. This however does not correspond to the positions of the structures of the plan, as they are defined using the user origin point. The plan will instead appear translated by a constant in each axis direction. This issue can be mitigated by subtracting the user origin from the isocentre position in all axes. After applying this correction, the plan will be centred at the correct location within the standard coordinate system.

Upon pressing the Calculate button, the plan will be uploaded into Eclipse and calculated directly. This means that for each instance, a corresponding plan will be generated, filled with the corresponding fields. The plan can be inspected and saved as in Eclipse.

2.3.5. Export model

The data stored in the *calculation model* is handed over to the *export model* during the export step. The interaction with the *export view* consists of choosing a location for the file to be saved at and choosing a name root. More information about the inner workings of the export implementation can be found in section 2.4.

It might be worth mentioning here why the decision was made to execute both the calculation and export in the same script, instead of creating a script with the sole purpose of exporting files. This decision was made due to the necessity to save all changes made to a patient prior to opening another script in Eclipse. Thus, all beams calculated would have to be saved to the patient. Whilst this might not be an issue on the virtual machine, set in a research environment, this could produce serious clutter in a clinical system. By executing the export process after the calculation step in the same script, the created beams can be directly accessed without the need to save any of

them, only adding them to the patient temporarily. Thus, once the export has finished, all calculated beams can be deleted without any need to save the changes made to the patient, keeping the patient tidier.

2.3.6. Workflow description

To upload and calculate a plan in Eclipse, the Eclipse script opt4D2Eclipse can be used. To execute this script, the patient should be opened in Eclipse. Then one can navigate to "Tools" on the toolbar. On the dropdown menu that is opened, "Scripts..." should be chosen. There, the script can be started by clicking the "Run" button. This will open up the GUI.

By pressing the "Calculation" tab, the GUI switches to the section taking care of uploads and calculations. Figure 15 shows how the GUI will look.

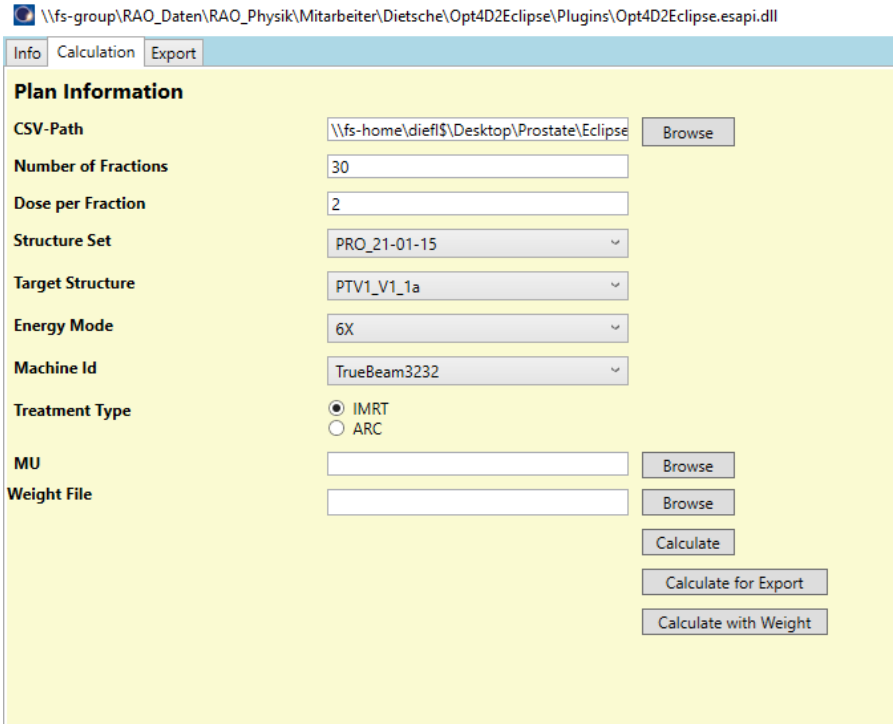


Figure 15: GUI of opt4D2Eclipse for calculations

The "Calculate" button can be used to upload and calculate a plan without export. This was mainly implemented to inspect the resulting plans qualitatively, e.g., to view the leaf positions and the treatment structure.

If the apertures should be exported as covered in section 2.4, then one should select the option "Calculate for Export". **Important:** If you want to export the apertures, do not

close the script after the calculation has finished.

The fields "MU" and "Weight file" can be left empty for the time being. They will become important in section 2.6.

2.4. Aperture Export in Eclipse

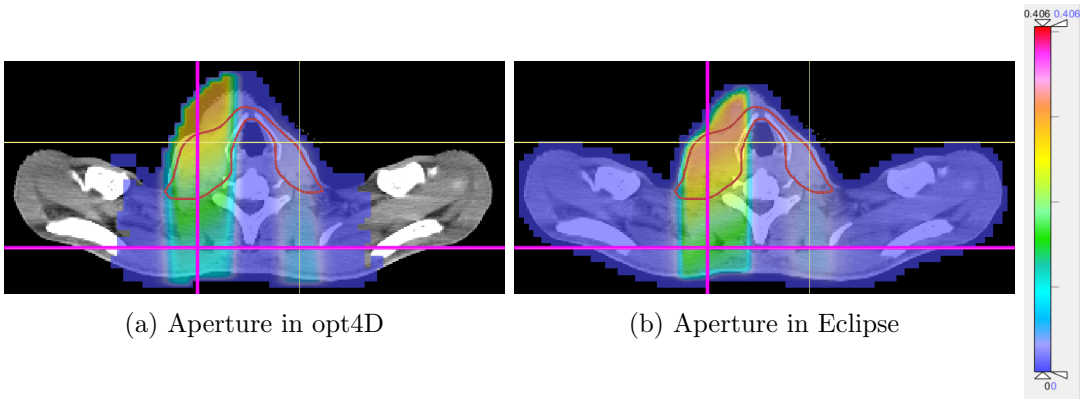
Figure 16 shows the difference in the dose distribution for apertures calculated in Eclipse and opt4D, respectively. opt4D apertures are calculated using dose-influence matrices from CERR (section 2.2), whereas the dose distribution in Eclipse is calculated using the AAA algorithm. Not only might there be differences in the dose calculation algorithm, but Eclipse also takes other effects into consideration, e.g. transmission, leakage and the tongue-and-groove effect, which also influence the resulting dose distribution.

A noticeable difference between the dose distributions in Figure 16 a) and b) is that the dose calculation in Eclipse is conformal to the CT regarding the depth dose curve, whereas opt4D starts calculating the dose in a margin around the CT in beam direction. Both dose curves arrive at the same endpoint, but opt4D shows a lack of build-up region within the body. Looking at panel c), one can see that despite this, the depth dose falloff is quite similar. The curves for the lateral dose profile, panel d), are also similar, with the lateral dose profile of the plan in opt4D being a bit narrower within the region of around -10 to 0 cm. Furthermore, the lateral dose profile for opt4D underestimates the dose in a region around 0 cm, whilst overestimating the dose in the region around 2-6 cm. Furthermore, the dose tails of the outer beam are underestimated in opt4D.

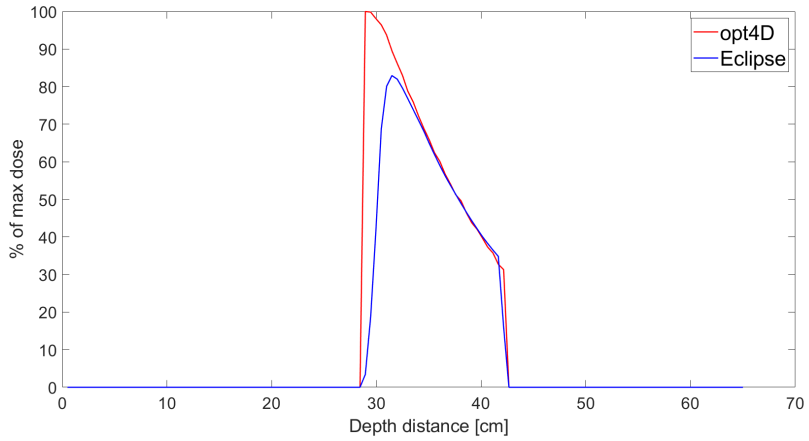
opt4D finds at least a locally optimal dose distribution for the given beam angles. However, when the plan is re-calculated in Eclipse and the resulting dose distribution changes, the aperture weights might not correspond to a (local) optimum any longer. Thus, these weights should be modified such that they result in at least a local optimum for the given aperture doses.

By re-optimizing the aperture weights without changing the aperture shapes, one can in part compensate these differences in the optimization process. This weight optimization however needs information about the dose distribution of each aperture as an input. Thus, these dose distributions need to be exported from Eclipse for further processing.

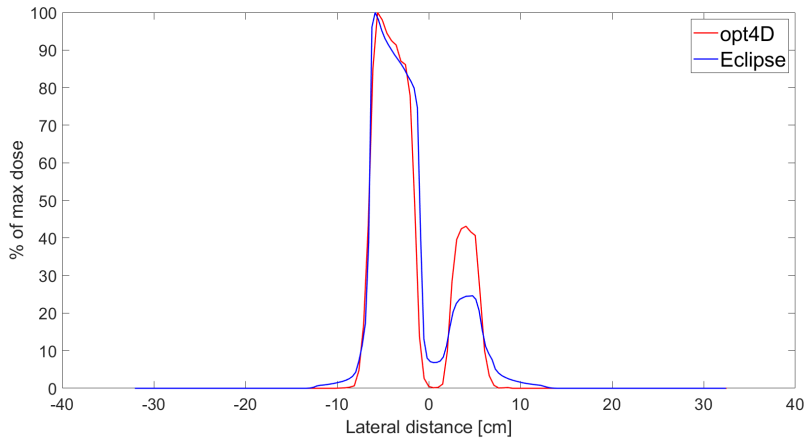
The aperture export denotes the next step in the workflow. Using the concepts introduced in the previous sections, a user can upload opt4D plans into Eclipse and export the gathered information for the optimisation steps to follow.



(a) Aperture in opt4D (b) Aperture in Eclipse



(c) Depth dose curve for a), b)



(d) Lateral dose profile for a), b)

Figure 16: Comparison between the same aperture calculated in opt4D and Eclipse. The same colormap was used for displaying both apertures in CERR. The lateral dose curves were normalized to correspond to the same maximum value, whilst the depth dose curve was normalized such that the depth dose falloff in the body matches. The magenta lines in a) and b) denote the slices taken for the graphs in c) and d).

Calculation The dose calculation in this step disregards the structure of treatment, instance, field, aperture and leaf positions. Instead, only apertures and their leaf positions are important for the algorithm. However, the apertures are still ordered by instance number, field number and aperture number (in that order). This approach was chosen to work around a limitation of ESAPI.

When a beam with multiple segments is created in ESAPI, it has to be given meterset weights in the initialisation function corresponding to the aperture weight. The function then uses this information to generate the correct amount of control points. More about this follows in section 2.6.

Once a beam object has been initialised, the meterset weight is set to a *get* variable, meaning that only access, but not manipulation of the value is allowed. Thus, one is not able to set other meterset weights to zero to calculate the dose matrix for a singular aperture. This issue was solved by applying the following work-around:

Each aperture can be added as a beam with a single aperture. This means that it is possible to access the beam weight, and set all of them except for one beam to 0. Thus, it is now possible to access contributions for each aperture. Furthermore, the exact MU values of each beam can be stored in a *.txt* file, which is important for the upload of the weight re-optimized plans in section 2.5.

ESAPI does not allow for dose calculations with preset MU values for beams with single apertures. Thus, when the weight re-optimization is done in section 2.5, the resulting weights are with respect to the MU values used in the export step. Due to this, MU values have to be stored.

The flattened structure is displayed in Figure 17.

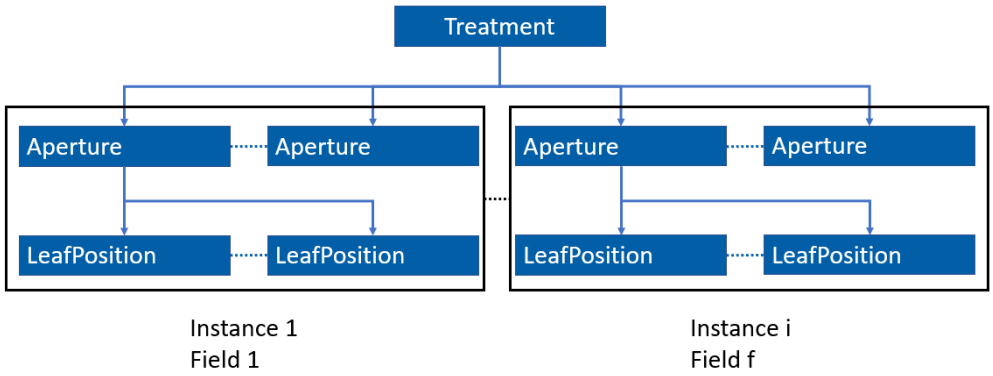


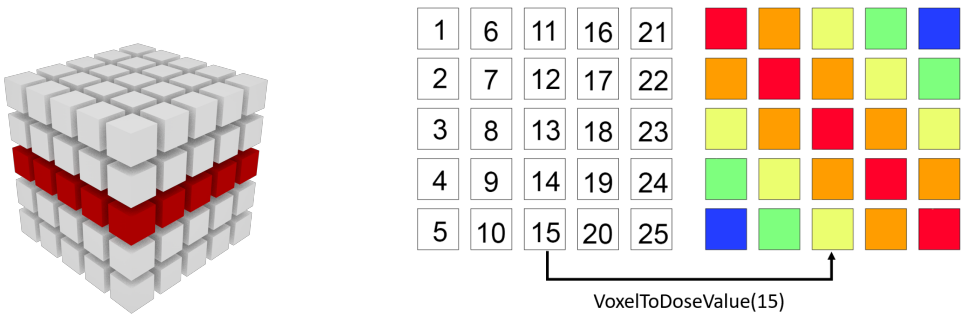
Figure 17: Flattened plan structure in the export algorithm. i corresponds to the maximum instance number, f corresponds to the maximum field number in an instance.

Export Once the dose calculation has finished, one can loop through the entire set of beams using a nested loop, such that the index of the first loop denotes the beam to be exported, whereas the inner loop concerns itself with setting the field weights of all other beams to zero. The beam weight of the beam at the outer index is set to 1. As the dose contribution of all beams except one is now zero, accessing the total dose matrix yields the values for one specific aperture. The code for the used algorithm is listed here:

```
for (int ApertureNr = 0; ApertureNr < AmountOfBeams; ApertureNr++)
{
  for (int BeamIt = 0; BeamIt < AmountOfBeams; BeamIt++)
  {
    var par = plan.Beams.ElementAt(BeamIt).GetEditableParameters();
    if (BeamIt == ApertureNr) { par.WeightFactor = 1.0; }
    else { par.WeightFactor = 0.0; }
    plan.Beams.ElementAt(BeamIt).ApplyParameters(par);
  }
  writeDoseToText(...)    #function to export the dose matrix
  writeInformationToText(...) #function to export geometrical
                              #information of the dose matrix
}
```

This value can then be written to a *txt* file for each beam in the treatment, yielding a total amount of files equal to the number of apertures in all fields. Alongside each dose matrix file, an additional info file is written, which contains geometrical information concerning the dose and CT grid used in Eclipse.

ESAPI does not allow direct access to the dose matrix. Instead, one can loop through each slice in z direction of the dose object. Using the z-slice as an input, the function GetVoxels then returns all voxel indices present in the specific plane. Looping through all voxel indices, one then can use the VoxelToDoseValue command to return the dose value at each voxel, which yields the entire dose matrix if done for each slice. The process is visualised in Figure 18.



(a) (b)

Figure 18: Process of accessing the dose matrix: a) selecting the desired index plane in z-direction and b) converting the selected entry of the index plane to a dose value

After all apertures have been exported, the beam weights are set back to one. This step is important as the MU values need to be stored for the upload of the weight optimized plan later in the workflow.

The naming convention for files produced in this step relies on a name root string to avoid confusing exported apertures for different patients. The name root is followed by information about instance number, field number and aperture number to keep the beams ordered by these quantities. Figure 19 illustrates the produced filename.

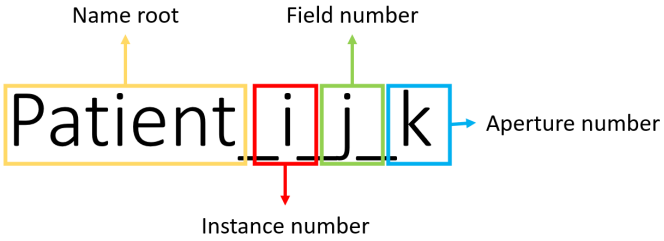


Figure 19: Example to illustrate the used naming convention for an example with instance number i, field number j and aperture number k

On a side-note, whilst instance, field and aperture numbers start from 0 in the *.csv* tables generated in section 2.1, the export starts the count at 1. This was implemented as the inputs for a weight re-optimization (section 2.5) also start at 1.

Dose matrices can also be exported as DICOM files. This process relies on setting up server connections between Eclipse and a DICOM file-service (DICOM Services Configuration), which can then be used to perform the data transformation. This however did not work on the virtual machine used for scripting during this thesis, and was thus replaced by using text-files for the time being.

The exported text-files containing the dose matrix information for an aperture were compared to the DICOM files containing the dose matrix for the same aperture, with them resulting in an identical dose matrix when parsed in MATLAB.

2.4.1. Workflow description

Continuing from section 2.3, once the Calculation has finished, the script should not be closed. Instead, one can navigate to the Export section within the GUI. Here, two textboxes are found. The first specifies the folder the files should be exported to. The second one defines the name root of the files. The interface is seen in Figure 20.

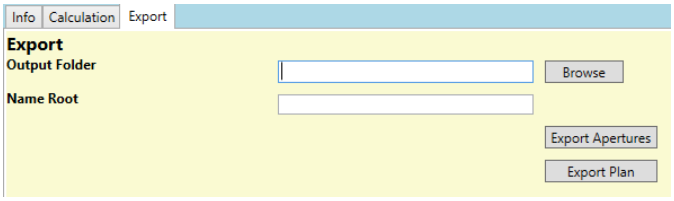


Figure 20: Export page of Opt4D2Eclipse

After the output information has been submitted, the export can be started by pressing the Export Apertures button. Export plan can be used to export the cumulative dose distribution of the entire plan.

This results in a folder containing the following files

- NameRoot_i_f_a_dose.txt for each aperture

  Contains the dose matrix

- NameRoot_i_f_a_info.txt for each aperture

  Contains information about shape and geometry of the dose matrix and CT

- NameRoot_i_MU.txt

  Contains the MU values used in the calculation

where i corresponds to the instance number of the beam, f corresponds to the field number, and a corresponds to the aperture number, as previously discussed.

2.5. Weight re-optimization

As seen in Figure 16, apertures calculated in opt4D and Eclipse are not identical. Thus the resulting cumulative dose distributions might present some differences. To reduce degradation of the dose distribution optimized in opt4D as much as possible, a weight optimization can be performed using the exported apertures calculated in Eclipse as an input. During such a weight optimization, the aperture shapes are not modified, as this would mean that the dose distribution would change and would have to be recalculated. Thus, only the aperture weights can be changed to improve the result.

To implement this step, both geometrical and technical aspects need to be taken into account, which are covered in this section.

2.5.1. Geometrical considerations

As the workflow combines files from both Eclipse and CERR, it is important to keep in mind where they differ and incorporate these changes into the workflow. The following

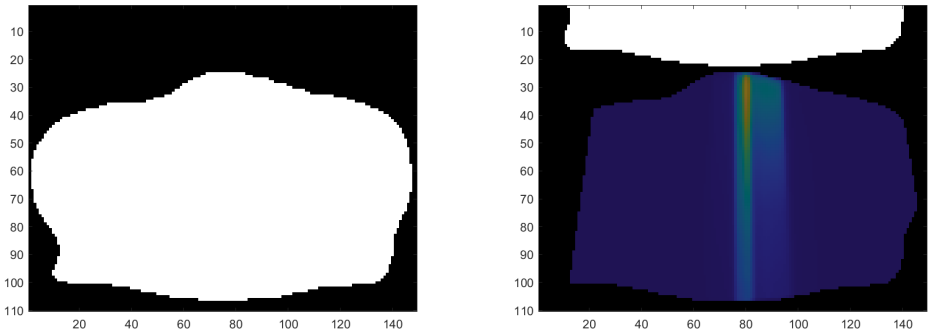
sections highlight the differences between CERR and Eclipse/opt4D and give information about how these differences were mitigated in the implementation of the workflow.

CERR/Eclipse Eclipse and CERR use different coordinate systems. Eclipse contains two different coordinate systems, the standard coordinate system and the DICOM coordinate system [6]. For this section, only the DICOM coordinate system, used for dose calculations, will be of interest. CERR, on the other hand, utilizes a unique coordinate system. The differences in axis orientation can be found in Table 2.

	DICOM	**CERR**
x	Shoulder-to-Shoulder	Shoulder-to-Shoulder
y	Front-to-Back	Back-to-Front
z	Feet-to-Head	Head-to-Feet

Table 2: Summary DICOM/CERR coordinate Systems

To create the vvfiles for the weight optimization, the CT image needs to be interpolated to match the size of the dose matrix, and the coordinates of the dose matrix have to be translated to correct for some intrinsic misalignment arising from the different coordinate systems. Figure 21 shows the case where no translation was applied after the interpolation, whereas Figure 22 shows the correct result after performing the interpolation.



(a) Interpolated BODY structure (b) Matching Matrices without translation

Figure 21: For illustrational purposes, the translation was applied only on the z-axis, such that the shift of the dose grid and ct grid on the same slice can be observed.

Whilst the interpolation can be done directly in MATLAB, to perform the translation accordingly, the property *Plan.Dose.Origin* of the plan structure in ESAPI needs to be

exported as well. The information needed is stored within the plan.Dose field. Once a plan has been calculated, its origin can be accessed by using the following command:

Command 2.5: Access the DICOM Origin of a Dose matrix

Plan.Dose.Origin
The origin property of the Dose class yields a VVector which contains the following attributes

- Item = Indexing of the x, y, z components
- Length = Length of the Vector
- x = X coordinate of the origin
- y = Y coordinate of the origin
- z = Z coordinate of the origin

Let **X, Y, Z** be vectors of length dim(x), dim(y), dim(z) respectively, such that

$$\begin{aligned} \mathbf{X} &= [0, 1, ..., \text{dim(x)}]^T \\ \mathbf{Y} &= [\text{dim(y)}, \text{dim(y)-1,...,0}]^T \\ \mathbf{Z} &= [0, 1, ..., \text{dim(z)}]^T \end{aligned} \tag{2.5.1}$$

Then, to get a vector corresponding to the dose coordinates, one can use

$$\begin{aligned} \tilde{X} &= plan.Dose.XRes \cdot \mathbf{X} \\ \tilde{Y} &= plan.Dose.YRes \cdot \mathbf{Y} \\ \tilde{Z} &= plan.Dose.ZRes \cdot \mathbf{Z} \end{aligned} \tag{2.5.2}$$

Then, these vectors can be transformed to the CT coordinate system via a translation

$$\begin{aligned} \bar{X} &= \tilde{X} + plan.Dose.Origin.x \cdot \mathbf{1}_{\text{dim(x)}} \\ \bar{Y} &= \tilde{Y} - plan.Dose.Origin.y \cdot \mathbf{1}_{\text{dim(y)}} \\ \bar{Z} &= \tilde{Z} - plan.Dose.Origin.z \cdot \mathbf{1}_{\text{dim(z)}} \end{aligned} \tag{2.5.3}$$

where $\mathbf{1}_{\text{dim(.)}} = [1,1,...,1]^T \in \mathbb{R}^{\text{dim(.)}}$

Thus, the interpolation can now be carried out in MATLAB as the coordinate systems have been matched.

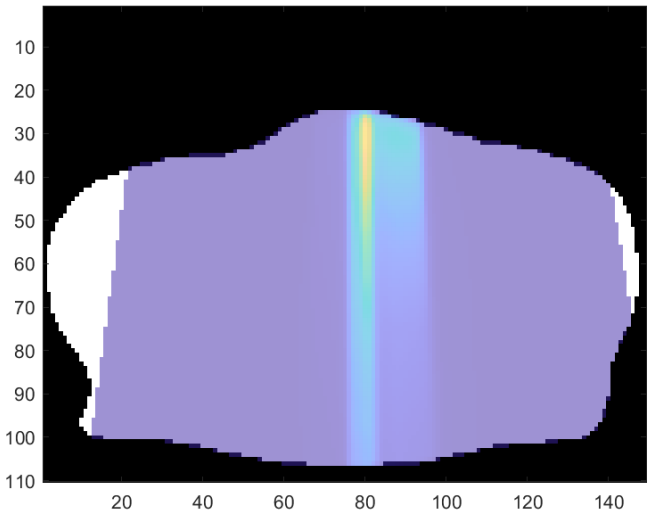


Figure 22: Match between Dose and CT matrix after interpolation and the corresponding translation in x, y, z dimension

While the z-axis is mirrored between the CERR and DICOM system, the vector is still saved in the DICOM direction in CERR, meaning that only the y coordinate needs to be flipped for the coordinate system matching step, as seen in equation 2.5.1.

CERR/opt4D Once the files are prepared for the import to opt4D, an additional modification must be performed. In CERR, matrices are stored in a format such as [y,x,z], meaning that the first index accesses the y-position of a voxel, whereas the second index stores the x-coordinates. However, in opt4D, the matrix structure is [x,z,y].

This misalignment can be corrected by utilizing a built-in function in MATLAB, namely the *ipermute* function

Command 2.6: ipermute

ipermute(matrix, dimorder)

- matrix is the matrix to be permuted
- dimorder yields the order of the dimensions used for the permutation

ipermute and permute correspond to two Matlab functions with the following feature

$$\textbf{permute}(\textbf{ipermute}(\text{matrix}, [i,j,k]), [i,j,k]) = \text{matrix} \tag{2.5.4}$$

where i,j,k correspond to any of the matrix axis.

Assuming that the starting point is a matrix in CERR format [y,x,z], ipermute corresponds to the following permutation matrix

$$\textbf{ipermute}([3,1,2]) = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \tag{2.5.5}$$

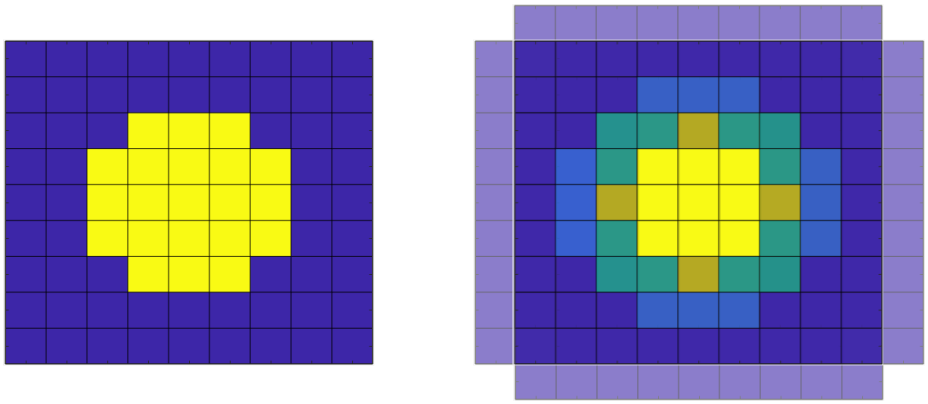
Applying this to the initial matrix thus yields a dose matrix which corresponds to the desired opt4D format.

2.5.2. Technical aspects of the implementation

The .dij and .dif files are generated from the exported dose matrix and dose info files from Eclipse. The entire process consists of parsing the exported text-files, and transforming them into a dij-struct in MATLAB, which can then be encoded into .dij and .dif files using existing implementations in MATLAB.

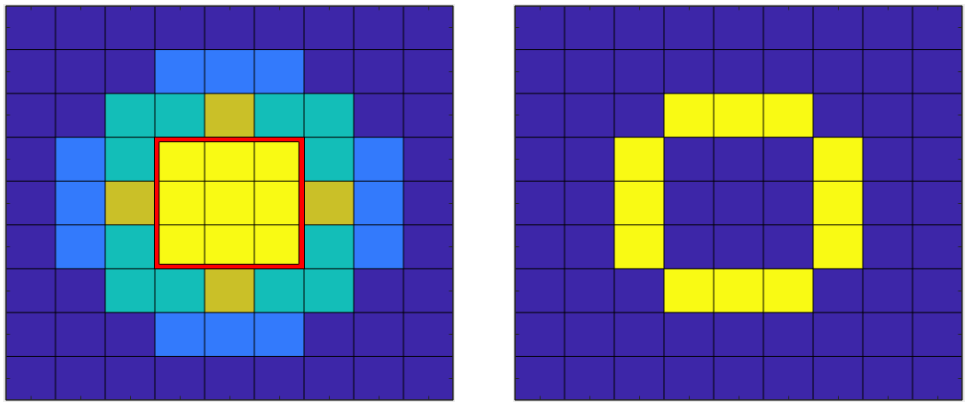
To incorporate the previously mentioned geometrical considerations, one can use existing interpolation methods on the VOI matrix such that it conforms to the dose matrix shape. Then, the translation can be applied. Important to note here is that one could interpolate the exported doses instead of the VOI matrices. However, the amount of apertures in a plan can become quite large for spatio-temporally fractionated plans, which means that interpolating the VOI matrix, which is not affected by the total amount of apertures, is beneficial for the overall runtime.

To create an NTO objective file, the target structure needs to be interpolated to the shape of the dose matrix. Then, one needs to find all edges of the interpolated target structure. This step can be vectorised for runtime improvements. VOI matrices contain 0 values for each voxel outside of the VOI, and a 1 otherwise. This means one can sum up dose matrices where each one is shifted by a value of one of the axis dimensions. This results in a total of 7 matrices in the 3D case (1 matrix without a shift, 2 matrices for a shift in each direction). In case no border is present in the proximity of a voxel, then the total value of the resulting matrix at this point will be equal to 7. If this is not the case, the voxel is next to a border. The algorithm is visualised in Figure 23.



(a) Dummy VOI for distance calculation algorithm example

(b) Sum of VOIs with an index shift of one added on top of a)



(c) The red square denotes the interior of the structure

(d) The modified VOI matrix only contains the pixels at a border.

Figure 23: Index shift procedure visualisation for the 2D case. a) shows the initial VOI matrix, b) shows the resulting matrix by adding matrices with an index shift of one on top. In panel c), pixels inside the structure are highlighted, which are not needed for the calculation. Panel d) shows the resulting VOI matrix, which uses the information from panel c) to discard all values which are not at a border. This concept can be extended to the 3D case.

Each point of the VOI not containing a border is then set to a value of -1, which indicates to the algorithm that it should be ignored for the target distance calculation. This means that all points at a border contain a value of 1, whilst all points outside of the VOI are denoted with a 0. Thus, on can now calculate the distances to all bordering points and choose the minimum value as the distance from the target volume.

For each point, this distance is stored as a matrix. Should a voxel be within the target volume, this value is simply set to 0. At the end of the algorithm, the matrix is then stored as a .dat file.

2.5.3. Workflow description

To perform weight optimization using the exported apertures from Eclipse, the same file types as for the initial optimization are necessary

- dij
- dif
- structures.voi, structures.floatvoi, structures.vv, structures.dif
- plan.pln
- runopt4D

The dij, dif and vvfiles can be created using the exported files from Eclipse via MATLAB functions.

At the beginning, similar to the process for the first optimisation, the patient needs to be opened in CERR. Then, on the command line, the following command should be used:

Command 2.7: Create dij/dif from Eclipse

Read_Output_Folder(Folder, NameRoot, Output, save, load)

- Folder refers to the location of your dose.txt and info.txt files
- NameRoot is the naming convention that you have chosen previously in Eclipse
- Output is the location where the files should be generated
- Save is a boolean and gives you the option to save the dose matrices as .mat files (not recommended for large amounts of apertures)
- Load is also a boolean and gives you the opportunity to load the previously saved .mat files again, however, this should mainly be used for debugging. You have to be in the folder where the .mat files are saved in Matlab to access them.

The progress can be observed from the command line. This will result in the dij and dif files for the weight optimization. To get the vvfiles, one can then use

Command 2.8: Create vvfiles from Eclipse

make_vv_and_voi_from_eclipse(Folder, File, 'structures', [FirstStrNum:LastStrNum])

- Folder refers to the location of your dose.txt files
- File refers to the name root of a dose/info file pair. As an example, to select "Patient_1_1_1_dose / info .txt, one would use "Patient_1_1_1" as an input. It does not matter which file (i.e. which aperture) is selected.
- 'structures' is used to name the files created, however, you can chose another name. To keep consensus with the CERR instructions, it is recommended to keep the name structures.
- FirstStrNum is often set to 1, whereas LastStrNum is the last structure assigned in CERR.

Another file that is still missing is the distance file. This can be generated using the command

Command 2.9: Create distance files from Eclipse

generate_distance_Eclipse(strNum, dif_file, NameRoot, FolderInput, output)

- strNum is the VOI number of the target structure in CERR
- dif_file is the dif file containing information about the dose matrix shape
- NameRoot is the name root of a dose/info pair exported from Eclipse. It does not matter which one is chosen.
- FolderInput denotes the location of the dose/info files exported from Eclipse.
- output is the name of the output file

Once these files have been created, one can follow the same steps as in the instructions for the initial optimization, by running the plan optimization in opt4D with the same objectives as previously used in step 2.2 of the workflow. It is also possible to adjust the planning objectives in this step, if the user wants to adjust the dose distribution.

2.6. Import and calculation of a weight re-optimized plan to Eclipse

This version of the calculation algorithm incorporates the structure of the treatment class into the plan generation. In total, there are three different cases to be considered, which are discussed in the following section.

The concept of meterset weight and cumulative meterset weight is introduced here as it is a crucial aspect of the upload and calculation process. The meterset weight denotes the ratio of the monitor units (MU) delivered during an aperture to the total planned MU for a beam. Thus, the cumulative meterset weight denotes the ratio of the MU delivered up to a certain aperture to the total planned MU of the beam.

IMRT Field with one aperture In case only one aperture is present in the active field, one may use ESAPIs AddMLCBeam function. This function takes as input the machine parameters, a pre-defined float containing all leaf positions, a VRect object storing the Jaw positions, gantry, collimator and table angles, and the isocentre position.

As there are no meterset weights to be set, the field weight can be altered instead.

Step-and-Shoot IMRT If a field class contains multiple apertures, a different approach has to be chosen. In this case, one first has to define a double array containing the cumulative meterset weights. Important to note is that this double array's size needs to be twice the total amount of apertures in the field in case of step-and-shoot IMRT. The reason for this is that two consecutive control points are forced to have the same leaf positions for step-and-shoot delivery.

The cumulative meterset weights are assigned as follows. The initial meterset weight is 0. The value of the next point is the normalized weight extracted from opt4D. All points with an even index (2,4,6,...) are assigned to the value of the point prior to them, whilst all points with odd indexes (3,5,7,...) are defined as the sum of all normalized weights up to this point.

As an example, assume a plan has 3 apertures, each with a normalized weight of 1/3. Then, the metersetWeights array would be defined as follows:

$$[0, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{2}{3}, \tfrac{2}{3}, 1]$$

So the first value is set to 0, the second value is set to the sum of all normalized weights to this point $\frac{1}{3}$. The next value is assigned to the previous value $\frac{1}{3}$ as its index is even. For the following index (odd), the sum of all normalized weights up to this point is taken ($\frac{1}{3} + \frac{1}{3}$), and this procedure is repeated until the entire array is filled. Due to the normalization step, the last value is always 1. Figure 24 displays the meterset weights of the dummy example in a graph.

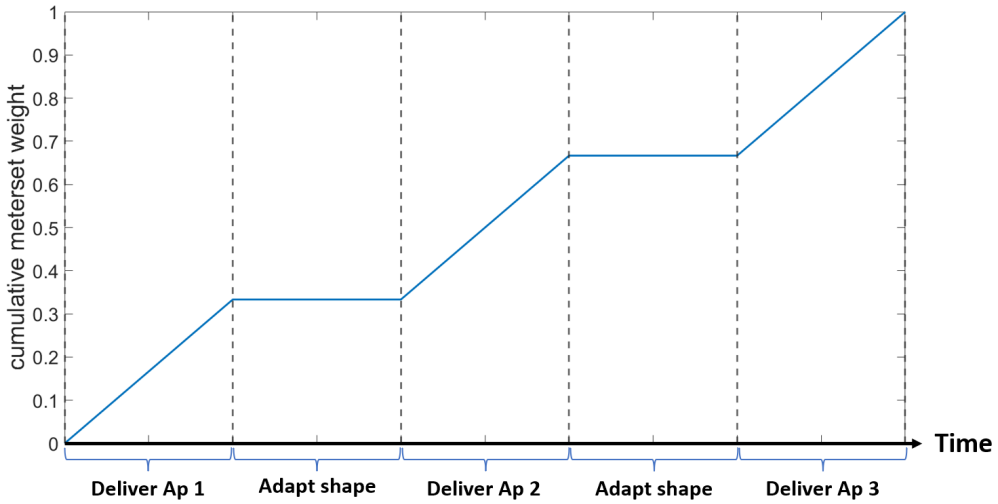


Figure 24: Illustration of the meterset weight structure for step-and-shoot IMRT based on the dummy example as a graph over time.

The reason for this is that Eclipse needs information about how many MU's are to be delivered between two control points. For the first control point, 0 MU are delivered, as the Leafs need to move to the defined leaf positions. Between the first and second control point, the shape is not altered, but a defined amount of MU are delivered. When going to the third control point, no MU's are delivered again, whereas between control point 3 and 4, which have the same leaf positions, MU are delivered. This also explains why the cumulative meterset weights only change for every entry with an uneven index. This structure stems from the concept of the step-and-shoot delivery technique.

VMAT The general structure of VMAT plans is similar to the structure of step-and-shoot IMRT plans, but only one field is used. However, the number of control points is different in this case compared to IMRT. In the VMAT case, the total amount of control points corresponds to the total amount of apertures in the arc. The cumulative meterset weight of the first control point has to be set to zero.

This structure stems from the concept of VMAT. Delivery during the leaf motion is included in the calculation, which means that the beam does not have to be turned off in between apertures (which is the definition of VMAT). Thus, there is no need for additional control points to move the MLC leaves without MU delivery.

When a VMAT plan is calculated in opt4D, the first aperture is assigned a weight as well. This is problematic when importing the plan to Eclipse, as the treatment starts with the MLCs corresponding to the shape of aperture 1, which, as aforementioned, is assigned a cumulative meterset weight of 0 per default. This means that the delivery starts when the arc between aperture 1 and aperture 2 is crossed. To still incorporate the weight of aperture 1, an approximation was made. The cumulative meterset weight

of aperture 2 was set to be the sum of the meterset weights of aperture 1 and aperture 2.

Weight normalization The cumulative meterset values in an ESAPI plan need to start at zero and are not allowed to exceed the maximum value of one due to the definition given in the previous section. Thus, weights extracted from opt4D need to be normalised to be within this range. This could usually be done by dividing each weight by the sum of all weights within the same field. However, all weights generated in opt4D during the weight optimization step are given with respect to the MU values exported from Eclipse. Thus, instead of normalizing the meterset values based solely on the total field weight, the ratio of the MU values for each aperture need to be taken into account as well.

Weight re-assignment is thus done using the following formula:

$$\text{Aperture Weight}_i = \frac{w_i * \text{MU}_i}{\sum_{j \in \text{Field}} w_j * \text{MU}_j} \tag{2.6.1}$$

where w_i denotes the weight of the i-th aperture of a field, calculated in opt4D, and Field denotes the set of all apertures within the same field as i. MU at index i corresponds to the i-th MU value exported from Eclipse.

It can also occur that beams have a weight of $w_i < 0.0001$, meaning that their contribution to the calculation step in Eclipse is negligible. Such apertures are excluded from the calculation step entirely, as they are not beneficial for the overall result. A weight below 0.0001 is outside of the precision range of Eclipse, and thus does not contribute to the dose matrix, i.e., the dose matrix solely contains zeros.

The weight re-optimization is carried out using the exported apertures, whose dose values were calculated using a default MU value within Eclipse. Thus, the resulting weights for each aperture are given with respect to the MU value used during the export step. When the plan is calculated in Eclipse, it is important that the MU values for the beam are set accordingly. To guarantee that the MU values for the calculation are set correctly, one can sum the weights and MU values for each aperture in a beam, and store the result as a preset value.

As mentioned previously in section 2.4, such a preset value calculation can not be carried out for IMRT beams containing a single aperture. Instead, the beam is calculated with a default value, and the MU value corresponding to the beam is stored. After the calculation has finished, one can normalize the beam to adjust the default MU value to the stored value. To do this, the following formula is applied to the beam

$$\text{Beam weight}_i = \frac{w_i * \text{MU}_i}{\text{MU}_{\text{default,i}}} \tag{2.6.2}$$

where $MU_{default,i}$ denotes the default MU value set by Eclipse after the dose calculation, and MU_i refers to the stored MU value.

Changes on the beam weight only have to be carried out on IMRT beams with single apertures, as this step is mitigated by the use of preset values for IMRT with multiple apertures. For VMAT plans, this problem does not arise, due to all apertures being added to an arc beam.

2.6.1. Workflow description

Prior to uploading the plan to Eclipse, two files need to be generated in MATLAB. The first file contains all the optimized beam weights calculated by opt4D during the weight re-optimization step. To do so, one can use the following command:

Command 2.10: Get weight file

Get_Weight_File(Folder, Output, Instances)

- Folder is the location where the outputs of the plan you just ran are stored
- Output is the location where the generated weight file should be stored
- Instances is an optional parameter, set to 1 per default. If STF is applied, this parameter needs to correspond to the amount of different instances.

If multiple instances have been exported, multiple MU files have been generated as well. To merge all of the results into one file, the command

Command 2.11: Get MU file

Get_MU_File(Folder, Output, NameRoot)

- Folder is the location where the MU files exported from Eclipse are stored.
- Output is the location where the generated MU file should be stored.
- NameRoot corresponds to the NameRoot chosen for the Export in Eclipse.

can be used. This step can be ignored for uniformly fractionated plans.

When navigating to the calculation section of opt4D2Eclipse's GUI, there are two additional textboxes which have not yet been used. In the weight optimized upload, the file-path of both the weight file and the MU file can be added here. Figure 15 in section 2.3 displays them. Then, the "Calculate with Weight" button should be clicked. This will generate the final weight re-optimized plan in Eclipse.

2.7. Plan evaluation

Eclipse is not capable of calculating the BED of a patient on its own. To do this, a script by the GitHub user "brjdenis" was downloaded and added to Eclipse [7]. This script is capable of converting a given dose distribution to EQD2 or BED, given a table of

α/β values for each structure. By implementing this script into the workflow, it is now possible to convert the dose distribution for each instance into BED, which can then be summed up using a plan sum in Eclipse.

2.7.1. Workflow description

The plan which should be converted to BED needs to be dragged into the window of the Eclipse GUI to load it. After this is done, the script "EQD2Converter" can be selected from Tools → Scripts, similarly to the script "Opt4D2Eclipse" in section 2.3. Figure 25 displays the GUI of the script.

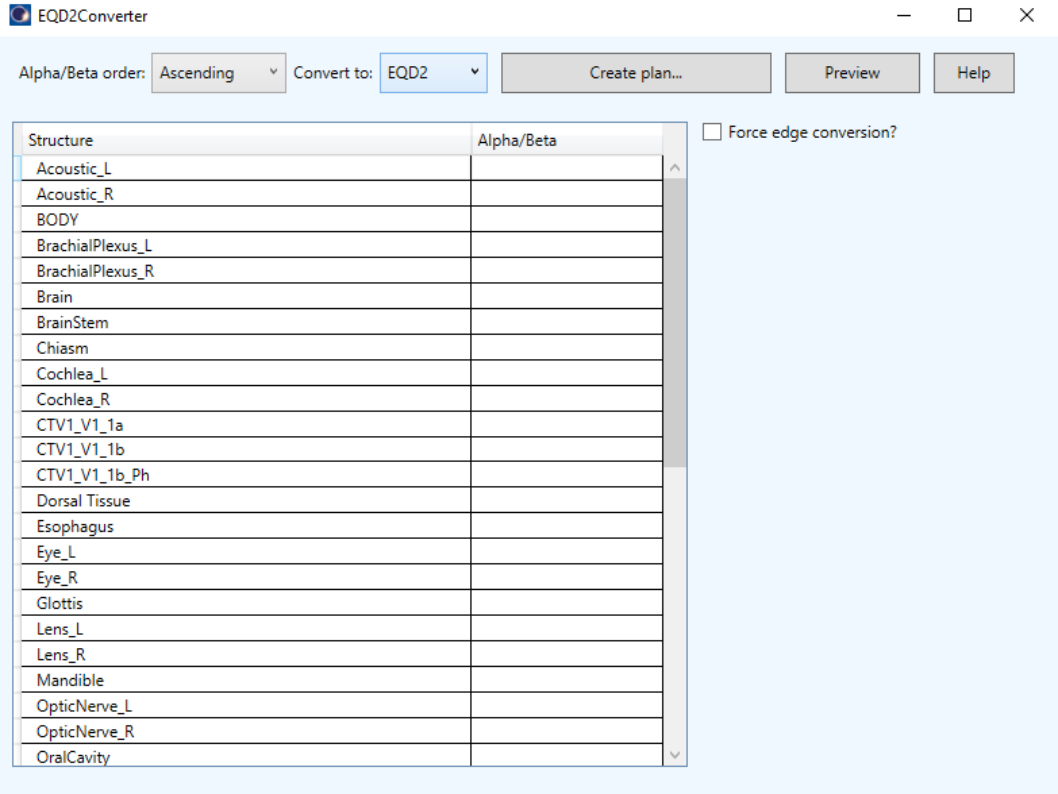


Figure 25: GUI of the script "EQD2Converter" by GitHub user "brjdenis", used to convert physical dose distributions to BED or EQD2 distributions.

Clicking EQD2 next to "convert to:" opens a drop-down menu, where BED should be selected instead of EQD2. Then, the α/β values for all structures of the patient need to be added to the respective field next to the structure name. After all information has been added, the "Create plan..." button can be clicked. The script will then automatically create the BED distribution, and display a short report after the calculation is done. At this point, the script can be closed.

3. Testing of the Workflow

To test the feasibility of the workflow to generate high quality treatment plans in Eclipse starting from opt4D, both a state-of-the-art treatment plan for a prostate patient and an STF plan for a patient with a large head-and-neck-tumour have been investigated.

3.1. Patients

The following section contains details about the patients and the planning objectives used for the plan optimization.

3.1.1. Prostate Patient

To validate the workflow, a prostate patient was inspected. The objectives for an existing plan in the clinical system were extracted and used as an input for the direct aperture optimisation in opt4D. The same objectives were used to create plans in Eclipse for the comparison. The plans were optimized based on physical dose.

Two plans were created in both Eclipse and opt4D for the validation. For the step-and-shoot IMRT plan, an Eclipse plan was made were the leaf sequencing parameters were altered such that the resulting plan had 60 apertures. The corresponding plan calculated in opt4D contained 60 apertures as well. For the VMAT plan, 181 apertures were used in both Eclipse and opt4D for the comparison, for an arc from 180.1 to 179.9°. The prescribed dose for both treatment types was set to 30 x 2 Gy.

The reasoning behind choosing a prostate patient was that prostate cancer is a well researched and clinically standardized type of cancer. This allowed for a more precise assessment of potential shortcomings of the workflow, as well as possible benefits and disadvantages of the in-house optimizer.

Table 3 contains information about the target structures of the patient. Figure 26 displays a slice of the CT of the prostate patient.

Structure	**Volume** [cm^3]
PTV1 V1 1a	126.4
CTV1 V1 1a	34.7

Table 3: Spatial information about the target structures of the prostate patient.

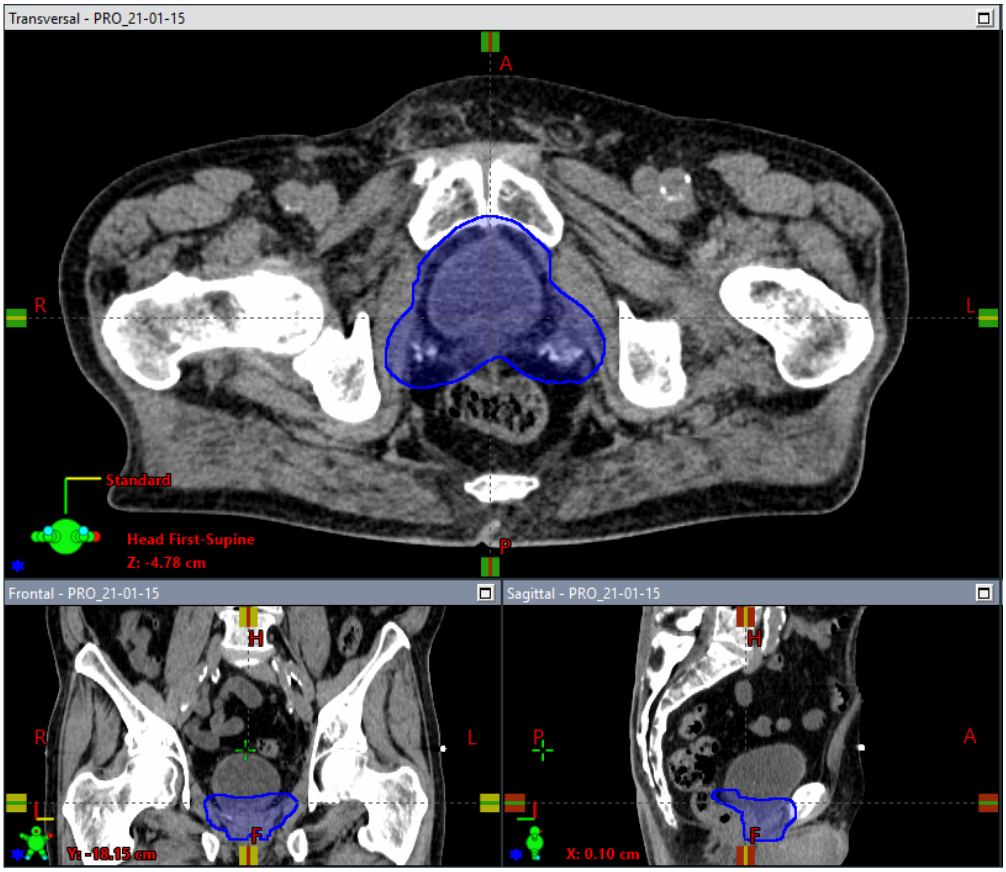


Figure 26: CT of the prostate patient displaying the contour of the target structure PTV1 V1 1a (blue)

Optimization objectives for Prostate Patient The objectives used for the optimization problem of the prostate patient can be found in Table 4 and Table 5:

Structure	Objective	Dose [Gy]	Weight
PTV1_V1_1a	Upper	63.00	32
	Lower	58.00	32
Bladder	Upper	62.00	1
Bladder_PTV	Upper	62.00	1
	Mean	0	1
Bowel	Upper	2.5	1
Femur Head Left	Upper	27.00	1
Femur Head Right	Upper	27.00	1
Penile Bulb	Upper	55.0	1
PTV1a_Ring_PH	Upper	56.00	5
RectalWall_PH	Upper	16.00	13
Rectum	Upper	60.00	13
Sigma	Upper	3.00	1

Table 4: Planning objectives for the prostate patient with 30x2 Gy fractionation scheme

Additionally, an normal tissue objective (NTO) was defined with the following specifications for the dose falloff:

Structures	start distance	start dose	end dose	falloff parameter	weight
Body	0.25	57.0	20.0	0.2	5

Table 5: Normal tissue objective for the prostate patient

Scope of the test The test using the prostate patient was implemented to evaluate the capability of the workflow to create a plan comparable in quality to a plan made entirely in Eclipse for both step-and-shoot IMRT and VMAT. This should signify whether the optimization workflow works in its current state and can produce results that obey the objectives defined for it, i.e. that there are no steps in the workflow which malfunction and deteriorate the obtained result. Furthermore, the minimum and maximum dose values of the plans made in opt4D and Eclipse should give more insight about the differences in precision.

3.1.2. Large Head-and-Neck Patient

The benefits of the BED-optimisation capabilities of the workflow were inspected for a patient with a large Head-and-Neck tumour undergoing a palliative treatment. This type of cancer currently does not have an accepted clinical standard for treatment. Furthermore, the complex shape of the tumour made an interesting case to inspect the potential of spatio-temporal fractionation.

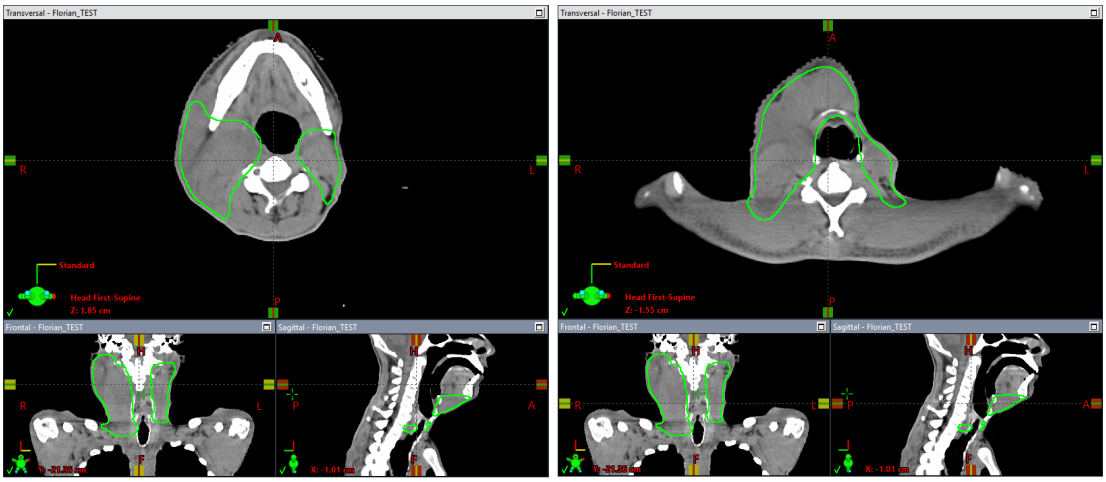
Once again, the treatment objectives are based on an existing accepted plan from Eclipse. The objectives were then translated into BED values and optimized in opt4D.

The α/β values for target structures was set to 10, the value for healthy tissue was set to 4. The prescribed dose was set to 8 x 2 Gy, which corresponds to 8 x 2.4 Gy_{10} in terms of BED in the target. 40 apertures were generated per fraction in opt4D.

Table 6 contains information about the target structures present for the patient, Figure 27 displays the CT for two different slices.

Structure	**Volume** [cm^3]
CVT1 V1 1b Ph	186.9
PTV1 V1 1b Ph	630.6
PTV1 Ph	4.2

Table 6: Spatial information about the target structures of the head-and-neck patient.



(a) (b)

Figure 27: Two slices of the CT for the head-and-neck patient, containing the contour of the target structure PTV1 V1 1b Ph (green).

Optimization objectives for Head-and-Neck Patient The objectives used for the optimization problem of the H&N-tumour patient can be found in Table 7 and Table 8:

Structure	Objective	$BED_{\alpha/\beta}$ [Gy]	Weight
CTV1_V1_1b_PH	Upper	20.124	52
	Lower	18.936	24
PTV1_PH	Upper	20.016	70
	Lower	18.936	190
PTV1_V1_1b_PH	Upper	20.160	327
	Lower	18.948	237
Brachial Plexus Left	Upper	11.370	2
Brachial Plexus Right	Upper	16.395	2
Brain Stem	Upper	8.205	6
Cochlea Right	Upper	0.54	1
Dorsal Tissue	Upper	19.215	34
Glottis	Upper	13.650	2
Mandible	Upper	22.860	4
Oral Cavity PH	Upper	18.165	17
Parotid Left PH	Upper	9.765	1
Parotid Right PH	Upper	5.160	2
Pharynx Constrictor PH	Upper	11.160	2
Ring_1b	Upper	24.330	34
Salivary Gland Left	Upper	12.045	1
Soft Palate	Upper	11.385	1
Spinal Cord	Upper	13.020	17
Thyroid	Upper	9.195	1
Tissue PH	Upper	13.650	17

Table 7: Objectives for Head-and-Neck tumour patient with 8x2 Gy fractionation scheme. The α/β values for the target structures are set to 10, for healthy tissue they are set to 4

Additionally, an NTO was defined with the following specifications for the dose falloff:

Structures	start distance	start dose	end dose	falloff parameter	weight
BODY	0.25	23.940	8.400	0.14	4

Table 8: Normal tissue objective for the H&N tumour patient

Scope of the test This test was implemented to show that the workflow does not only work for plans optimized on physical dose (section 3.1.1), but also produces acceptable results when the optimization is performed on the BED instead.

Furthermore, this test was used to investigate whether spatio-temporal fractionation (STF) is beneficial for the chosen patient, compared to a plan made in opt4D using uniform fractionation (UNI). Both plans were optimized on BED.

3.2. Quality measurement

It can be rather cumbersome to compare two plans and determine which one is deemed "better". For this thesis, the plans were compared using extracted DVH files from Eclipse, based on the quantities min/max/mean dose.

Eclipse allows the export of a DVH as a *.txt* file. Such a file contains dose-volume statistics for each structure. To parse these files, a MATLAB script was implemented, such that the different quantities could be inspected in greater detail.

To compensate for the different Dij's used in the calculation step in CERR and Eclipse and other influence on the dose calculation, such as dose leakage, and to make the comparison more robust, another method of quantification was implemented as well, using $d_{2\%}$ values [6]. This refers to the highest dose that 2% of the volume receives.

$d_{2\%}$ calculation was implemented using MATLABs intern *interp* function, based on a neighbourhood of the desired volume fraction, in this case, 2%.

For the results section, the quantities Homogeneity index (HI) and Conformity index (CI) of the target structure will be defined as follows [7]:

$$\mathrm{HI} = \frac{d_{95\%}}{d_{5\%}} \tag{3.2.1}$$

$$\mathrm{CI} = \frac{V_{\mathrm{RI}}}{V_{\mathrm{PTV}}} \tag{3.2.2}$$

V_{PTV} denotes the total volume of the target, and V_{RI} denotes the total volume receiving the reference isodose. In accordance with ICRU recommendations, the reference isodose volume was set to 95% of the prescribed dose [8].

The HI measures the ratio between the dose that at least 95% of the volume receives and the dose that at least 5% of the volume receives. A high HI value indicates that the dose is evenly distributed throughout the target volume, whereas a low HI value may indicate under-dosage.

The CI measures the ratio between the volume receiving at least the prescribed dose and the volume of the target structure. Thus, the CI measures how well the dose conforms to the tumour shape and size. A value below 1 indicates that parts of the tumour do not receive the prescribed dose, whereas a value above 1 indicates that healthy tissue is not spared sufficiently.

Thus, a result as close to 1 as possible is desirable for both quality measurements.

4. Results

4.1. Prostate Patient

For the prostate patient, both a step-and-shoot IMRT plan and a VMAT plan were made using the proposed workflow. These plans were compared to a step-and-shoot IMRT plan and a VMAT plan made in Eclipse, respectively.

4.1.1. Step-and-shoot IMRT

The dose in the target structure was normalized to a mean dose of the prescribed 60 Gy for both the plan calculated in opt4D and the plan calculated in Eclipse for this comparison. In opt4D, no normalization is considered.

The DVH for the target structures of the prostate patient can be found in Figure 28, the DVH for the organs with tumour overlap in Figure 29, and the DVH for the organs without tumour overlap in Figures 30/31. Table 9 displays the dose statistics, whereas table 10 reports HI and CI.

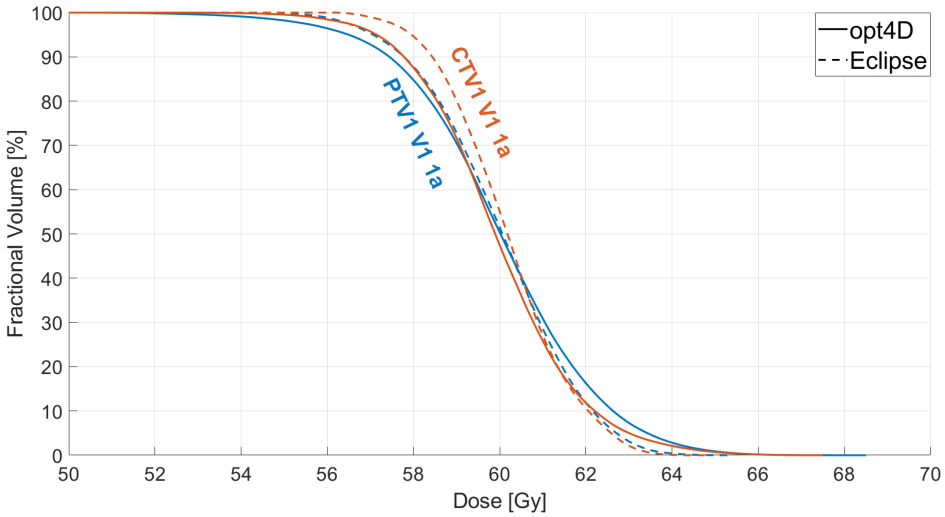


Figure 28: Comparison of the DVH for targets structures of plans made in Op4D and Eclipse. The x-axis was cropped a region of 50-70 Gy to focus on the relevant range.

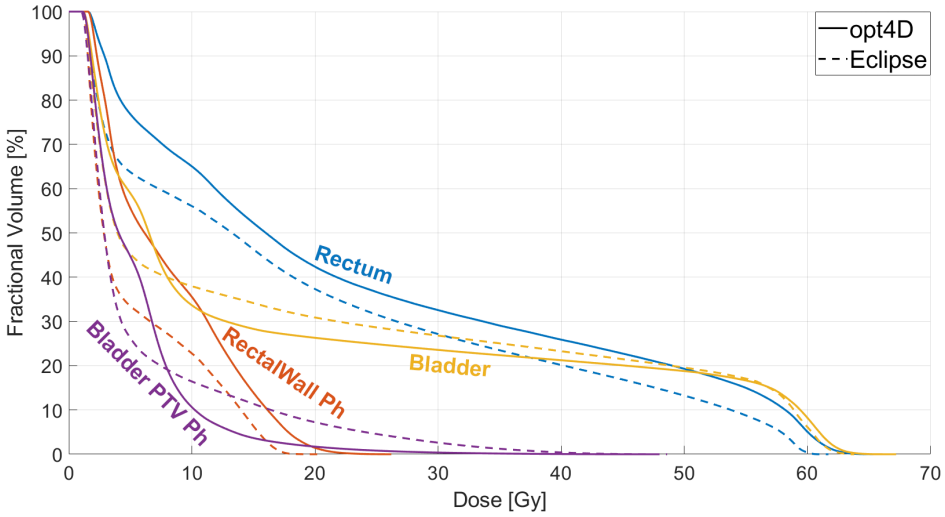


Figure 29: DVH comparison for OARS with target overlap and the corresponding part of the structure without the target structure.

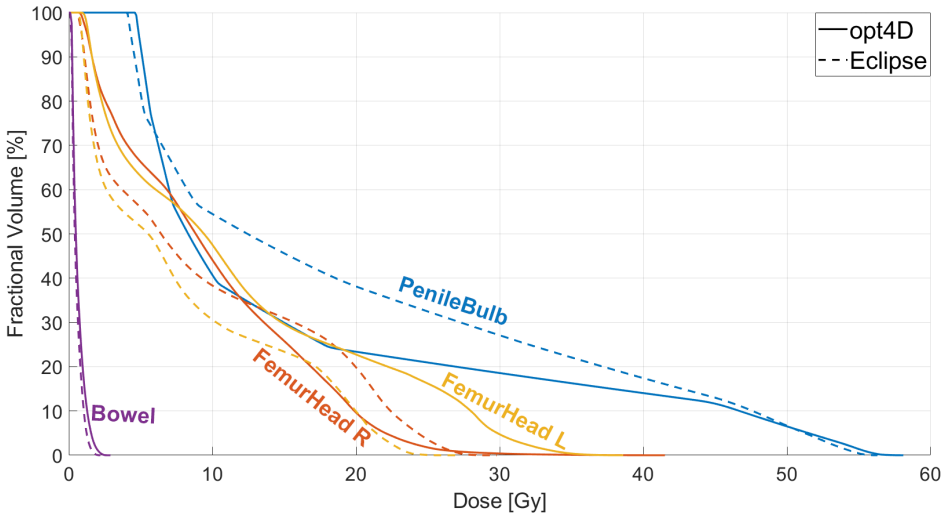


Figure 30: DVH comparison for OARS without target overlap

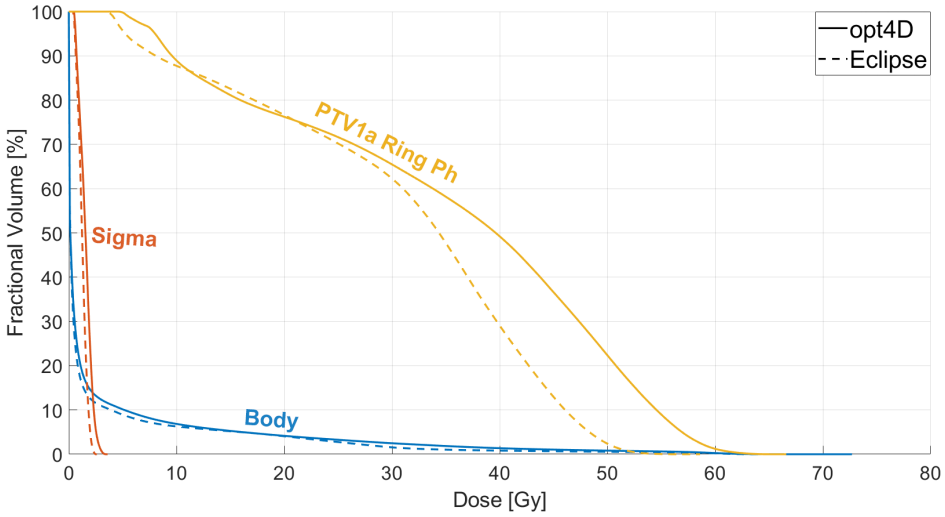


Figure 31: DVH comparison for OARS without target overlap

Structure	opt4D d_{min} [Gy]	opt4D d_{max} [Gy]	opt4D d_{mean} [Gy]	opt4D $d_{2\%}$ [Gy]	Eclipse d_{min} [Gy]	Eclipse d_{max} [Gy]	Eclipse d_{mean} [Gy]	Eclipse $d_{2\%}$ [Gy]
CTV1 V1 1a	52.761	67.465	59.959	64.060	55.826	64.827	60.189	63.030
PTV1 V1 1a	50.006	68.430	60.000	64.321	52.394	65.389	60.000	63.246
PTV1a Ring Ph	4.627	66.580	35.498	59.071	3.325	58.595	30.893	**50.338**
Rectum	1.546	66.263	23.710	61.384	1.151	61.630	19.728	**59.061**
Rectal Wall Ph	1.546	26.125	7.959	19.503	1.151	20.312	5.470	**16.423**
Bladder	1.140	67.104	17.555	62.078	1.019	63.746	18.276	**61.334**
Bladder PTV Ph	1.140	47.334	**5.498**	**18.934**	1.019	48.593	5.955	31.959
Penile Bulb	4.613	58.009	16.016	54.467	4.032	56.246	19.643	**53.519**
Femur Head R	0.712	41.457	9.840	**24.688**	0.502	29.702	9.552	25.870
Femur Head L	0.867	38.539	11.635	32.332	0.586	26.873	7.875	**22.757**
Bowel	0.149	2.820	0.640	1.855	0.137	2.155	0.535	**1.474**
Body	0	72.797	2.512	33.844	0	65.389	2.171	**28.017**
Sigma	0.465	3.499	1.543	2.808	0.395	2.502	1.241	**2.153**

Table 9: min/max/mean dose and $d_{2\%}$ values for plans made in opt4D and Eclipse. All quantities are given in Gy. Better results are highlighted in bold.

Structure	HI opt4D	HI Eclipse	HI Ratio [%]	CI opt4D	CI Eclipse	CI Ratio [%]
PTV1 V1 1a	0.887	**0.910**	97.5	1.328	**1.019**	130.3

Table 10: Dose quantity comparison for opt4D and Eclipse based plans according to eq. 3.2.1 and 3.2.2 for the target volume. The ratio is given as opt4D/Eclipse. Better results are highlighted in bold.

The plan generated in opt4D using the proposed workflow performs better than the plan generated entirely in Eclipse for the OAR "Bladder PTV Ph". For the target structures, as well as the OARs "Rectum", "Bladder", "Penile Bulb", "Femur Head R" and "Bowel", a similar performance was achieved. The plan made entirely in Eclipse outperforms the plan made using the proposed workflow regarding the OARs "PTV1a Ring Ph", "Rectal Wall Ph", "Femur Head L", "Sigma" and the "Body" structure.

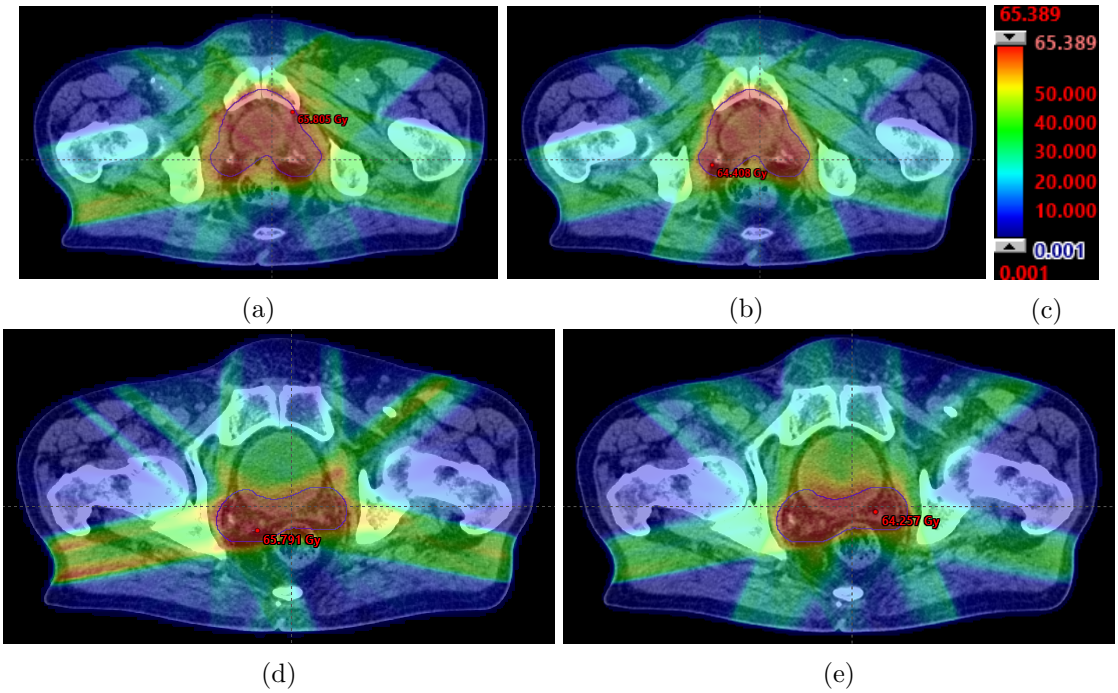


Figure 32: Comparison of dose distributions of a plan made in (a,d) opt4D and a plan made in (b,e) Eclipse for two different slices displayed using the same (c) colorbar.

Figure 32 shows the dose distributions of a plan made entirely in Eclipse and a plan made using the proposed workflow for the same slice and the same colormap. Analysing the image qualitatively, the dose distributions appear to be quite similar, with slight discrepancies outside of the target structure, as well as a higher entrance dose for the beam angle 257.1°.

4.1.2. VMAT

Similarly to the step-and-shoot IMRT result, the dose in the target structure was normalized to a mean dose of the prescribed 60 Gy for both the plan calculated in opt4D and Eclipse for this comparison.

The DVH for the target structures of the prostate patient with a VMAT treatment can

be found in Figure 33, the DVH for the organs with tumour overlap in Figure 34, and the DVH's for the organs without tumour overlap in Figure 35/36. Table 11 displays the dose statistics, and Table 12 displays reports HI and CI.

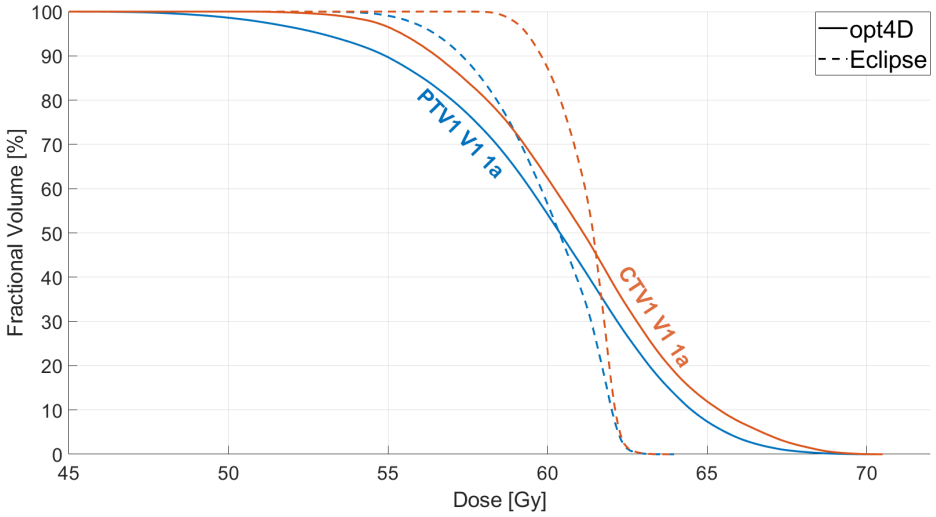


Figure 33: Comparison of the DVH for targets structures of plans made in Op4D and Eclipse. The x-axis was cropped a region of 45-72 Gy to give more insight into the curve characteristics.

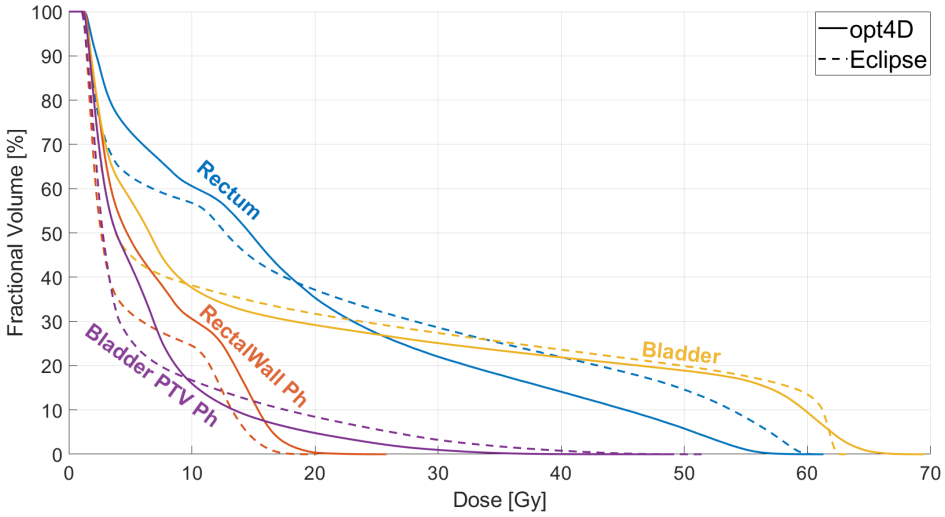


Figure 34: DVH comparison for OARS with target overlap and the corresponding part of the structure without the target structure.

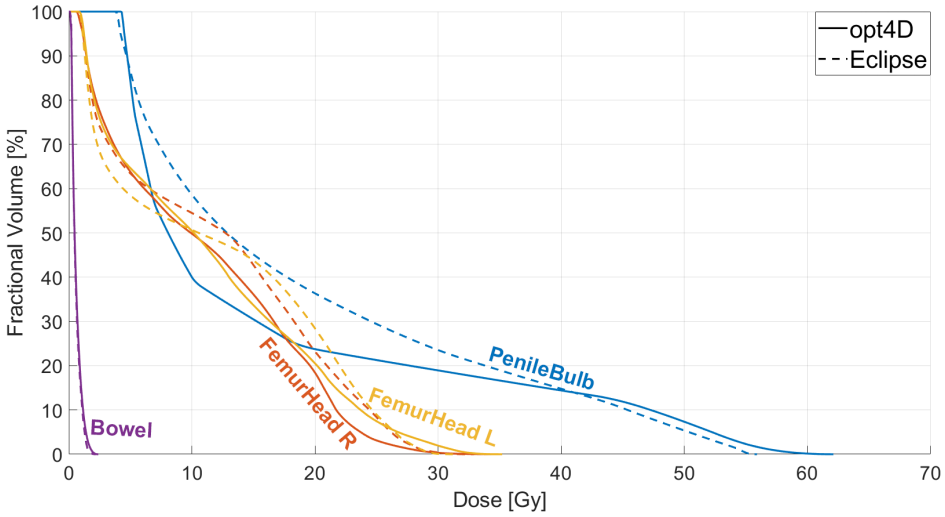


Figure 35: DVH comparison for OARS without target overlap

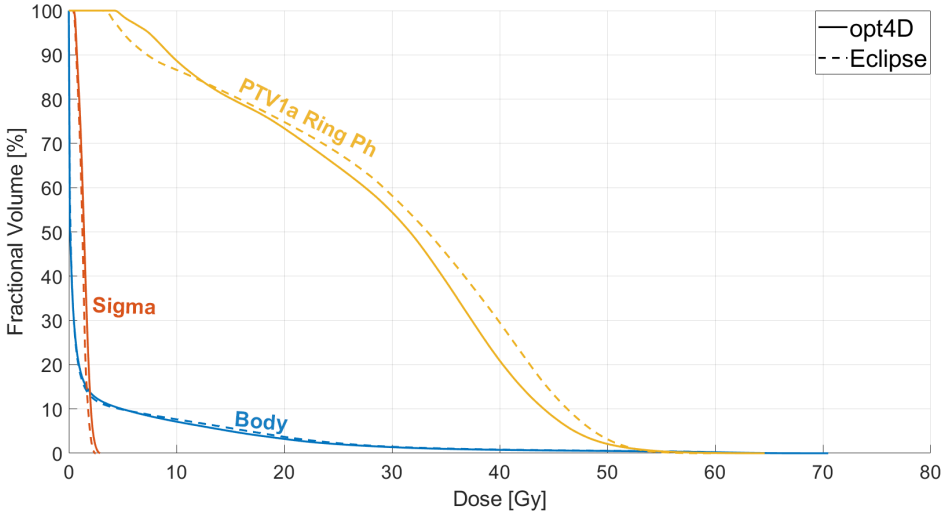


Figure 36: DVH comparison for OARS without target overlap

Structure	opt4D				Eclipse			
	$\mathbf{d}_{\text{min}}$ [Gy]	$\mathbf{d}_{\text{max}}$ [Gy]	$\mathbf{d}_{\text{mean}}$ [Gy]	$\mathbf{d}_{2\%}$ [Gy]	$\mathbf{d}_{\text{min}}$ [Gy]	$\mathbf{d}_{\text{max}}$ [Gy]	$\mathbf{d}_{\text{mean}}$ [Gy]	$\mathbf{d}_{2\%}$ [Gy]
CTV1 V1 1a	50.768	70.507	61.023	67.883	57.288	63.812	61.176	62.442
PTV1 V1 1a	44.486	70.507	60.000	66.709	49.506	63.988	60.000	62.399
PTV1a Ring Ph	4.150	64.591	29.057	**50.172**	3.038	59.891	30.275	51.126
Rectum	1.237	61.285	18.343	**53.830**	1.070	61.289	19.944	58.492
Rectal Wall Ph	1.237	25.710	7.115	17.980	1.070	20.054	5.261	**15.713**
Bladder	1.140	69.447	18.420	63.617	1.045	63.037	18.737	**62.094**
Bladder PTV Ph	1.140	48.834	6.178	**26.163**	1.045	51.315	**6.138**	34.053
Penile Bulb	4.262	62.032	16.056	55.406	3.770	55.855	19.045	**53.551**
Femur Head R	0.620	34.616	10.887	**26.378**	0.654	32.821	11.956	27.864
Femur Head L	0.807	35.104	11.260	29.936	0.776	31.530	11.770	**27.985**
Bowel	0.132	2.355	0.594	1.658	0.159	2.138	0.566	**1.458**
Body	0	70.507	2.180	**25.263**	0	63.988	2.282	26.338
Sigma	0.499	2.897	1.423	2.451	0.404	2.460	1.256	**2.143**

Table 11: min/max/mean dose and $d_{2\%}$ values for plans made in opt4D and Eclipse. All quantities are given in Gy. Better results are highlighted in bold.

Structure	HI			CI		
	opt4D	**Eclipse**	**Ratio** [%]	**opt4D**	**Eclipse**	**Ratio** [%]
PTV1 V1 1a	0.807	**0.908**	88.9	0.869	**0.969**	89.7

Table 12: Dose quantity comparison for opt4D and Eclipse based plans according to eq. 3.2.1 and 3.2.2 for the target volume. The ratio is given as opt4D/Eclipse. Better results are highlighted in bold.

The VMAT plan made in opt4D displays a better performance for the OARs "PTV1a V1 Ring Ph", "Rectum", Femur Head R" and the entire body structure, whereas comparable results to the Eclipse made VMAT plan were achieved for the OARs "Bowel" and "Sigma". For the OARs "Rectal Wall Ph", "Bladder", "Penile Bulb" and "Femur Head L", the Eclipse made VMAT plan performed better. Regarding the OAR "Bladder PTV Ph", the mean dose objective was fulfilled better in the Eclipse plan, whilst in terms of the maximum dose constraint, the opt4D plan displayed a better performance.

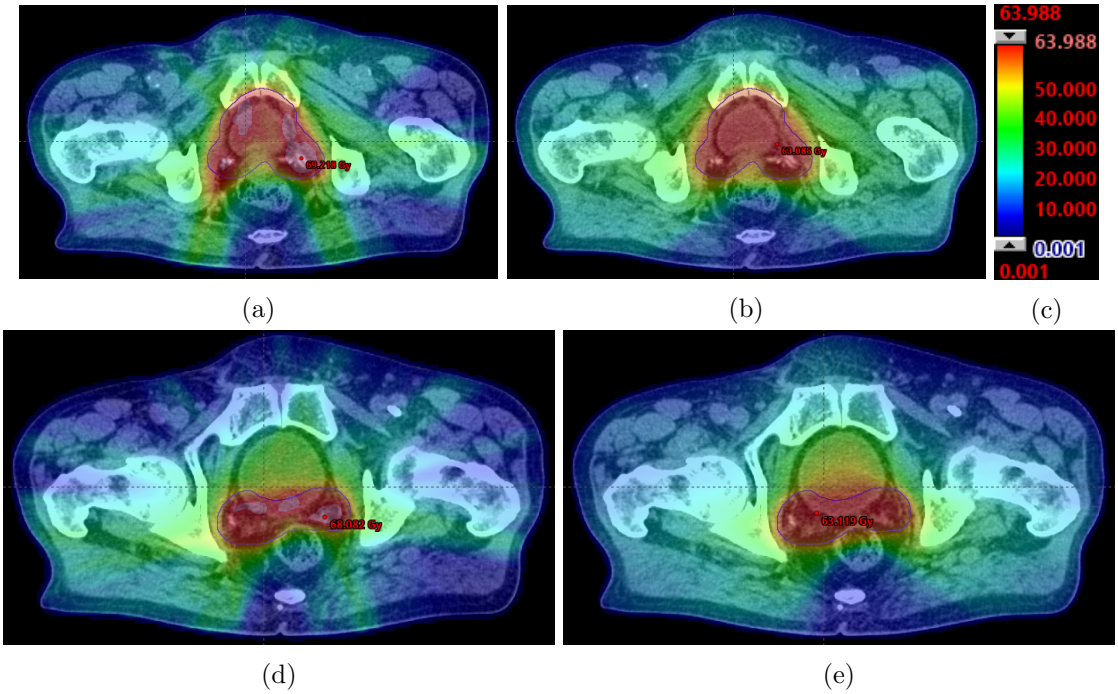
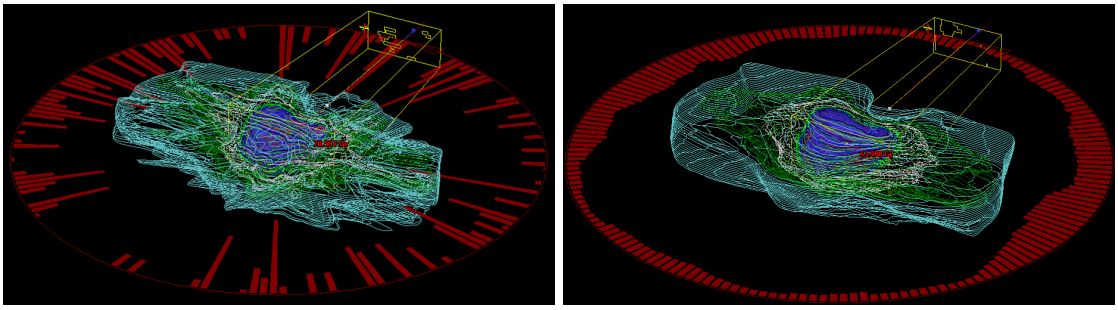


Figure 37: Comparison of dose distributions of a plan made in (a,d) opt4D and a plan made in (b,e) Eclipse for two different slices displayed using the same (c) colorbar. The same slices were taken as for Figure 32.

Figure 37 shows the cumulative dose distribution for both the plan made using the proposed workflow and the Eclipse plan for the same slices as in Figure 32. Inspecting the distributions qualitatively, the dose distributions seem to be quite similar in the VMAT case as well. A difference that can be seen from this qualitative comparison is that the plan made in Eclipse seems to distribute the dose more uniformly regarding the beam angles, whilst the opt4D plan seems to select arcs where more dose should be delivered, whilst ignoring other arcs entirely.

This is further illustrated in Figure 38, which displays the visualization of the delivered MU of each arc. For the (a) opt4D plan, many arcs have no dose contribution, whilst other arcs display large spikes. For the (b) Eclipse plan, no such spikes are present, and the dose is distributed more uniformly. On this image, one can also see the iso-lines of the dose distribution, which are more uniformly distributed for the Eclipse made VMAT plan as well.



(a) (b)

Figure 38: Comparison of the monitor units delivered per arc for (a) opt4D and (b) Eclipse made VMAT plans.

Regarding the total treatment time, the delivery of the VMAT plan made in Eclipse is 60 seconds, whilst delivering the VMAT plan made in opt4D would take approximately 131 seconds. This corresponds to a ratio of total treatment time of approximately 218.3%.

4.2. Large head-and-Neck Patient

The DVH for the target structures can be found in Figure 39, the DVH for organs possessing a left/right counterpart is displayed in Figure 40, and the DVHs for the other OARs are found in Figures 41 and 42. All results of this section are given in terms of BED.

Table 13 displays the dose statistics, whilst Table 14 shows the curve quantities for the target structures. All ratios displayed in this section are defined as STF/UNI.

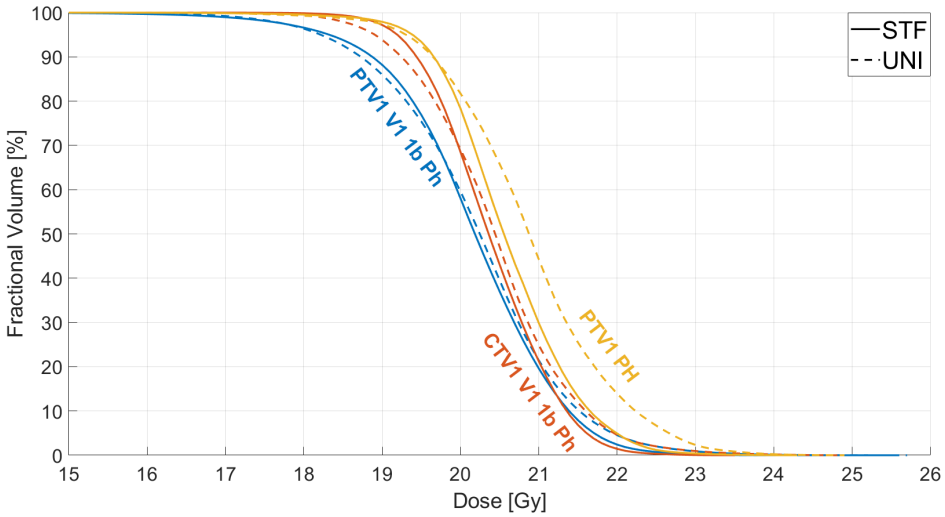


Figure 39: DVH comparison for target structures, cropped to a range of 15 - 26 Gy_{10} for visibility

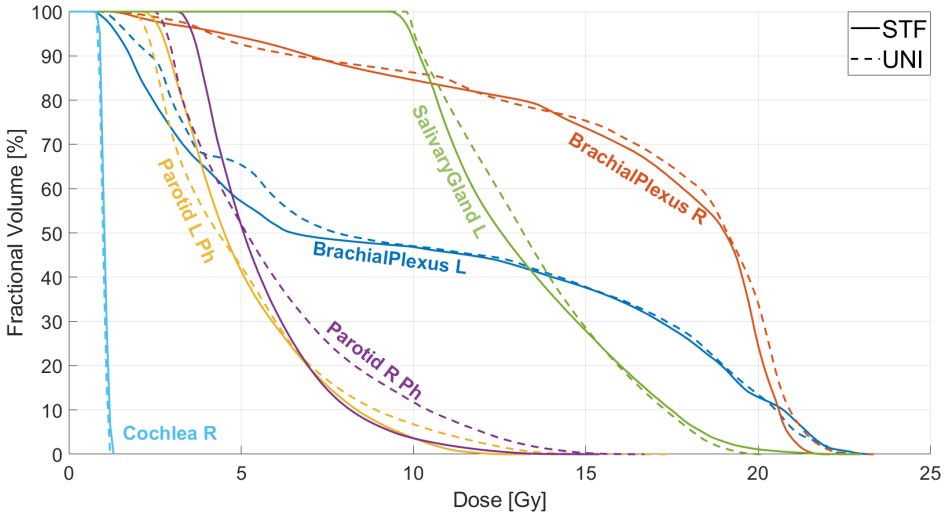


Figure 40: DVH comparison for OARS with a left/right counterpart given in Gy_4

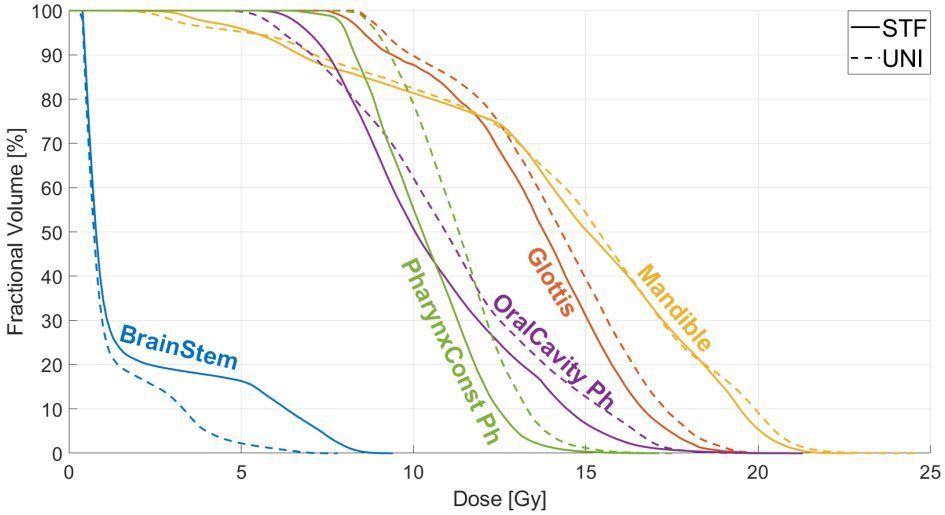


Figure 41: DVH comparison for OARS given in Gy_4

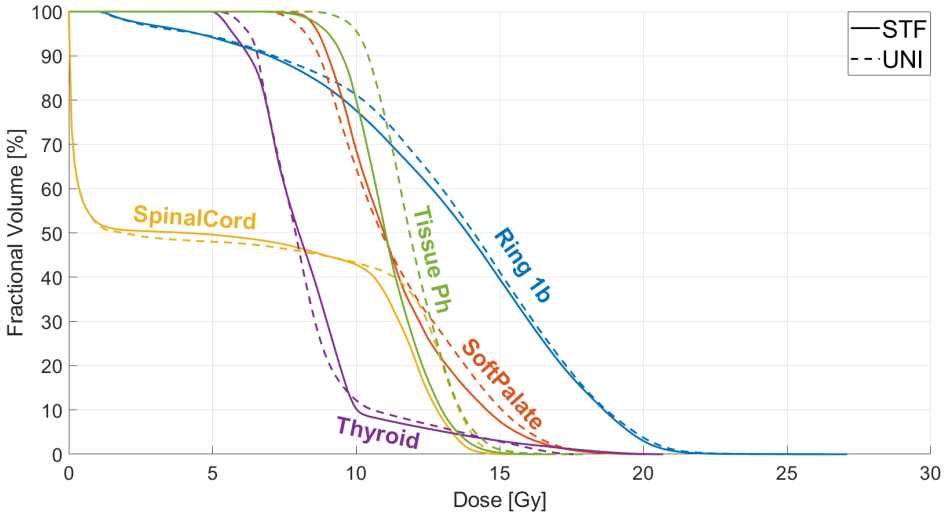


Figure 42: DVH comparison for OARS given in Gy_4

Structure	UNI		STF		Ratio (STF/UNI)	
	$BED_{2\%}$ [Gy]	Mean [Gy]	$BED_{2\%}$ [Gy]	Mean [Gy]	$BED_{2\%}$ [%]	Mean [%]
CTV1 V1 1b Ph	22.357	20.422	21.899	20.381	98.0	99.8
PTV1 Ph	23.079	20.894	20.872	20.600	96.8	98.7
PTV1 V1 1b Ph	22.412	20.205	22.073	20.136	98.5	99.7
Brachial Plexus L	22.218	10.667	**21.902**	10.174	98.6	95.4
Brachial Plexus R	22.520	17.121	**21.173**	16.462	94.0	96.2
Brain Stem	**5.059**	1.230	7.907	1.852	156.3	150.6
Cochlea R	**1.187**	0.999	1.284	1.041	108.2	104.2
Dorsal Tissue	17.500	13.805	**17.347**	13.970	99.1	101.2
Glottis	18.878	14.085	**18.124**	13.516	96.0	96.0
Mandible	21.060	14.597	**20.675**	14.319	98.2	98.1
Oral Cavity Ph	17.272	11.182	**16.367**	10.592	94.8	94.7
Parotid L Ph	12.087	5.107	**10.610**	5.138	87.8	100.6
Parotid R Ph	13.485	5.977	**11.033**	5.605	81.8	93.8
Phar. Const. Ph	14.564	11.293	**13.668**	10.349	93.8	91.6
Ring 1b	20.762	13.569	**20.324**	13.237	97.9	97.6
Sal. Gland L	**18.858**	13.441	19.360	13.273	102.7	98.8
Soft Palate	16.950	11.438	**16.731**	11.392	98.7	99.6
Spinal Cord	14.391	6.111	**13.766**	5.856	95.7	95.8
Thyroid	**15.911**	8.209	16.306	8.398	102.5	102.3
Tissue Ph	14.395	11.782	**14.098**	11.139	97.9	94.5

Table 13: $BED_{2\%}$ and mean BED values for plans made in opt4D using uniform and spatio temporal fractionation. All quantities are given in $Gy_{\alpha/\beta}$.α/β is set to 10 for target structures and to 4 for OARs. Lower $BED_{2\%}$ values in the OARs are noted in bold.

The spatio-temporally fractionated treatment performs better in terms of $BED_{2\%}$ than the plan using a uniform fractionation scheme regarding the target structures "PTV1 V1 1b Ph", "CTV1 V1 1b Ph" and "PTV1 Ph", with improvements of up to ∼4.4%. Regarding the OARs, an improvement of more than 5% is achieved for the structures "Brachial Plexus R", "Oral Cavity Ph", "Parotid L Ph", "Parotid R Ph" and "Pharynx Constrictor Ph". Similar results were achieved for "Brachial Plexus L", "Dorsal Tissue", "Glottis", "Mandible", "Ring 1b", "Salivary Gland L", "Soft Palate", "Spinal Cord", "Thyroid" and "Tissue Ph". The uniformly fractionated plan performs better for the OARs "Brain Stem" and "Cochlea R".

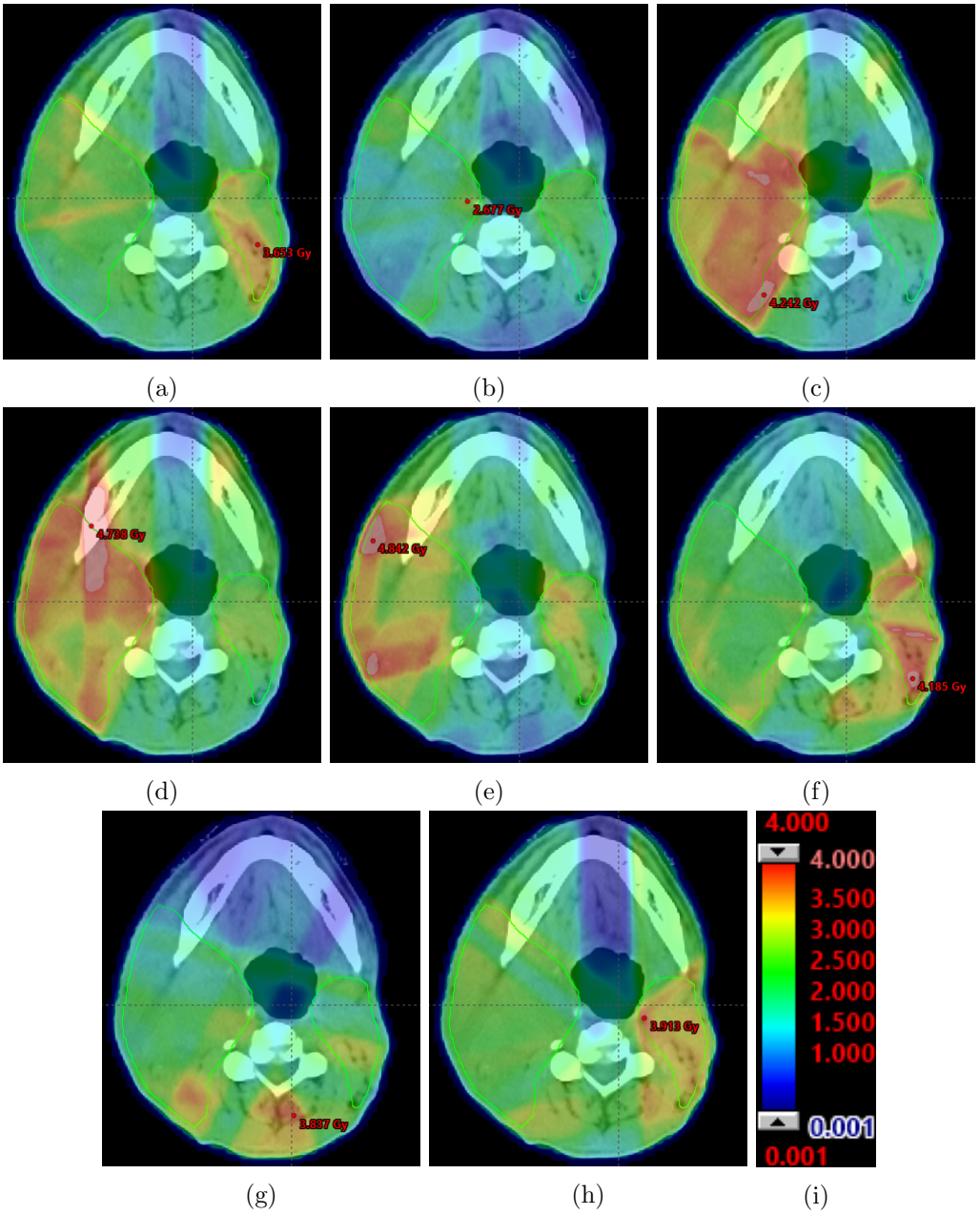
	HI			CI		
Structure	**UNI**	**STF**	**Ratio [%]**	**UNI**	**STF**	**Ratio [%]**
PTV1 V1 1b Ph	0.828	**0.844**	101.9	1.337	**1.327**	99.3

Table 14: Dose quantity comparison for uniformly fractionated and spatio-temporally fractionated plans according to eq. 3.2.1 and 3.2.2. Better CI/HI values are noted in bold.

Comparing two plans to determine which one is better is often a difficult task. To give a better understanding about the differences between the uniformly and spatio-temporally fractionated plans, Table 15 contains the resulting objective values of the weight re-optimization. Whilst these quantities are not necessarily clinically relevant, they can serve as a means of quantification to understand the deviations from the defined plan for both optimization techniques, and may yield more insight about their performance with respect to different structures.

Structures	Objective value		
	UNI	**STF**	**Ratio** [%]
CTV1 V1 1b PH	14.050	**6.664**	47.4
PTV1 PH	73.224	**43.201**	59.0
PTV1 V1 1b PH	216.426	**166.819**	77.1
Brachial Plexus L	94.612	**84.292**	89.1
Brachial Plexus R	**65.510**	79.936	122.0
Brain Stem	**0**	0.016	-
Cochlea R	**0.106**	0.133	124.9
Dorsal Tissue	2.800	**2.389**	85.3
Glottis	23.424	**17.837**	76.1
Mandible	1.499	**1.192**	79.5
Oral Cavity PH	2.956	**2.909**	98.3
Parotid L PH	0.501	**0.101**	20.3
Parotid R PH	12.500	**3.761**	30.1
Phar. Const. PH	5.691	**2.935**	51.6
Ring 1b	2.908	**2.662**	91.5
Sal. Gland L	**15.675**	17.620	112.4
Soft Palate	7.847	**6.675**	85.1
Spinal Cord	11.307	**4.781**	42.3
Thyroid	**4.980**	6.548	131.5
Tissue PH	5.588	**3.530**	63.2
NTO	**0.276**	0.329	119.2
Total	561.88	**454.33**	80.9

Table 15: Objective values for a completed weight re-optimization for both an spatio-temporally and a uniformly fractionated plan. Lower values are noted in bold.

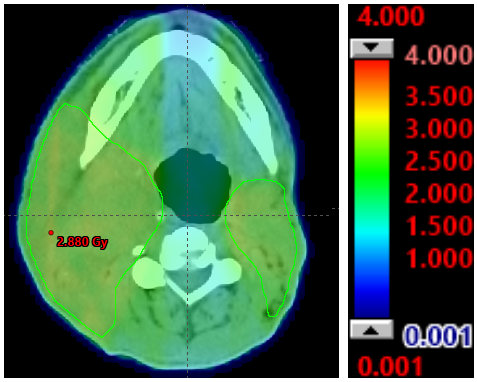


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 43: All (a-h) 8 fractions of the spatio-temporally fractionated plan for the same slice, using the same (i) colormap.



(a) (b)

Figure 44: The fraction of the uniformly fractionated treatment alongside the colorbar which was also used for Figure 37.

Figure 43 displays all 8 fractions of the spatio-temporally fractionated plan (a-h) alongside the colorbar (i), whereas Figure 44 displays the (a) fraction of the uniformly fractionated plan alongside the colorbar (b). It is visible that the spatio-temporally fractionated fractions hypo-fractionate parts of the target structure, opposed to the uniformly fractionated plan. Furthermore, the fractions of the spatio-temporally fractionated plan (Figure 43) show differences regarding their dose distribution.

Figure 45 displays the cumulative BED distributions for both the spatio-temporally fractionated plan and the uniformly fractionated plan. What can be seen when analysing this BED distribution qualitatively is, that the spatio-temporally fractionated treatment appears to achieve better homogeneity in the target structure, whilst depositing less dose to the surrounding healthy tissue. This can be seen especially on panels (a,b) and (e,f). Table 11. supports this claim.

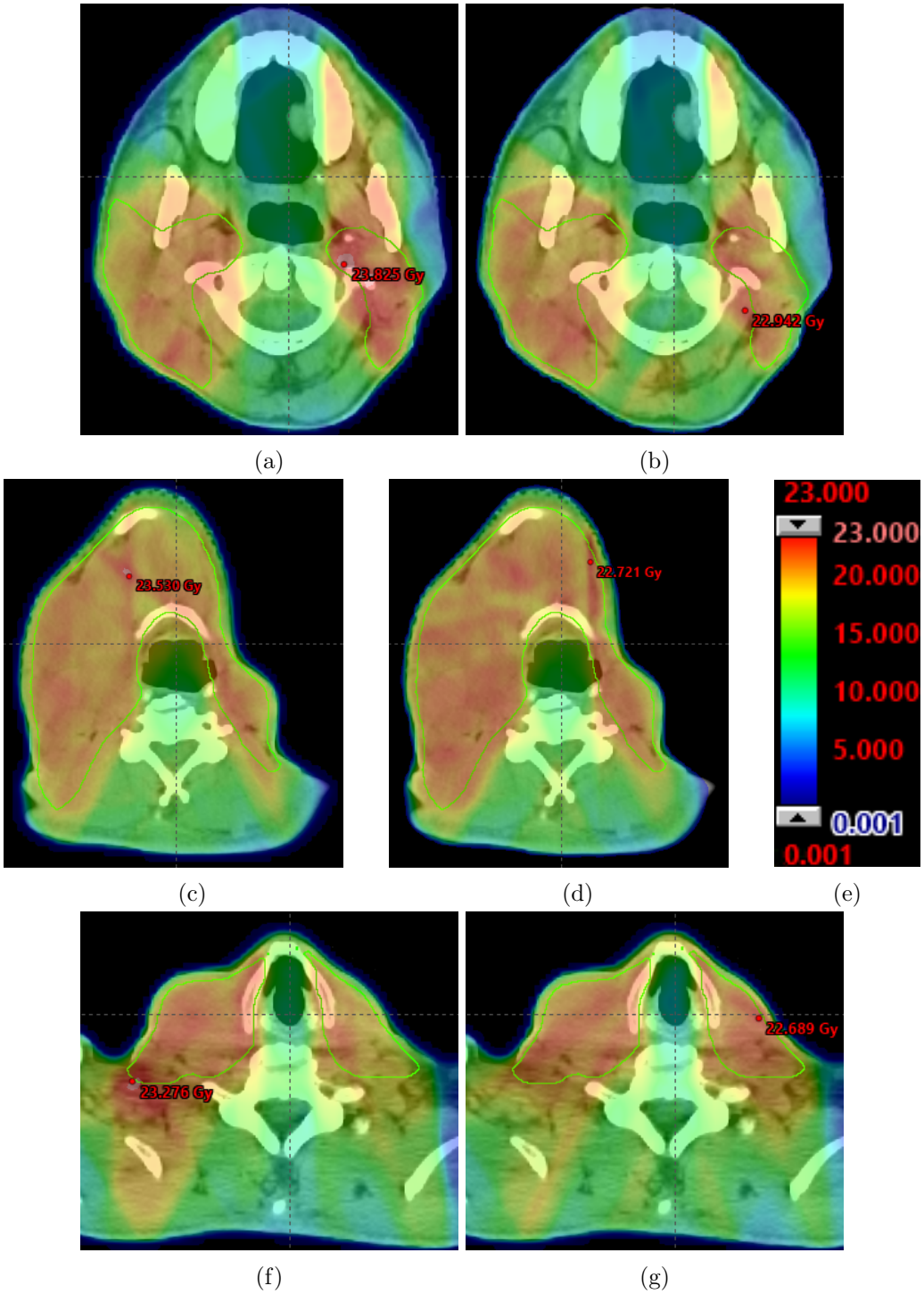


Figure 45: Comparisons between the cumulative BED distribution for the (a,c,f) uniformly fractionated plan and the (b,d,g) spatio-temporally fractionated plan for the same slices using the same (e) colormap.

5. Discussion

5.1. Prostate Patient

Looking at the target structure, PTV1 V1 1a, and the corresponding CTV1 V1 1a of the step-and-shoot IMRT plan, there is a notable difference between the dose tail in the DVH. The opt4D dose tail is larger, and contains hot spots of a small volume which receives a much higher dose than the prescribed maximum dose value of 63.00 Gy. Furthermore, the high value of the conformity index displays a lack of conformity to the target structure.

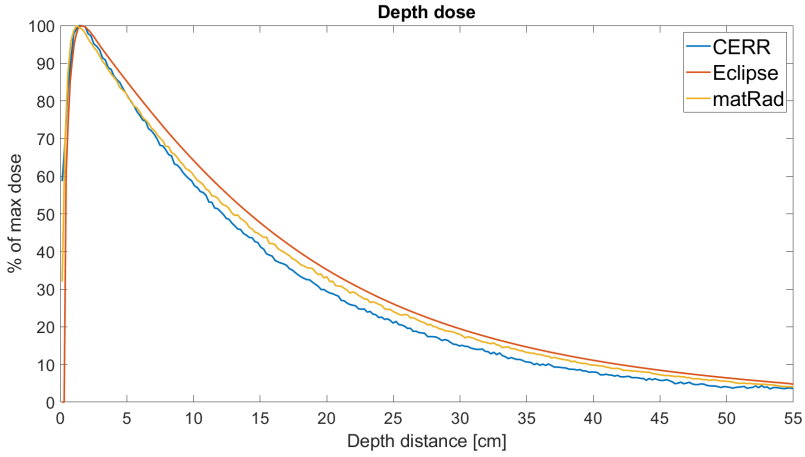
The situation is similar for most OARs, where even organs receiving an overall smaller mean dose, e.g. the penile bulb, display these high dose regions.

A possible cause for this is due to the differences in the Dij's created in CERR and the ones used in Eclipse. The weight re-optimization can only correct this up to a certain threshold, as the shape cannot be modified in this step. To assess this, plans based on Dij's made in CERR and matRad were compared to plans made directly in Eclipse.

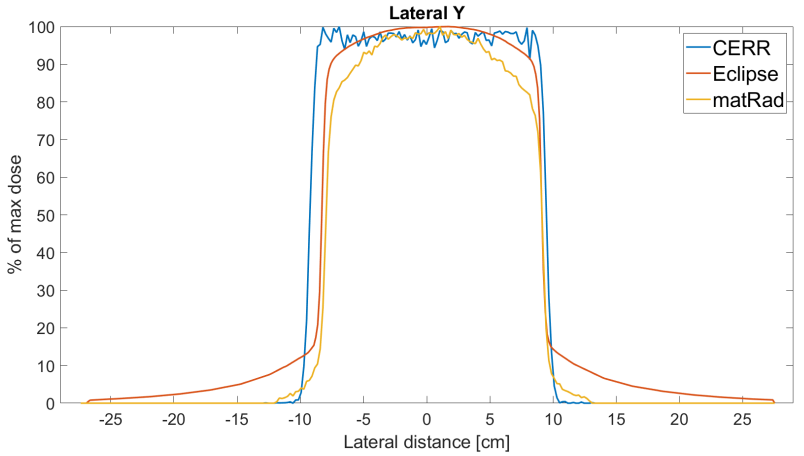
5.1.1. Dose profiles of CERR, matRad and Eclipse

To get more insight into the influence of different Dij's on the plan quality, three plans were made for a 55 x 55 cm water phantom containing a PTV, which was irradiated using a field of size 10 x 10 cm. The Dij's of matRad and CERR were used as input for plan calculation in opt4D with 0 iterations with unit fluence. By setting the number of iterations to 0, no optimization is performed, and the output corresponds to the Dij's given as input. Additionally, a plan was made in Eclipse corresponding to the same setting. The plans were uploaded to CERR for comparison.

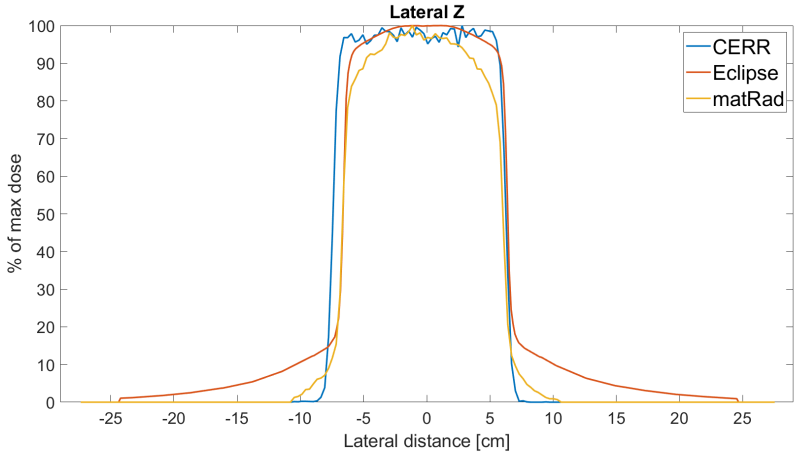
The resulting dose profiles were normalized based on their maximum dose values in each direction, such that the curves could be compared qualitatively. This procedure yielded the graphs displayed in Figure 46 a), b) and c).



(a)



(b)



(c)

Figure 46: (a) Depth dose curve, (b) lateral dose profile in y direction and (c) lateral dose profile in z direction

Whilst the depth dose curves are quite similar, despite the Eclipse curve being much smoother, noticeable differences arise in the lateral dose profile. The lateral spread of the dose in CERR and matRad is vastly underestimated, whereas Eclipse contains a spread out lateral curve with fatter dose tails, probably due to scattering being neglected in CERR. Going a bit more into detail, CERR seems to underestimate the contribution of the scattered photons to the penumbra, whilst producing a similar beam profile in regions where the Compton electrons dominate.

This observation yields more insight into what could happen during the calculation step to create such hot spots. As the leaf positions in opt4D are based on the Dij's in CERR, they also underestimate the lateral spread. Thus, when the calculation is carried out in Eclipse, dose accumulates on the outer regions of the beams, as this dose is not taken into account during the optimization step. Thus, there will be a large discrepancy between Eclipse and opt4D, as there is more dose contribution of neighbouring beams than initially considered.

Whilst the Eclipse based plan performs better regarding both the homogeneity and conformity index, the opt4D plan reaches a comparably close value for the homogeneity index despite the disadvantage of less precisely calculated Dij's and the neglect of influences like leakage during the optimisation step. For the conformity index on the other hand, the Eclipse based plan performs much better, which might be caused by the previously discussed difference in lateral spread due to the neglect of scattering.

To further improve the results, it would make sense to test the performance of plans using Dij's calculated using different research software (e.g. matRad) and find a way to incorporate the fat dose tails into the calculation, such that the dose distribution calculated in opt4D is closer to the actual dose distribution Eclipse produces. Especially for plans with many apertures, this could help to reach more precision and produce better plans.

Regarding the VMAT plans made for the prostate patient, the opt4D based plan performs much better when compared to Eclipse than for the step-and-shoot IMRT plan, despite the disadvantage of the Dij's. For the OARs "Rectum" and "Bladder PTV Ph", the $d_{2\%}$ value was reduced by around 4.5 and 8 Gy respectively compared to the results of the Eclipse plan. Whilst the Eclipse plan still performs better for some OARs, such a large different in $d_{2\%}$ was not observed the other way around.

The curve characteristics calculated still result in better values for both the conformity index and the homogeneity index. However, the conformity index achieves a much lower value for the opt4D VMAT plan, indicating an underdosage in the target rather than depositing more dose in the healthy tissue surrounding the tumour, as was the case for the step-and-shoot IMRT plan made in opt4D. Furthermore, there were no visible hot-spots in the cumulative dose distribution for the VMAT case.

An interesting property of the opt4D VMAT plan is that instead of delivering the dose more or less uniformly, the weight re-optimization seems to lead to the selection of fewer arcs contributing more dose, whilst no dose is delivered for the multiple arcs.

The resulting cumulative dose distribution resembles a dose distribution that would be expected by a step-and-shoot IMRT plan more than the expected result of a VMAT plan. This can be seen on Figure 37.

5.2. Head-and-Neck Patient

Looking at the results obtained by the comparison of a uniformly fractionated plan and a spatio-temporally fractionated plan, optimized based on BED, one may first note that the quantities defining the curve, seen in table 8, are better for the target structure for the STF plan, with differences of up to ∼2.0%. The conformity index is, as for the prostate patient, far from the optimal value of 1. It is again advised to re-calculate the plans using different Dij's to improve this quantity.

The dose quantities for OARs were inspected with respect to the $BED_{2\%}$, as the objective functions were defined using upper bounds instead of minimising the mean dose value. As mentioned in section 3.2, the $BED_{2\%}$ was chosen instead of using the maximum BED value due to it being more robust.

For the BED_2, the STF fractionated plan performs better than the uniformly fractionated plan for a total of 13/17 OARs (∼76%), the uniformly fractionated plan performs better than the STF plan for a total of 4/17 OARs (∼34%).

Interestingly, whilst the structure "Brachial Plexus R" performs better with respect to both $BED_{2\%}$ and mean BED, whilst the objective value is significantly worse. This might be caused by a larger high dose tail, resulting in a larger max dose, containing only a small fraction of the entire volume for both plans.

In general, only the structures "Brain Stem", "Cochlea R", "Salivary Gland L" and "Thyroid" receive a smaller $BED_{2\%}$ in the uniformly fractionated plan, with "Salivary Gland L" still receiving a larger mean dose compared to the STF counterpart.

Regarding the brain stem, the uniform fractionation scheme shows an improvement of 156.3% in terms of $BED_{2\%}$. However, the $BED_{2\%}$ value for the STF plan still falls below the constraint on the BED defined in table 5.

For the STF plan, improvements of more than 10% are found for both "Parotid L Ph" and "Parotid R Ph", with improvements of 12.1% and 18.2% respectively. Whilst the upper BED value constraints in table 5 are still violated, this corresponds to an improvement on achieving the defined optimization constraints.

Summarizing the results in table 9 and 10, for a total of 13/17 OARs, the received $BED_{2\%}$ was reduced, whilst improving both the homogeneity and conformity index for all target structures when using spatio-temporal fractionation as opposed to uniformly fractionated plans.

5.3. Reducing runtime

The runtime of the entire workflow is quite high, spanning over multiple days for spatio-temporally fractionated plans. The largest contributions to this large runtime value are sections 2.2 (Direct aperture optimization) and 2.4 (Aperture Export in Eclipse). Due to the focus of this thesis being on the implemented workflow, the direct aperture optimization runtime was not addressed. The export algorithm on the other hand left some room for further improvements.

The export step requires around 4 minutes to export a single aperture. Thus, the export of a plan with 40+ apertures already takes around 3 hours. Most of the runtime in this step comes from writing the strings to the text-files. The following section will cover some possibilities to improve the runtime of this step of the workflow and reduce it to an acceptable value.

Parallelisation Parallelisation in the sense of multi-threading is not possible in ESAPI. Whereas C# supports multi-threading, Eclipse crashes as soon as a script utilizing it is opened.

However, there is a possible workaround using multiple workstations. STF plans may easily have 40 apertures per instance. This means that during the export step, 40 beams per instance have to be added, calculated and exported. This results in Eclipse experiencing serious frame drops and lags, which slow down the export further.

To circumvent this, a C# script has been implemented which is capable of splitting the *.csv* file obtained during the calculation process in opt4D into multiple *.csv* files based on their instance number. Thus, it is possible to calculate and export apertures of different instances simultaneously by using multiple workstations, which brings the total time down to the runtime of a uniformly fractionated plan with the same amount of apertures at best.

Reducing the runtime complexity of the export algorithm Whilst the approach and algorithm described in section 2.4 provides correct results, the algorithm contains a total of three for-loops, leading to a high runtime complexity. The .NET framework used in WPF applications provides alternatives which can reduce this complexity.

Instead of using three for loops, one can "cut out" one column of the dose plane, and transform it into a string directly. Thus, one entire loop can be saved. This speeds up the computation from about 4 minutes per aperture (depending on the writing method used and the positioning of the writer in the for loop) to about 30 seconds.

In general, two commands were modified to increase the runtime:

- Instead of looping through all points of a 1D slice of the dose matrix, one can use *Enumerable.Range(0,plan.Dose.YSize).Select(r => doseMatrix[x,r]).ToArray()* where

x denotes the row of the 2D dose matrix slice discussed in section 2.4. This returns an array containing all values within the selected slice. Then, the command *String.Join(" ", slice)* can be used to convert the array into a string, where each entry is separated by a white-space.

- Whilst it is common to use a *StreamWriter* for these tasks, utilising a *StringBuilder* for building the entire dose plane string, then writing the dose plane using *File.AppendAllText* performs much faster.

6. Conclusion

The workflow developed during this thesis has shown to be capable of producing plans of acceptable quality within Eclipse, despite the differences in accuracy of the Dij's used for the calculation. This was validated for both step-and-shoot IMRT plans, as well as VMAT plans. Furthermore, the workflow was able to upload plans optimized on BED to Eclipse.

Regarding the H&N tumour patient, spatio-temporal fractionation showed improvements on a total of 13/17 OARs, whilst still performing better regarding homogeneity and conformity indices of the target BED curves.

Possible next steps would include incorporating the Eclipse leaf motion constraints into opt4D, such that the MLC positions calculated during the direct aperture optimization (section 2.2) can be uploaded to Eclipse without setting them to a maximum threshold value. Furthermore, improving the Dij calculation, especially for the dose tails, could also help to bring the plan for the prostate patient closer to the result obtained by Eclipse. Doing this might yield further insight into the potential of spatio-temporally fractionated treatments.

7. Acknowledgements

I would like to thank my supervisors prof. Jan Unkelbach and Nathan Torelli as well as the rest of the USZ radiation oncology group for their continuous help and guidance during my thesis.

Besides writing a thesis, studying physics also comes with many exam periods. I am grateful that I did not have to manage to do all of this on my own, and want to thank Mischa Stifter, Ron Cohn-Wagner, Janic Weber, Lars Widmer and Amadeo Nalbantis, with whom I have shared many days studying and laughing during our time at the University of Zurich.

Finally, I would like to thank my parents, friends, and girlfriend, for providing continuous support during all stages of my studies, especially during the more stressful periods.

8. References

1. Halperin, E. C., Wazer, D. E., Perez, C. A., & Brady, L. W. (2018). Perez & Brady's Principles and Practice of Radiation Oncology. Wolters Kluwer.
2. Unkelbach J, Papp D, Gaddy MR, Andratschke N, Hong T, Guckenberger M. Spatiotemporal fractionation schemes for liver stereotactic body radiotherapy. Radiother Oncol. 2017 Nov;125(2):357-364
3. Unkelbach J, Bussière MR, Chapman PH, Loeffler JS, Shih HA. Spatiotemporal Fractionation Schemes for Irradiating Large Cerebral Arteriovenous Malformations. Int J Radiat Oncol Biol Phys. 2016 Jul 1;95(3):1067-1074. doi: 10.1016/j.ijrobp.2016.02.001. Epub 2016 Feb 6. PMID: 27113566; PMCID: PMC4909567.
4. Gaddy MR, Yıldız S, Unkelbach J, Papp D. Optimization of spatiotemporally fractionated radiotherapy treatments with bounds on the achievable benefit. Phys Med Biol. 2018 Jan 5;63(1):015036. doi: 10.1088/1361-6560/aa9975. PMID: 29303116.
5. Varian Medical Systems, Inc., TrueBeam Technical Reference Guide - Volume 1, September 2021.
6. Varian Medical Systems, Inc., Eclipse Photon and Electron Reference Guide, Jan. 2016.
7. Petrova D, Smickovska S, Lazarevska E. Conformity Index and Homogeneity Index of the Postoperative Whole Breast Radiotherapy. Open Access Maced J Med Sci. 2017 Sep 17;5(6):736-739. doi: 10.3889/oamjms.2017.161. PMID: 29123573; PMCID: PMC5672112.
8. "The International Commission on Radiation Units and measurements," Journal of the ICRU, vol. 10, no. 1, 2010.
9. https://github.com/brjdenis/VarianESAPI-EQD2Converter
10. https://github.com/ahmedeltaher/MVVM-Kotlin-Android-Architecture

A. Coordinate Systems

The technical information for the following sections was taken from Varian's Eclipse Photon and Electron Reference Guide [6].

A.1. Eclipse

Eclipse contains two different coordinate systems.

Properties and Methods working with distances or positions are measured in mm. 3D Positions are returned with respect to the DICOM coordinate system. These coordinates however differ from the visualisation observable in the Eclipse User Interface, which is measure in cm and corresponds to the Standard coordinate system.

A.1.1. Standard Coordinate System

By default, the standard planning coordinate system is the standard coordinate system.

- X-axis: points to the right when facing the gantry
- Y-axis: points towards the floor
- Z-axis: points towards the gantry

The orientation can also be seen in Figure 47.

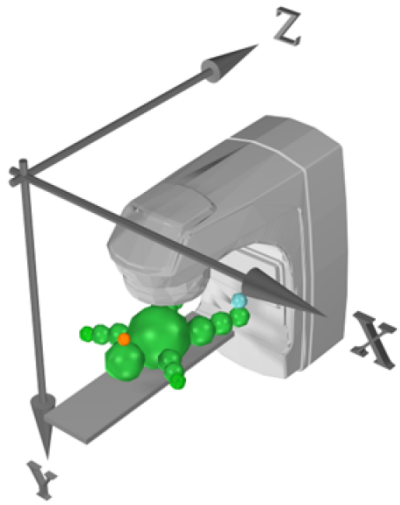


Figure 47: Standard Coordinate System

The orientation of the DICOM patient does not effect the standard coordinate system. The unit of measurement is cm.

A.1.2. DICOM Coordinate system

The DICOM coordinate system is based on three axis. In relation to the patient, the following holds:

- X-axis shoulder-to-shoulder axis
- Y-axis front-back axis
- Z-axis feet-head axis

The orientation can also be seen in Figure 48.

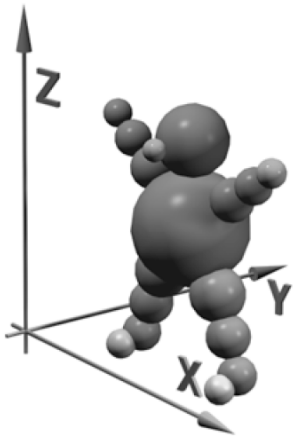


Figure 48: DICOM Coordinate System

The unit of measurement is mm.

A.1.3. Calculation Grid AAA-Algorithm

Inspecting exported plans from Eclipse shows that the dose grid does not correspond to the CT-grid dimensionality-wise when using AAA. The resulting matrix is smaller in size.

The size of the resulting matrix depends on the jaw positions. Furthermore, a margin is added, used to calculate the low dose tail due to head scatter and phantom scatter. The margin itself depends on the field size and the desired calculation resolution. Per default, the margin corresponds to a size of 12cm. This value might change, as the margin is set such that the total number of calculation points at the isocentre plane does not exceed 74'000. The formula is given here

$$\frac{(\mathrm{FS}_X + 2 * M) \times (\mathrm{FS}_Y + 2 * M)}{h^2} \leq 74000 \tag{A.1.1}$$

- FS_X = Field size in cm in X-direction
- FS_Y = Field size in cm in Y-direction
- M = Size of the additional margin in cm (always at least 7cm)
- h = Calculation grid size in cm

Thus, the margin M is a value within the range of 7-12cm beyond the field edge at the isocentre plane.

On a side-note, when using Acuros XB, the process is similar, but the phantom scatter may extend further due to the calculation being based on Cartesian coordinates.

In the axis perpendicular to the image slices, AAA adapts the grid resolution such that the dose is calculated exactly on the image slices. Should the slice thickness be larger than the defined grid size, dose calculation may be skipped in some slices.

Let slice thickness be denoted by IS. Then, AAA calculates

$$\mathrm{IS}/n ... \mathrm{IS}/3, \mathrm{IS}/2, \mathrm{IS}, \mathrm{IS} * 2, .., \mathrm{IS} * n \tag{A.1.2}$$

and sets the dose plane spacing to the value which is closest to the defined grid size.

If the slice separation is larger than the grid size, AAA will produce internal images with a smaller separation. If the slice thickness is divided by an even number, one of the extra slices is located in-between the middle of the two original slices, meaning that possible sharp boundaries in the original image become blurred, as the extra slice uses intermediate density.

A.1.4. Summary

To summarize, Eclipse contains the following coordinate systems:

	Standard	**DICOM**
x	Right when facing Gantry	Shoulder-to-Shoulder
y	Towards the floor	Front-to-Back
z	Towards Gantry	Feet-to-Head
Origin	Middle point	Bottom-left point

Table 16: Summary Standard/DICOM coordinate System in Eclipse

Then, within the DICOM coordinates, one can make a further distinction between dose and CT coordinates:

CERR uses the same axis orientation as DICOM, with the change that in CERR, the y-/z-axis are inverted. Thus, when CERR coordinates are imported into Eclipse (e.g.

	CT	Dose
Grid	Based on CT resolution	Based on Dose calculation resolution
Dimensions	Based on CT Scan	Based on Jaw positions and margin
Origin	Bottom-Left of CT matrix	Bottom-Left of Dose matrix

Table 17: Summary CT/Dose coordinate System in Eclipse

isocentre positions), it is important to multiply them by -1. In comparison with DICOM, this yields

	DICOM	CERR
x	Shoulder-to-Shoulder	Shoulder-to-Shoulder
y	Front-to-Back	Back-to-Front
z	Feet-to-Head	Head-to-Feet

Table 18: Summary DICOM/CERR coordinate Systems

Another bit of important information is the way in which Matlab stores matrices. The CERR x axis goes shoulder-to-shoulder, and y goes Back-to-front. As the first matrix index in Matlab always corresponds to the rows, the y axis corresponds to the first dimension of the matrix when viewing the patient in CERR, Head-First-Supine.

Thus, upon inspecting the dose matrix using the global *planC* variable, one ends up with an orientation of:

$$\text{size(doseMatrix)} = (\text{dim(y)}, \text{dim(x)}, \text{dim(z)}) \tag{A.1.3}$$