**Dissertation**
**submitted to the**
**Combined Faculties for the Natural Sciences and for Mathematics**
**of the Ruperto-Carola University of Heidelberg, Germany**
**for the degree of**
**Doctor of Natural Sciences**

**Put forward by**


| | |
|---|---|
| **Diplom-Physiker:** | **Peter Ziegenhein** |
| **Born in:** | **Gera, Germany** |


**Oral examination: July 24, 2013**

# Realizing a new paradigm in radiation therapy treatment planning

**Referees:** **Prof. Dr. Uwe Oelfke**
**Prof. Dr. Wolfgang Schlegel**

## Umsetzung eines neuen Paradigmas zur Erzeugung von Strahlentherapieplänen
## Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der sogenannten Interactive Dose Shaping (IDS) Methode. Das IDS Modell gibt dem Planer die Möglichkeit direkt, über eine graphische Benutzerschnittstelle, Merkmale des Therapieplans zu verändern. Im Gegensatz zur konventionellen Planungsmethode ist hierfür keine iterative Optimierung oder die Definition einer Zielfunktion notwendig.

Im ersten Teil dieser Arbeit wird die Umsetzung der konventionellen Planungsmethode auf einem modernen Rechner untersucht. Es werden Konzepte für eine Implementierung beschrieben, die eine nahezu optimale Laufzeit erreichen. Der zweite Teil dieser Arbeit stellt die IDS Methodik vor. Das IDS Konzept überwindet einige wesentliche Nachteile der konventionellen Optimierung bei gleichzeitiger, effizienter Nutzung moderner Hochleistungsrechner durch interaktive Strategien zur Therapieplanerzeugung. Die Umsetzung des IDS Konzepts besteht aus drei Teilen: (1)Einer zwei-Schritt Dosis Variation and Recovery Strategie, die in der Lage ist lokale Merkmale des Plans zu verändern und die unvermeidlich korrelierten Änderungen im Plan zu kompensieren. (2)Einer neuen Methode zur Dosisberechnung. (3)Einer IDS Planungsumgebung, die ein umfassendes graphisches Interface zur Verfügung stellt.

Es konnte gezeigt werden, dass mit der vorgestellten Interactive Dose Shaping Methode konventionell erstellte Pläne reproduziert werden konnten. Des Weiteren ist es gelungen das IDS Konzept so zu implementieren, dass dieses in Echtzeit auf modernen Computern ausgeführt werden kann.



## Realizing a new paradigm in radiation therapy treatment planning
## Abstract

This thesis investigates the feasibility of a new IMRT planning paradigm called Interactive Dose Shaping (IDS). The IDS paradigm enables the therapist to directly impose local dose features into the therapy plan. In contrast to the conventional IMRT planning approach, IDS does not employ an objective function to drive an iterative optimization procedure.

In the first part of this work, the conventional IMRT plan optimization method is investigated. Concepts for a near-optimal implementation of the planning problem are provided. The second part of this work introduces the IDS concept. It is designed to overcome clinical drawbacks of the conventional method on the one hand and to provide interactive planning strategies which exploit the full potential of modern high-performance computer hardware on the other hand. The realization of the IDS concept consists of three main parts. (1)A two-step Dose Variation and Recovery (DVR) strategy which imposes localized plan features and recovers for unintentional plan modifications elsewhere. (2)A new dose calculation method (3)The design of an IDS planning framework which provides a powerful graphical user interface.

It could be shown that the IDS paradigm is able to reproduce conventionally optimized therapy plans and that the IDS concepts can be realized in real-time.

# Contents

# 1. Introduction

It is well recognized that scientific computing became the third pillar of science, standing right next to experiments and theoretical analysis. The way scientific discoveries are made changed dramatically with the availability of inexpensive and powerful computational resources in the new century. Computing becomes a crucial tool for testing complex models or simulating experiments which cannot be done in reality.

The process of generating a clinical acceptable plan for the radiation therapy of cancer is a typical example for the use of computers in radiation oncology. The delivery of ionizing radiation must be precisely simulated and optimized in order to eradicate the disease while minimizing side effects. Fundamentally, this is a problem of computation. The quality of practical solutions to the therapy planning problem arises from the performance of state-of-the-art computer technology. Recently, powerful image processing technologies have been used to monitor the patient geometry during the course of treatment. Based on these images it is possible to individualize the treatment plan to the patient. In order to accomplish that a huge amount of computations must be carried out which has become feasible today.

Although, a lot of technical advancements in radiation oncology have been achieved, only little effort was made to improve the concept of how intensity modulated treatment plans are derived. The traditional method which is used still today in most clinics and research facilities has been proposed more than 20 years ago. The conventional planning algorithm has been ported to massively parallel computing architectures (GPUs) but the method itself did not change. In contrast to that, computers have changed a lot in the last two decades and the question arises if there is another method to generate a radiation treatment plan which on the one hand became just feasible on todays high performance computers and on the other hand adapts to the recent developments in radiation therapy.
In this thesis such an alternative planning method which relies on the performance potential of modern computers is presented. This is an interdisciplinary work, which combines high performance computing and medical physics. A deep knowledge of both research fields is necessary to succeed in developing a new planning paradigm. Therefore, both aspects will be discussed in this thesis.

## 1.1. Objectives of this work

Conventional methods of generating IMRT plans are based on the optimization of a single or multi-criteria objective function that specifies the desired dose to the patient. Although, this method had been used successfully in clinical application for over 20 years, it suffers from several inevitable drawbacks which create a conflict to future developments in therapy planning. First, the objective function contains a set of dose volume constraints for pre-segmented, organ specific volumes of interest. Generally not all objectives can be fulfilled at the same time, so that the optimization is driven by penalty factors which assign a relative importance to each dose constraint. This denotes a trade-off between the clinical goals which is not intuitive to the planner. In addition to that, the optimization is an automated process which does not allow any interactivity with the therapist during the plan generation. The transfer of clinical knowledge from the clinician to the planning algorithm is limited by the mathematical specification of the objective function. Second, local dose features can only be controlled on the level of pre-segmented volumes. A prescription of plan features to sub-organ structures is difficult and requires the introduction of artificial organs. Third, it is difficult to adapt already optimized plans to a new patient geometry which is detected by modern image guidance methods. The plan optimization is always a

global operation to the whole plan. An attempt for a local adaptation can eradicate already established, desirable features in the plan elsewhere. In addition to that, the algorithms used for IMRT planning have not been changed significantly in the last two decades, but personal computers on which the optimization procedure is taken out have changed dramatically during this time.

The aims of this thesis are therefor, to introduce a new concept for the generation of a three-dimensional radiation therapy plan to overcome the clinical drawbacks of the conventional optimization. The new planning paradigm changes the way a plan is retrieved. It should provide a set of interactive tools which enables the therapist to specify a series of local features to the plan. The new planning method should not use a conventional optimization or the definition of an objective function. This is particular interesting for the application in the field of adaptive radiation therapy (ART). It is highly desirable that the interactivity between the therapist and the plan generation algorithm operates on a real-time scale. Local planning features should be imposed by the therapists in the order of one second as part of the planning process. In order to realize the real-time aspect of the planning algorithm it is necessary to exploit the capabilities of modern computational hardware. Thus, the new paradigm must be designed by having both, the clinical planning aspect and the computational feasibility of that aspect in mind. In order to achieve the computational feasibility, the algorithm of the conventional optimization method must be analyzed and improved mechanism must be concluded.
The goal of this thesis is to demonstrate the feasibility of this new planning paradigm. It has to be shown that the concepts which implement the new paradigm are fast enough on modern computational hardware and at the same time clinical effective to allow the generation of a acceptable plan. The assessment of the new concept requires the development of a planning environment which consists of three main parts: First, a set of strategies of how the interactive planning approach is realized. Second, a new dose calculation method which is designed to operate very efficiently together with the planning strategies and third, a planning framework which provides the graphical interface. The strategies for the interactive planning and the new dose calculation method will be discussed in detail in this work. The implementation of the planning framework will only be discussed briefly since this is the objective to another dissertation.

The material in this thesis is organized as follows:

- Chapter 2 summarizes the basic concept of intensity-modulated radiation therapy (IMRT) and introduces a numerical solution to the inverse planning problem employing an iterative Newton optimization method.
- Chapter 3 presents one physical method to calculate dose in the patient according to Bortfeld et al. (1993). This method is adapted in chapter 5 for the new planning paradigm.
- In chapter 4, the conventional IMRT plan generation module using Newton's method is discussed from a computational point of view. Key concepts for an optimal realization of the planning problem will be provided for shared-memory systems and distributed systems. The chapter ends with a discussion of the performance results and gives an outlook on the future of therapy planning from a computational point of view.
- Chapter 5 introduces the realization of the new planning paradigm - Interactive Dose Shaping - for three-dimensional clinical therapy planning. The three main components of the new paradigm - the Dose Variation and Recovery strategy, a new method for implementing small modifications of the radiation fluence into the plan and the design of a new planning framework - will be introduced from a clinical and computational perspective.
- Finally a summary of this work and an outlook are given in the fourth chapter.

# 2. Intensity Modulated Radiation Therapy

The introduction of intensity-modulated radiation therapy (IMRT) has been considered as one of the most interesting development in radiation oncology in the last three decades (Bentzen 2005, Nutting 2003). It took more than 80 years to come from a clinical practice using uniform intensity beams to the idea of applying intensity modulated beams. One of the key enabling feature for the development of IMRT was the clinical introduction of volumetric imaging techniques such as computed tomography (CT). With the help of a CT image, the radiation oncologist was able to determine the geometry of complex target volumes and surrounding healthy tissue which called for a more precise method the deliver dose to patient.
The basic idea of IMRT is to modulate the intensity of the incoming beams of radiation (in contrast to conventional therapy with uniform intensity beams). The modulation allows a higher degree of agreement (conformity) of the resulting dose pattern with the target volume (Bortfeld 2006).One would apply a higher intensity to areas of the beam which mostly affect target volumes and a lower intensity to areas of the beam also affecting healthy tissue. An example comparing the conventional treatment and IMRT for a complex tumor geometry is shown in figure 2.1

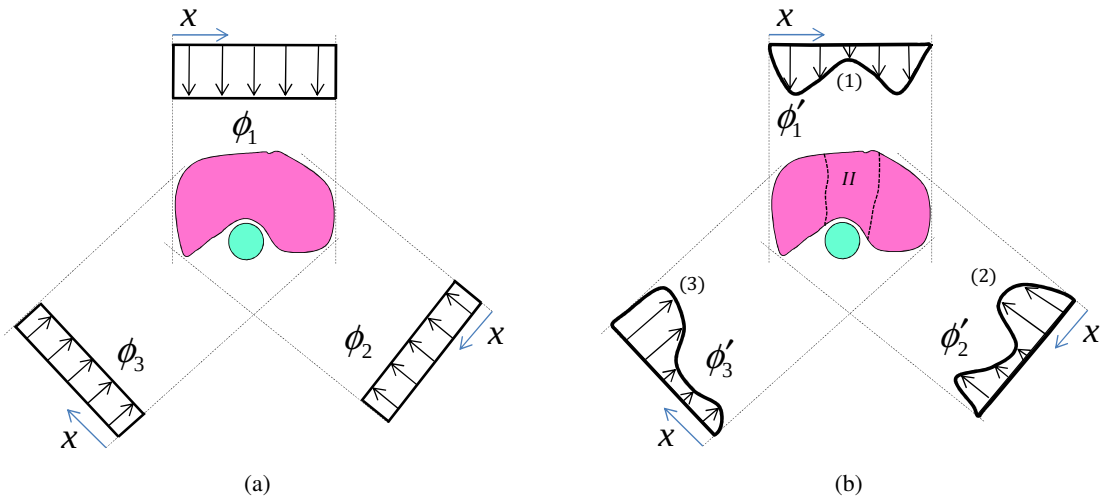



**Figure 2.1.:** A transversal slice through a complex tumor planning scenario irradiated with three beams. A critical healthy tissue volume (green) is located closely to a tumor volume (red) which is treated using a) conventional beams and b) IMRT .

Physical dose is delivered to the patient along the path of the incident beam (extrapolation of the black arrows). The resulting dose distribution (dose pattern) in the patient is simply the linear superposition of the physical dose contribution from all incident beams. The planning scenario using conventional treatment applies a constant fluence which is localized between the projection of the tumor volume onto the beam (dotted lines). Thus, the whole tumor area and the healthy tissue is equally irradiated for that geometry. The intensity may vary between different beams but is limited by the maximum tolerable dose of the healthy tissue volume.
The intensity-modulated planning setup in figure 2.1(b) relies on the assumption, that not all parts of the tumor need to be irradiated by one beam equally. The intensity modulation is derived in a way, so that the resulting dose pattern in the patient shows a high conformity to the target volume only. This is possible by exploiting the *synergistic effect* of applying beams from multiple directions. For instance $\phi_1$ shows an intensity valley (1) in

the locality which is projected to the healthy tissue volume resulting in an under-dosage of part II in the tumor. This under-doseage can be compensated by $\phi_2$ and $\phi_3$ boosting the intensities in areas (2) and (3), respectively. This results in a satisfying coverage of tumor part II while keeping the radiation exposure of the healthy tissue to a minimum.
For conventional treatment, the planner can derive a beam setup containing the quantity, the angle and intensity of each beam manually by a trial-and-error approach using his clinical experience. This technique is called forward planning, since the planner suggests a beam setup, evaluates the resulting dose distribution in the patient and adapts the beam setup accordingly. For complex IMRT planning scenarios, deriving an acceptable modulation of the beam fluencies is a complicated endeavor and needs to be solved by an automated strategy using a computer program. Here, the therapist prescribes a desired dose pattern to the contoured organs of the patient and lets the computer program find a modulation of the radiation fluences that implement the prescription at best. This strategy formulates an *inverse planning problem* (Brahme et al. 1982) which is one approximate solution to generate a clinical radiation therapy plan to treat cancer.

## 2.1. Numerical methods for solving the inverse planning problem in IMRT

The inverse problem has no exact solution. A therapist would prescribe a high dose to the tumor while the surrounding healthy tissue should receive no dose at all. This is physically not possible due to the nature of radiation. The goal in therapy planning is to find a physical fluence modulation that realizes a therapy plan which is as close as possible to the ideal dose prescription. The first who formulated IMRT planning as an optimization problem was Webb (1989). Webb used the simulated annealing method to minimize a cost function in order to find the optimal therapy plan. Nowadays, the inverse problem for IMRT is commonly solved numerically using a quasi-Newton optimization algorithm based on the low memory BFGS method (Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970) on a computer planning system. In order to do that, a discretization of the IMRT therapy planning scenario must be defined:

$$\boldsymbol{r} \rightarrow i \tag{2.1}$$

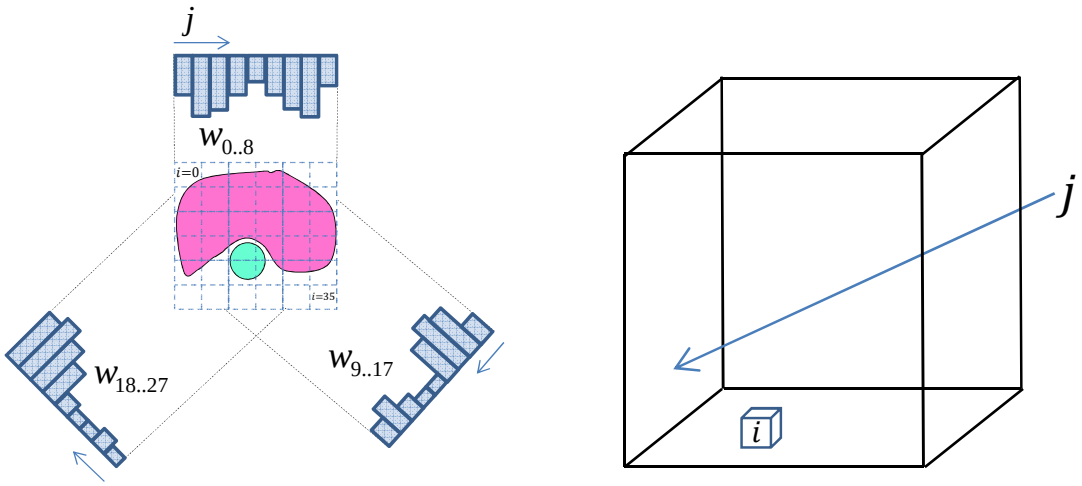
$$\phi_k(x) \rightarrow w_j, j \in B$$

The patient volume is discretized into a number of equally sized voxel which are numbered consecutively and given an index $i$. The modulation of the beam fluences is described by a set of smaller sub-beams called bixel each having an individual but constant intensity (weight) $w_j$. Using these discretizations, the transversal plane of the complex tumor case introduced in figure 2.1(b) would be described as shown in figure 2.2(a)
The physics of radiation dose deposition is encapsulated in the *dose influence matrix* $D_{ij}$ (Censor et al. 1988). The element found in the i-th row at the j-th column describes the portion of dose which is delivered by the j-th bixel to the i-th voxel in the patient (see figure 2.2(b)). The physical dose $d_i$ in the i-th voxel is then derived as the superposition of all dose contributions over all bixel in the beam setup $B$:

$$d_i = \sum_{j \in B} D_{ij} w_j, \tag{2.2}$$

The dose influence matrix is only sparsely populated, since one voxel is generally not influenced by all bixel. A typical clinical IMRT plan comprises in the order of $10^6$ voxel and $10^4$ bixel.

The inverse planning problem tries to solve the following question: 'Given a fixed beam setup $B$, which set of intensity values $w_j, j \in B$ realizes a certain prescribed dose distribution $d^{pres}$'. Furthermore, all intensity values $w_j$ have to be positive. A negative intensity would correspond to a dose erasing bixel which is physically impossible. The prescribed dose distribution is defined as dose volume histogram constraints for pre-segmented, organ specific volumes of interest. For example, in the segmented target volume the therapist usually defines a minimum required dose $d_t^{min}$ to irradicate the disease and a maximum dose $d_t^{max}$ to limit radiation side effects.




(a) (b)

**Figure 2.2.:** a) The discretization of the IMRT planning problem at the example of a transversal slice through a complex tumor planning scenario irradiated with three beams. b) Describing the physics of dose using the concept of the dose influence matrix $D_{ij}$.

For segmented radio-sensitive healthy tissue (Organs At Risk (OAR)) which should be spared from dose as much as possible only a maximal tolerable dose $d_r^{max}$ is defined. Due to the physical limitations in the radiation setup, the ideal dose distribution as prescribed by the therapist would not be possible to realize in general. For instance, the ideal dose distribution for the planning case drawn in figure 2.2(a) would be to prescribe a homogeneous high dose escalation to the tumor volume while setting the maximum dose to the OAR in the emargination to zero. No physically possible radiation beam modulation could realize that. In order to control these trade off scenario, the therapist defines the importance of a dose constraint by assigning a penalty factor to it. Given this information, the quality of a treatment plan is measured by an objective function $F = \sum_i f_i$ while $f_i$ defines the weighted distance between the actual dose $d_i$ an the prescribed dose in voxel $i$:

$$f_i(d_i) = \begin{cases} s_u^t[d_t^{min} - d_i]_{\geq 0}^2 + s_o^t[d_i - d_t^{max}]_{\geq 0}^2 & i \in Target \\ s_o^r[d_i - d_r^{max}]_{\geq 0}^2 & else \end{cases} \tag{2.3}$$

while $s_o$ and $s_u$ denote the penalties for over- and under-doseage, respectively. The operator $[\cdot]_{\geq 0}$ ensures that only positive terms are considered. With the objective function $F$ the optimzation problem is defined as:

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, F(\mathbf{w}), \tag{2.4}$$

The basic work flow to solve this problem by an iterative Newton method is shown in figure 2.3 First, an initial set of bixel weights $w^0$ is guessed. Usually all weights are set to a constant value (no modulation). Then, a new set of bixel weights is found by navigating through the solution space in direction $p$ with a step length of $\alpha$. The quality of the new (and potentially better) set of bixel weights $w^{x+1}$ is then tested by a physical dose calculation (back-projection). The resulting dose distribution in the patient is then evaluated by the objective function $F$. The stopping criterion checks whether the improvement on the objective function compared to the previous iteration is sufficient to continue the optimization process. If not, one can assume that a stationary point is found and the optimization ends. Conventionally, an improvement of 0.1% is required to continue the optimization loop.

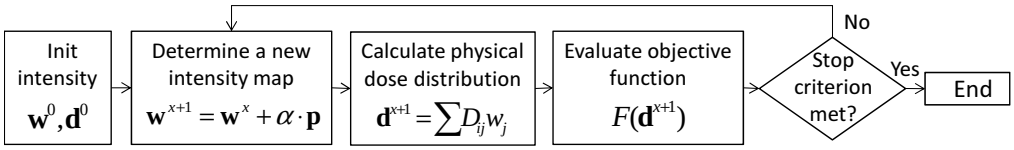



**Figure 2.3.:** Basic workflow of an IMRT treatment plan optimization algorithm based on Newton's optimization method.

There are different methods of how to find $\mathbf{p}$ and $\alpha$. The most accurate (and also the most time-consuming) method given the objective function 2.3 is to derive:

$$\mathbf{p} = F''(\mathbf{w})^{-1} F'(\mathbf{w}) \tag{2.5}$$

This is the coventional Newton method which assumes that $F$ is well approximated by its second order Taylor expansion. In fact, using a quadratic objective function like in 2.3 the second order Taylor expansion is equal to $F$ itself. Thus, the optimization process would find the optimal bixel weight in only one step using the direction $\mathbf{p}$ in equation 2.5 and a step length of $\alpha = 1$. While the gradient can be calculated efficiently by:

$$\nabla F_j(\mathbf{w}^x) = \sum_i \frac{df_i}{dw_j} = \sum_i 2 s_i \left[ d_i - d_i^{pres} \right] D_{ij} \tag{2.6}$$

the second derivative of $F$ would make a huge matrix having a rank in the dimension of the number of bixel used in the planning setup which makes it is infeasible to calculate the new direction according to equation 2.5.

Nill (2001) proposed a modified gradient approach to implement a feasible solution to the inverse planning problem. The second Hessian matrix of $F$ is approximated by it diagonal elements:

$$\frac{d^2 F}{dw_j^2} = 2 \sum_k s_k D_{kj}^2 \tag{2.7}$$

so that the projection operation of finding a new set of bixel weights can be formulated as

$$w_j^{x+1} = \left[ w_j^x - \alpha \frac{\sum_i 2 s_i [d_i - d_i^{pres}] D_{ij}}{2 \sum_i s_i D_{ij}^2} \right]_+ \tag{2.8}$$

The voxel specific penalty factor $s_i$ depends on the dose in voxel $i$. For $d_i < d_i^{pres}$ the factor penalizes a under-dosesage ($s_i = s_i^u$), for $d_i > d_i^{pres}$ an over-dosesage has to be considered ($s_i = s_i^o$)
Nill (2001) implemented his treatment planning system in a modular fashion, which separates the process of the dose calculation from the optimization loop. The matrix $D_{ij}$ which holds the physics of radiation transport is calculated once before the optimization loop in figure 2.3 starts. The dose distribution in the patient (back-projection) and the new direction $\mathbf{p}$ can then easily be calculated by performing simple matrix multiplication operations according to equation 2.2 and 2.8. This strict separation makes it possible to generate this matrix using another third party algorithm on another computer system. Even more, the optimization loop does not have to know anything specific about the process of creation, e.g. about the modality of radiation or which algorithm was used.

Bangert (2011) proposed another quasi-Newton solution based in the L-BFGS method in combination with Armijo's rule for a backtracking line search to determine the step length. The construction of the second derivative and its inversion is bypassed by using a two loop recursion algorithm (Nocedal and Wright 1999) which directly approximates the search direction **p**. A new set of bixel weights is only accepted if the improvements on the objective function satisfies Armijo's rule:

$$F(\mathbf{w}^{x+1}) \leq F(\mathbf{w}^{x}) - \alpha c p^{x\top} \nabla F(\mathbf{w}^{x}) \tag{2.9}$$

If not satisfied, the step length is reduced iteratively. It can be shown that one can always find a sufficiently small value for $\alpha$ which eventually satisfies Armijo's rule. The expected step length $\alpha$ given a quadratic objective function $F(\mathbf{w}^{x})$ is 1.0. However, in the first iterations when the L-BFGS approximation has not reached a sufficient quality, it is desirable to apply a smaller step length.

The method of solving the inverse problem using the Newton approach has been widely used in the last decade. For modern application of IMRT (e.g. in the field of adaptive radiation therapy) or for large scale problems (e.g. for beam orientation selection) a fast implementation of the IMRT planning solution on modern computational hardware is necessary. In chapter 4 the computational characteristic of IMRT planning is discussed and an optimal strategy for its implementation on CPU-based systems is proposed.
Even with an ultra-fast solution of the inverse problem using Newton's method, this approach has several inevitable drawbacks which are discussed in section 4.7 and 5.1. In an attempt to overcome these drawbacks an alternative interactive approach called Interactive Dose Shaping is introduced in chpater 5.

# 3. Fast 3D Dose Calculation based on the Decomposition of Pencil Beam Kernels

The concept of intensity modulated radiation therapy can only be fully exploited, if there exists a fast method for calculating accurate dose distributions for irregular shaped radiation fields. In the early days of IMRT most calculation algorithms did not satisfy this requirement. They were based on measured dose profiles for regular (e.g. circular or quadratic) shaped fields. The calculation of irregular fields using these *conventional* methods is only a crude approximation (Bortfeld et al. 1993). Accurate dose calculation methods of that time used a pencil beam convolution technique to properly account for scattered particles. The convolution was carried out between the primary fluence and a *kernel* which described the dose deposition of a narrow beam at a series of depth. Although, the accuracy of these convolution techniques is much higher, the long calculation times prevented a wide use in clinical practice.
Bortfeld et al. (1993) proposed a fast convolution technique for accurate dose calculation of irregular shaped radiation fields 20 years ago. Since then it has been one of the most popular algorithms used for IMRT planning systems. Nowadays, it is mostly replaced in research and clinical practice by more accurate Monte Carlo methods or superposition algorithms. However, the concept is very fast on modern computers. An adapted version of the convolution technique is used for the realization of the Interactive Dose Shaping method presented in chapter 5. For that reason, this section will present the basic concept of the dose calculation based on the decomposition of pencil beam kernels.

The approach for calculating accurate dose of a irregular shaped radiation field according to Bortfeld et al. (1993) is:

$$D'_{irreg}(x_p, y_p, d) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Psi'(x,y)F(x,y) \times K(x-x_p, y-y_p, d)dxdy \tag{3.1}$$

The equation describes the dose $D'_{irreg}$ at one point in the patient which is off the central axis by $(x_p, y_p)$ and situated at a radiological depth $d$ (see sidebar coordinate systems on page 20). The calculation can be recognized as a two-dimensional convolution between the a pencil beam kernel $K$ and the product of the primary energy fluence $\Psi'(x,y)$ and the transmission factor $F(x,y)$. The kernel $K(x-x_p, y-y_p, d)$ describes the dose contribution to the central axis of an infinitely narrow pencil beam incidenting at $(x_p, y_p)$. It is assumed that the energy spectrum of the beam is independent from the lateral position $(x,y)$, so that the kernel can be considered lateral invariant. This assumption allows for a more intuitive understanding of the kernel: it is the dose contribution to the point $(x_p, y_p)$ of a pencil beam incidenting the plane $d$ at $(x,y)$. The transmission function $F(x,y)$ describes the shape of the radiation field. The values range from $F(x,y) = 0$ (no transmission at point $(x,y)$) to 1 for full transmission. Every possible field modulation generated by blocks, wedges or MLCs can be realized through a transmission function $F$. Figure 3.1 illustrates the notion of kernel and transmission function. The figure shows one radiation field with an arbitrary shape $F(x,y)$. The transmission value within the shape is 1, outside of the shape the value is set to 0. The origin of the coordinate system divides the radiation field into 4 equally sized quadrants. A circular pencil beam kernel is shown which propagates through the field at point $(x_p, y_p)$ and deposits primary dose within the shape of the beam. The dose deposition depends on the depth of penetration into the patient, thus it is a three-dimensional kernel. In the example shown in figure 3.1 the kernel gets larger for a plan which lies deeper into the patient volume. Thus, the convolution in equation 3.1 has to be carried out for all depth of interest.

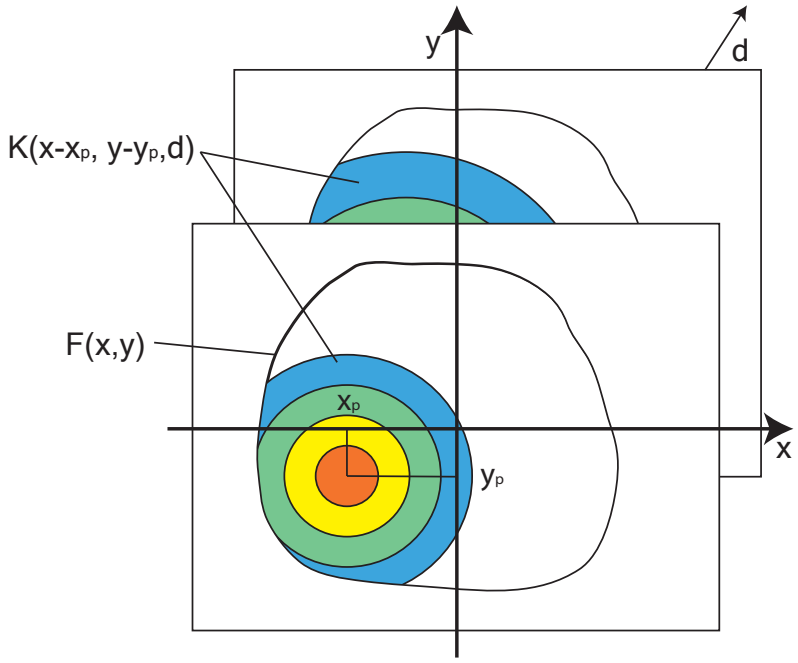



**Figure 3.1.:** Illustration of the pencil beam convolution technique on the example of an irregularly shaped field. The pencil beam kernel depends on the depth of interest.

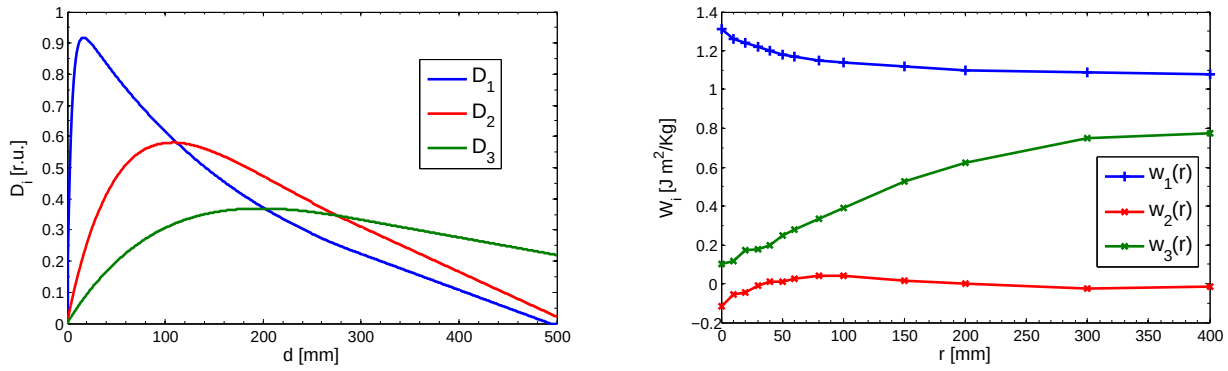
Performing a two dimensional convolution for ever depth is very time consuming even on modern computational hardware. Bortfeld et al. (1993) propose a decomposition method to separate the kernel into two parts, one which only depends on the lateral displacement $x$ and $y$ (or $r$, circular coordinates will be used in the following) and another one which only depends on the depth $d$. Using the theory of the singular value decomposition, Bortfeld found out that the kernel can be decomposed into a sum of three terms:

$$K(r,d) \approx \sum_{i=1}^{3} w_i(r) D_i^{'}(d) \tag{3.2}$$

The use of more terms did not lead to any significant improvements. The depth dose components each have a physical meaning. $D_1$ can be considered as the depth dose component on the central axis. $D_2$ describes the scattered radiation which comes from small fields and $D_3$ is the contribution due to scattered radiation coming from large fields. These components can be determined using a Monte Carlo method which is described in (Mackie et al. 1985). The weights $w_i(r)$ are a result of a least square fit to the central axis dose and a subsequent deviation. The final equation for calculating accurate dose for irregular shaped fields which includes the kernel decomposition method can be derived to:

$$D_{irreg}^{'}(x_p, y_p, d) = \sum_{i=1}^{3} D_i^{'}(d) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Psi^{'}(x,y) F(x,y) \times w_i(x - x_p, y - y_p) dx dy \tag{3.3}$$

With the help of the kernel decomposition the two-dimensional integral no longer depends on the depth of the radiation field. For the plan delivering hardware used for research purposes on the German Cancer Research Center, the $D_i$ and $w_i$ data sets are shown in figure 3.2.
For implementing the dose calculating method, a practical discretization of the convolution problem has to be found. Bortfeld et al. (1993) calculates the weighting terms $w_i$ on a lattice of $128x128$ points over the whole radiation field. The convolution is carried out in the frequency domain using the fast Hartley transform (FHT).




(a) (b)

**Figure 3.2.:** Components of the singular value decomposition of the pencil beam kernel a) The depth components $D_i(d)$ b) The kernel weight functions $w_i(r)$.

The FHT method is chosen because its transform is purely real and the complex arithmetic used for instance in the FFT kernel function is not considered. To calculate the dose distribution for the whole patient volume, only three convolutions in frequency domain have to be carried out. The result of the convolution is another two dimensional field which hold the convolution for every point $(x_p, y_p)$. The three convolutions are carried out at the beginning of the algorithm and the result is stored in the memory of the planning system. The dose of each individual point in then derived as a sum over three multiplication term between the stored convolution material and the appropriate depth component.

A state-of-the-art implementation of the pencil beam convolution technique is for instance described by Siggel et al. (2012). In this work, the FHT is calculated by the widely known FFTW (Fastest Fourier Transform in the West) library (Frigo and Johnson 1998). The FFTW library is known as one of the fastest implementation for computing the discrete Fourier transform in one or more dimensions [1]. It is capable of exploiting modern multi-core processors and can be linked directly into existing C and C++ code. Siggel et al. (2012) also describes a fast solution for retrieving the depths $d$ of a certain point in the patient volume. The depth $d$ is understood as the radiological depth and has to be calculated for every point of interest in the patient volume. This is commonly done by a ray-casting method as for instance proposed by Siddon (1985). In the work of Siggel et al. (2012) this method is adapted for modern multi-core CPUs. A patient volume partitioning scheme is proposed which on the one hand exploits the transportation of clinical data from the RAM of the planning system to the processor. On the other hand the locality of clinical data is improved which significantly speeds up the algorithm. In addition to that the number of rays which have to be probed is greatly reduced by introducing a the ray-matrix concept. In a further investigation based on this paper, it is concluded that the ray-casting problem is one of the bottlenecks of the pencil beam convolution algorithm. On common desktop computers both problems, the determination of the radiological depth and the two-dimensional convolution in frequency domain need approximately the same amount of runtime to be executed for a complete dose calculation. The convolution is a compute-bound problem and can be accelerated using more computational cores and faster processors. The ray-casting problem is a bandwidth limited problem which is not easily accelerated using standard computer hardware. The determination of the radiological depth must only be carried out once, before the dose calculation starts. As long as the patient geometry does not change (which is assumed for standard IMRT wihtin one course of planning) the radiological depth also does not change. For adaptive therapy planning it might be necessary to update the radiological depth profile of the patient. For that, GPU accelerators can be used as shown by Gu et al. (2009). However, the ray-casting problem is not in the focus of this work and will therefore not be discussed in detail.

[1] http://www.fftw.org/speed/Pentium4-3.60GHz-icc/

Sidebar coordinate systems Most of the calculations in this section are performed in the so called 'fan-line' coordinate system. The fan-line system is based on the gantry system described by (Siddon et al. 1981). The gantry system and the fan-line system are illustrated geometrically in figure 3.3.

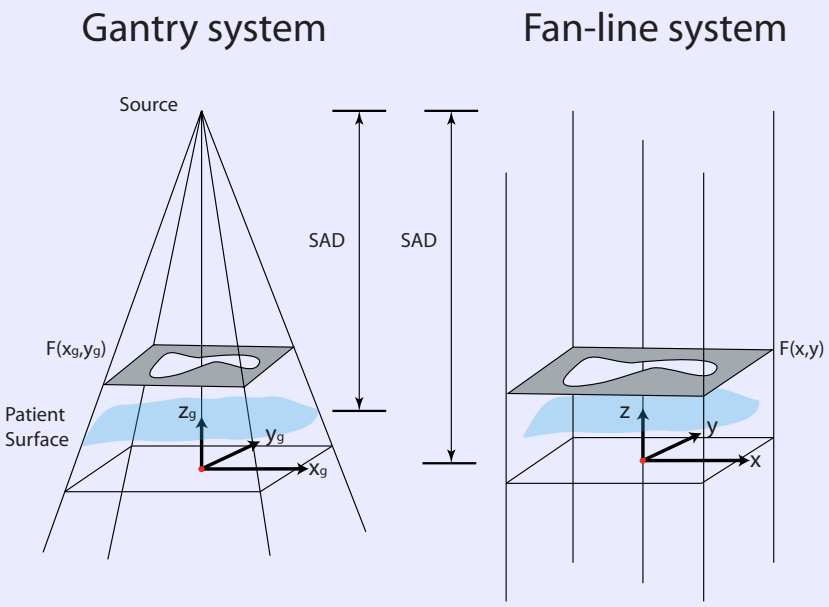



**Figure 3.3.:** Illustration describing the gantry coordinate system and the fan-line coordinate system for radiation therapy.

Both systems describe the beam setup and the correlation to the patient geometry for radiation therapy. In both, the gantry system and the fan-line system the z-axis $z_g$ is directed towards the source of the radiation while the y-axis $y_g$ is parallel to the gantry rotation axis. The origin of both coordinate systems is the iso-center of the treatment plan setup. The difference between both systems is that in the gantry systems all rays are originating one single source point while in the fan-line system these rays are transformed to parallel lines. The parallel geometry simplifies the notion of most equations concerning the dose calculation. Since the distance between the radiation source and the patient is usually relatively large, the opening angle of the ray gets small and the transformation between these two coordinate systems can be approximated by:

$$x = x_g \frac{SAD}{SAD - z_g} \tag{3.4}$$

$$y = y_g \frac{SAD}{SAD - z_g} \tag{3.5}$$

$$z = z_g \tag{3.6}$$

The kernel decomposition approach will be partially denoted in the cylindrical coordinates $(r,\varphi,d)$ which are in the fan-line system:

$$r = \sqrt{x^2 + y^2} \tag{3.7}$$

$$\varphi = \arctan \frac{y}{x} \tag{3.8}$$

$$d = -z + (SAD - SSD) \tag{3.9}$$

# 4. Efficient Computational Implementation of the Inverse Planning Problem

Solving the inverse planning problem is the commonly used means for clinical IMRT planning. Many treatment planning systems and research groups worldwide have implemented a plan optimizer which is based on the concepts introduced in section 2
Modern applications of radiation therapy planning are focused on the issue of adapting a therapy plan quickly to a changed patient geometry. The so called adaptive radiation therapy (ART) is a very popular field of research (Godley et al. 2009, Fu et al. 2009, Raaymakers et al. 2009, Wu et al. 2008, Lu et al. 2008, de la Zerda et al. 2007, Mohan et al. 2005, Birkner et al. 2003, Wu et al. 2002, Yan et al. 1997). In order to react to geometry changes in a reasonable time, it has become crucial that the runtime of the plan optimizer (time to plan) is as short as possible. In this chapter, the inverse problem is discussed from a computational point of view. First, the problem is analyzed and limitations are discussed. In the second part of this section highly efficient concepts for an ultra-fast implementation are described on different CPU-based platforms. Current clinical implementations of the plan optimizer exploit a runtime in the order of minutes to find a treatment plan. In the following it is shown how to achieve a result in a few seconds. Furthermore, it can be proven that the implementation introduced for shared memory systems is optimal on the given class of computer systems. It exploits nearly all the hardware resources efficiently, so that there cannot be another significantly faster realization of that specific planning problem.
The outlook in chapter 4.7 motivates the new planning paradigm IDS (chapter 5)which is the main topic of this work

## 4.1. The computational problem of IMRT treatment planning

Using the pre-calculated dose influence matrix approach as described in the previous section, the iterative optimization is a bandwidth limited problem. This can easily been see on the example of one back-projection step in the optimization loop. The algorithm executes only simple arithmetic operations while each iteration loops over the whole dose data stored in the dose cube and the $D_{ij}$ matrix. To update the physical dose in one voxel $i$ with the dose influence coming from a bixel $j$, the algorithm has to carry out five data related operations: Since the dose influence matrix is stored with full voxel index, the corresponding index is first extracted from the current dose matrix element. Second, the dose influence value (the actual $D_{ij}$ value) is read from memory. Third the current dose in voxel $i$ is read and after the update operation is done, the new dose value has to be written back at the appropriate position in the dose cube (see pseudo code A.1). In contrast to that, there are only two arithmetic operation involved in this update process: First, the multiplication of the dose influence value with the bixel weight and second the incrementation of the dose in voxel $i$ with the result of the former operation. Thus, the ratio between arithmetic operations and data transfer operations is 2:5. Given the fact that data is retrieved up to two order of magnitude slower from memory than it is processed by the arithmetic unit, this dose update operation causes an unbalanced used of these two computational components. It computer science this mismatch forms a so called von-Neumann bottleneck (Backus 1978) so that data is not provided to the arithmetic units at an adequate speed. The potential of the CPU is not fully exploited which results in an unsatisfying overall performance of the algorithm.
Modern CPUs are build to process instructions in parallel. Due to the limitation on ever increasing clock speeds the hardware industry was forced to come up with a new paradigm to further increase the performance of their

processors. Certain computational units (cores) are replicated on the processor die so that multiple instruction streams can be processed in parallel. Thus, high performance on modern CPUs comes from the strategy to process multiple instructions and data streams and not from executing operations itself just faster. Although, the IMRT planning problem is not computational intensive and planning performance depends on the data transfer rate, an efficient implementation of the inverse problem has to be parallel, too. The organization of higher memory in modern CPUs follows the multi-core architecture, so that the peak bandwidth corresponds to the aggregated bandwidth of all memory entities within one hierarchy. Unfortunately, even modern compilers are not capable of producing efficient parallel code automatically. The concurrency of an algorithm has to be described manually by an additional parallel API (application programming interface). This requires, that the underlying algorithm has to employ one of the following strategies: First, it must be possible to define a set of subtasks which are independent and can be executed concurrently. Second, the algorithm should execute a set of operations homogeneously on a large set of data, so that the workload can be apportioned and processed by concurrent execution streams. The IMRT planning algorithm is a typical example of the latter algorithm. Consider for example the physical dose calculation according to equation 2.2. The matrix-vector product between the dose influence matrix and the bixel weight vector is carried out for each voxel individually. Given that these two data structures are globally visible for the processor, a possible strategy to parallelize this operation would be to assign a certain subset of all voxels $i$ to each concurrent process, respectively. Each process executes the dose calculation according to equation 2.2 only on its subset of voxels, so that the effective runtime is expected to be $1/n$-th of the runtime compared to a serial execution of this operation, while $n$ is the number on concurrent processes. The same consideration applies to the calculation of the first gradient of the objective function (equation 2.6). After the operation has been carried out on all voxels, the processes have to be synchronized to exchange results which have been acquired in the concurrent runs. Synchronization is considered as one form of a *parallelization overhead* and has to be done as efficiently as possible.

## 4.2. Strategies on one-processor systems

Since the inverse planning problem is memory bound, an efficient data handling is vital for a fast solution. Two basic types of data structures are used in the implementation. The first basic data structure describes the *cubed data*, e.g. the dose cube $d_i$. Cube data is stored in a vector while the value at each index $i$ is assigned to the $i$-th voxel in the three dimensional patient cube. Second, a data structure for the $D_{ij}$ data set has to be defined. The dose influence matrix is sparsely populated and stored for each individual beam in an indexed method. The elements of a column $j_m$ in the dose influence matrix $D_{ij}$ are stored sequentially in a vector while each element is tagged with an voxel index $i$ representing its original position in the $i$-th row of the $D_{ij}$ matrix. The benefit of this method is that the index is directly accessible and stored spatially together with the dose influence matrix value. It could be shown (Ziegenhein 2008) that fast data access is guaranteed if the elements in vector $D_{ij_m}$ are sorted according to the corresponding index in the dose cube, so that $i_n < i_{n+1}$.
To account for the parallelization approach introduced before, a *domain decomposition* strategy is used which disassembles the dose influence data (see right side of figure 4.1). The dose influence data is decomposed along the z-axis into a set of slices containing one plain of voxels. Figure 4.1 shows the first two and the last slice of the patient cube and the corresponding dose influence elements in the $D_{ij_m}$ vector. Although, each slice contains an equal number of voxels, the number of dose influence elements can vary among the slices. A set of cube segments are defined which contain one or more slices of the patient cube and the corresponding influence data. Due to the decomposition of the $D_{ij}$ along the slices of the patient cube, it is reasonable to store other cubed data in a per slice fashion, too. Thus, the decomposition of the $D_{ij}$ data induces a decomposition of the cube data as well. A technical notation of the $D_{ij}$ data structure and the cube data structure on the example of the dose cube is shown in listing A.1. The $D_{ij}$ data is stored in a three dimensional array which sorts the dose contribution according: first, to the slice in the patient cube which is influenced, second the beam which

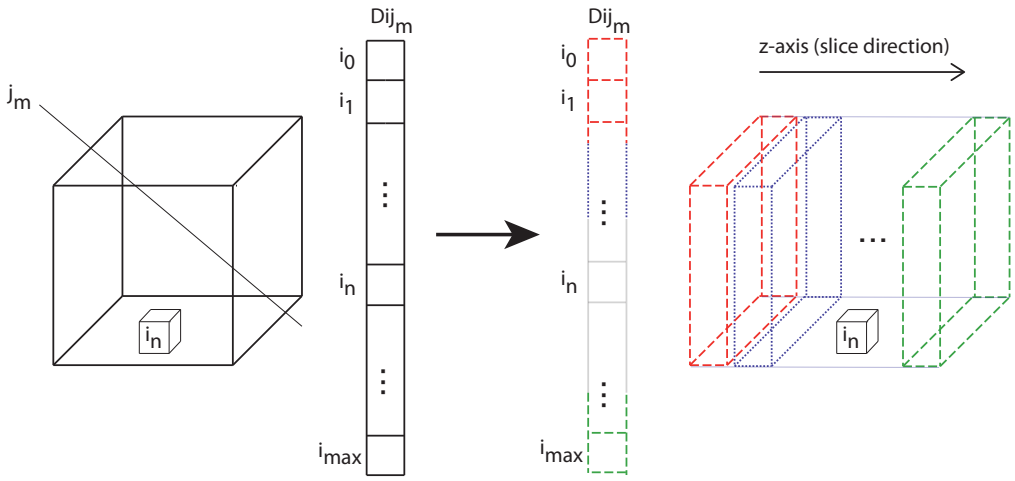



**Figure 4.1.:** The concept of the pre-calculated dose influence matrix. Left: One column of the $D_{ij}$ matrix contains the dose influence elements of one bixel to the voxels of the patient. Right: Segmentation of the dose influence data according to the slices of the patient cube.

is influencing and third the corresponding bixel number. The data is stored in packages (denoted by **struct**) of 4 elements. In listing A.1 the cube data structure is defined on the example of the physical dose which is stored as double precision floating point numbers (**double**). Other cube data instances, as for example the VOI cube uses an integer value which would be stored by data type **short**.
The implementation of the IMRT planning problem presented in this work uses the L-BFGS quasi-Newton approximation together with the Armijo line search which is introduced in section 2.1. The general workflow presented in figure 2.3 was extended to contain the Hessian approximation and line search method as shown in figure 4.2. The workflow contains three main modules which are iteratively executed in each iteration: the

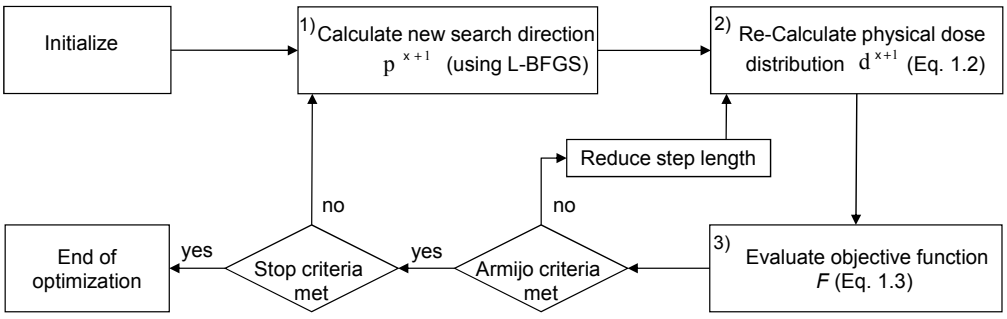



**Figure 4.2.:** Workflow and modules of the CPU-based ultra-fast therapy plan optimization implementation.

calculation of a new search direction which also includes proposing a new set of bixel weights using the standard step length $\alpha = 1.0$, the update of the physical dose using the new set of bixel weights and the evaluation of the new dose patter according to the objective function. Each module executes a certain operation on the clinical planning data. It is possible to define these operations as computational *kernels* which operate on one voxel only. Thus, the module is then executed by a set of kernels which each operate on a different voxel until the whole clinical data set is processed. This strategy guarantees an effective use of the data structures defined

in listing A.1, since the data is organized per slice containing a fixed number of voxels. Three basic kernels executing the three main modules of the workflow are defined: First, the *projection kernel* which calculates the gradient of the objective function to obtain a new search direction **p** and a new set of bixel weights. The kernel only contains the gradient determination, and not the L-BFGS approximation because on the one hand the computational effort of calculated the L-BFGS approximation is negligible and on the other hand this operation cannot be done efficiently in parallel. According to equation 2.6 the projection kernel can be identified as:

$$\frac{df_i}{dw_j} = 2s_i \left[d_i - d_i^{pres}\right] D_{ij} \tag{4.1}$$

The prescribed dose $d_i^{pres}$ and the penalty cube $s_i$ are both cube data structures which are updated in every iteration. $d_i^{pres}$ holds the prescribed dose for each voxel depending on the organ type of the voxel and the current dose distribution. $s_i$ provides the current penalty factor of voxel $i$ which has to be updated in each iteration, too. The second kernel that is introduced is the *back-projection kernel* which calculates the physical dose of one voxel $d_i$ according to the current set of beam weights. The definition is identical to equation 2.2 but repeated for the sake of clarity:

$$d_i = \sum_{j \in B} D_{ij} w_j, \tag{4.2}$$

The third kernel which evaluates the current dose distribution is the *objective function kernel* $f_i$ which calculates the weighted deviation of the current dose to the prescribed dose in voxel $i$:

$$f_i(d_i) = s_i \left[d_i - d_i^{Pres}\right]^2 \tag{4.3}$$

In order to run the kernels efficiently on modern hardware, is must be specified how the workload is distributed onto the CPU-cores of the processor. The basic structure of a typical desktop processor is shown in figure 4.3 The CPU consists of four identical arithmetic cores (light gray area) each of them is equipped with its own high level memory block (cache memory, blue area). The RAM (main memory) is not part of the processor (green area) and sits on the mainboard of the planning system. During the execution of an algorithm each CPU retrieves data from the shared main memory and holds a copy of it for faster access in cache memory. The architecture shown in figure 4.3 is called *Uniform-Memory-Access* (UMA) architecture, because the time for fetching an arbitrary data word from the main memory is the same for every CPU core. Modifications made to the data in one cache memory block are automatically propagated to the other three cache memory blocks. Thus, the UMA architecture implements a cache *coherent* systems which make is easier for the software developer to design programs for that parallel system.
An intuitive way of distributing the workload would be to divide the patient cube into 4 segments each containing an equal number of slices and assign one segment to each CPU core. Although the cube data would be distributed equally, the arithmetic workload would not. In a coplanar IMRT beam setup, the incident beams are targeted towards an iso-center point which is typically situated in the middle of the patient cube. This geometry creates a spheric-symmetrical dose intensity pattern, so that the total number of dose influence elements deposited in the patient cube is decreasing with $1/r^2$ with the distance $r$ from the iso-center. This implies that the work load for e.g. the gradient kernel (equation 4.1) is lower for slices situated on edge of the patient cube compared to slices situated in the middle of the patient cube. If this strategy would be realized in an implementation on the UMA architecture, the cores processing the segments including the edges of the patient cube would be faster than the core processing the middle segments. The workload is unbalanced and the total runtime depends on the slowest core. A more efficient way of distributing the workload is to assign individual slices dynamically to idle CPU cores. The dynamic scheduling is possible because of two reasons: First, due to the UMA architecture the clinical data has no prior affinity to the CPU cores and the overhead for creating

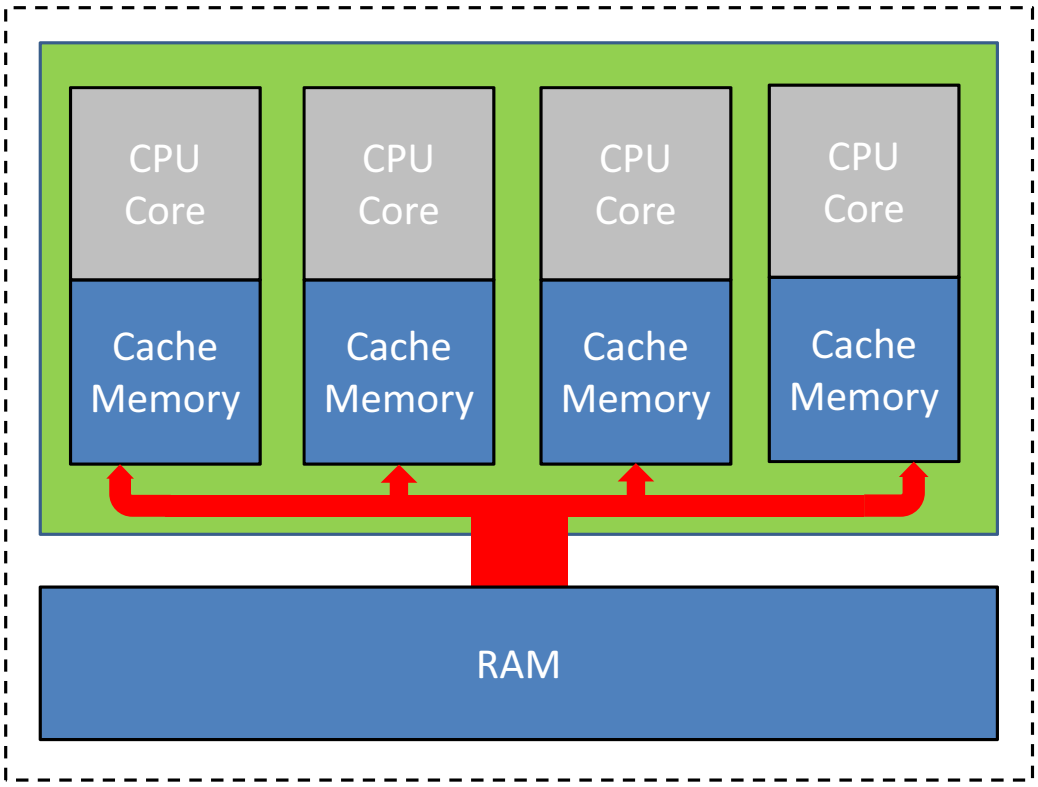



**Figure 4.3.:** Illustration of the UMA (Uniform Memory Access) shared-memory architecture of parallel computers. The configuration has 4 CPU cores (grey) each having a dedicated cache memory (blue) and interconnection (red) which allows for a uniform memory access.

threads to schedule the work is very low. Second, the data structures and arithmetic kernels are designed independently, so that the locality and order of execution can be chosen arbitrarily. Furthermore, due to the dynamic scheduling each CPU core retrieves a mix of cube slices which come from different locations in the patent cube instead getting one segment of consecutive slices. That way, the work load of each segment is balanced and the multiple cores are used efficiently. The general structure of the parallel algorithm is shown in listing A.2. Using the data structures defined in listing A.1 the dynamic scheduling of the parallel workload is triggered with the single OpenMP command in line 3. The developer does not have to specify any other parallel code which makes this approach very easy to reproduce.

## 4.3. Strategies on multi-processor systems

To improve the runtime even more, a higher number of processing cores having a fast connection to the main memory is needed. However, the uniform memory concept shown in figure 4.3 does not scale well for a core count in the order of over 10. Furthermore, all cores have to share the bandwidth of the network connecting the processor die (green array) with the main memory. Since, the IMRT planning problem is memory bound, a higher number of cores in the UMA architecture would not lead to a significant speedup. However, the runtime can be improved significantly, if the algorithm is adapted for multi-processor system, as shown in figure 4.4. In order to increase the number of CPU cores with a fast connection to the main memory, several (in this configuration 4) processors are placed within one computational system each having an own domain of main memory. One domain of main memory (RAM A,B,C or D) is assigned to each processor and interconnected by multiple

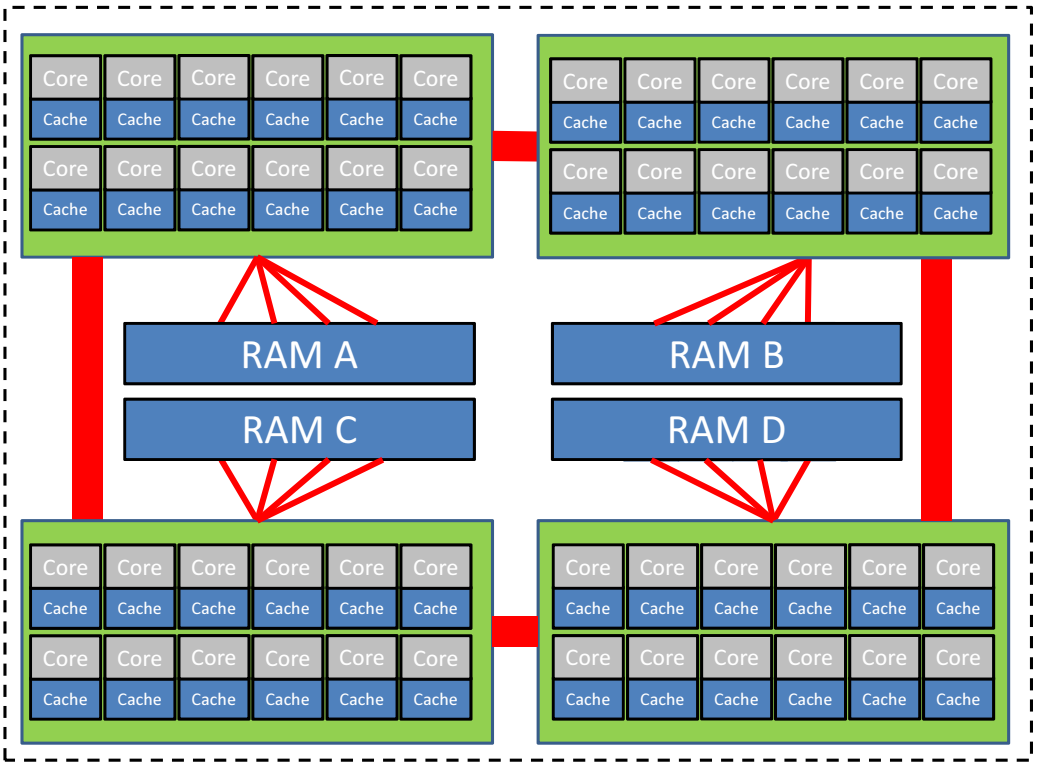



**Figure 4.4.:** Illustration of the NUMA (Non-Uniform Memory Access) shared-memory architecture of parallel computers. The configuration has 4 processors (green) each having 12 cores (grey) with dedicated cache memory (blue) and interconnection (red) which is non-uniform.

memory channels. Each CPU core has access to all of the main memory domains, although the time to access is not the same for all cores. For instance, cores from processor I (left, top) can access data in RAM A faster than data in RAM B. The cache of all 48 cores are still coherent, which is technically very challenging given the high number of cores (and caches) per processor. The architecture shown in figure 4.4 is called cc-NUMA (cache coherent non-uniform memory access) architecture, because of the cache coherency and the affinity of RAM domains to a processor.

Implementing an efficient IMRT planning algorithm for a NUMA system must take the non-uniformity of memory accesses into account. The developer has to guarantee two conditions: First, all planning data ($D_{ij}$ matrix and all cube data) has to be divided among the four memory domains, so that the workload lying in each domain is equally distributed. This can be done in a similar fashion as for the UMA architecture. One slice after another is assigned to the memory domains using a round-robin strategy (Silberschatz et al. 2009). Thus, slice 0,4,8,12... is assigned to processor 0, slice 1,5,9,13 to processor 1, slice 2,6,10,14 to processor 2 and so on. Second, to avoid long memory access times, the data in one memory domain has to be processed by the threads of the nearest processor only. This is not trivially implemented since thread usually migrate from one processor to another following a complicated hardware based scheduling system. Furthermore, the thread correlation must not change between different kernels and iterations. To accomplish that, the software developer must have a deep knowledge of the NUMA architecture and the operating system. For the implementation presented in this work, the libNuma library [1] on linux has been used. It provides a low-level interface to the developer for establishing a strict thread-node binding throughout the whole planning process. The paralleliza-

[1] http://linux.die.net/man/3/numa

tion has be described manually as shown in listing A.3 before the kernel execution starts. In line 1 of listing A.3 the ID of the current thread is determined. On a NUMA system as many threads as available CPU cores are launched. Each thread is bound to a specific node (processor) in line 2. The round-robin pattern mentioned above to distribute the clinical data among the memory domain is implemented by the loop shown in line 3 with the variable *thread_quan* holding the total number of threads used for the planning process. The round-robin is implemented by using the thread id as the initial value for the loop controlling variable.

## 4.4. Strategies on cluster systems

The highest performance in HPC (high performance computing) offers a cluster system. A cluster consists of several UMA or NUMA systems as described in the previous two sections. The entities are individual computational units that are interconnected via a network for exchanging data or for synchronizing during the parallel execution. Figure 4.5 shows an example for a cluster consisting of NUMA nodes. The memory and the CPU

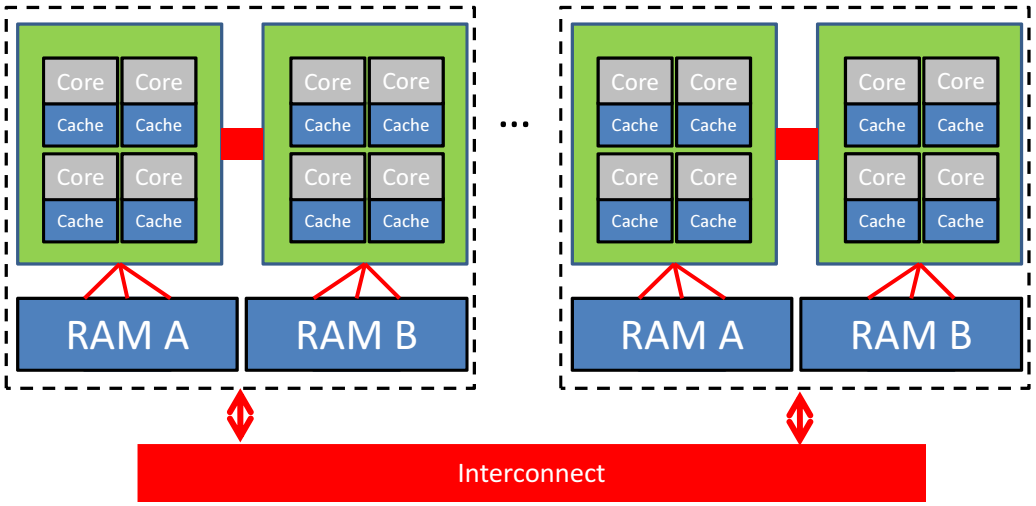



**Figure 4.5.:** Illustration of a cluster architecture. Several individual computers are interconnected by an external network system.

cores of the cluster are physically separated from each other and communication is expensive. However, the parallelism and the computational performance are very high.

The use of cluster computing in the clinical routine is very limited. Clusters are relatively expensive and need professional maintenance. It is not expected that clusters will be commonly used for therapy planning. However, this section describes the scaling of the proposed IMRT planning solution to a cluster environment and shows limitations of the algorithmic concept.
In contrast to the implementation on UMA and NUMA systems, the cluster implementation follows the simpler and more general workflow. The search direction is calculated using the approximation in equation 2.7 for the Hessian matrix. So, the projection kernel now consists of two sub-kernels $u_{ij}$ and $v_{ij}$ which denote the gradient

of the objective function and the Hessian approximation:

$$u_{ij} = p_i \left[ d_i - d_i^{pres} \right] D_{ij} \tag{4.4}$$

$$v_{ij} = p_i D_{ij}^2 \tag{4.5}$$

$$p_j = \frac{\sum_i u_{ij}}{\sum_i v_{ij}} = \frac{\sum_i p_i \left[ d_i - d_i^{pres} \right] D_{ij}}{\sum_i p_i D_{ij}^2} \tag{4.6}$$

Armijo's rule for evaluating the step length is not applied. Instead, a fixed step length $\alpha_0$ (called damping factor) is used which is calculated once at the beginning of the optimization according to chapter 2.4.1.2 in (Nill 2001).

As already discussed earlier, the kernels have several properties which are beneficial for a fast distributed implementation. First, the kernels can operate independently. No communication between the processes is required until every voxel in the dose segment was processed by one specific kernel. This observation motivates the definition of a *working unit* for the implementation on cluster which applies a kernel on all voxels within one segment (e.g. working unit $u_{ij}\{i \in b_k, j\}$ performs a calculation of kernel $u_{ij}$ on all voxels in segment $b_k$ which are influenced by bixel $j$). The result of a working unit represents the data which is exchanged with other processes via the network. Secondly, the optimization algorithm only depends on the superposition of the calculation kernels over all voxels. Also, the order in which the voxels were processed can be arbitrary. That means, a working unit of a specific kernel may add the result of each kernel immediately to a global result variable in each concurrent process. This immediate superposition (*pre-reduction*) of a kernel results will be marked with a '+' at the end of a working unit definition (e.g. $u_{ij}\{i \in b_k, j\}^+$). The size of the working unit result than equals the size of a single kernel result. For the objective function kernel, this is only one scalar per process. For the working units $u_{ij}\{i \in b_k, j\}$ and $v_{ij}\{i \in b_k, j\}$ one array per process is produced which holds the kernel results for each bixel. This is necessary, because new weights are updated for each bixel individually.

The communication between the processes is now easy to describe by using the aforementioned definition of kernels and working units. Communications are needed to share the results of the working units. According to the workflow in figure 2.3 this is necessary only twice every iteration. First, after calculating the working units $u_{ij}\{i \in b_k, j\}$ and $v_{ij}\{i \in b_k, j\}$ for updating the beam weights. Each concurrent process transfers the result of these two working units to every other process in the system so that every processes has all the information needed to calculate a complete set of new beam weights. The calculation of new weights is then carried out in each process by using the pre-reduced working units:

$$w_j^{t+1} = \left[ w_j^t - \alpha_0 \frac{u_{ij}\{i \in b_0, j\}^+ \dotplus \cdots \dotplus u_{ij}\{i \in b_{n-1}, j\}^+}{v_{ij}\{i \in b_0, j\}^+ \dotplus \cdots \dotplus v_{ij}\{i \in b_{n-1}, j\}^+} \right]_{\geq 0} \tag{4.7}$$

This operation can be executed very fast in each process because the the result of every working unit already has been calculated by the concurrent processes itself. Furthermore, the final reduction operator $\dotplus$ which adds up the pre-reduced working units is carried out during the communication process over the network. The operation $\dotplus$ is implemented in the network API and is already highly optimized. So, the only operation which is left to each process for one bixel after the working units have been transmitted, is to evaluate the fraction, multiply the damping factor $\alpha_0$ and subtract this result from the old bixel weight.
According to the workflow a physical dose calculation based on the new bixel weights has to be performed in the next step. This could be done for each dose cube segment individually, because the new weights were broadcasted to all processes in the last step. After finishing the dose calculation, there is no communication needed to transfer physical dose data between the processes. It is not necessary that a process needs to know the dose distribution of another segment.
The quality of the current plan is determined by the objective function as defined in equation 2.3. Taking into

| ID | beams | bixel | Voxel [$10^6$] | $D_{ij}$ elem. [$10^6$] | iter | UMA system runtime [s] | UMA system Bandwidth [GB/s] | NUMA system runtime [s] | NUMA system Bandwidth [GB/s] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 456 | 0.4 | 26.3 | 36 | 1.05 ± 0.01 | 20.2 ± 0.3 | 0.32 ± 0.01 | 73.24 ± 6.45 |
| 2 | 9 | 1261 | 1.7 | 101.7 | 84 | 10.50 ± 0.11 | 19.4 ± 0.6 | 2.13 ± 0.02 | 80.11 ± 3.09 |
| 3 | 9 | 3423 | 4.2 | 259.5 | 40 | 11.70 ± 0.10 | 19.3 ± 0.7 | 2.29 ± 0.01 | 85.602 ± 1.54 |
| 4 | 2 | 3825 | 6.8 | 25.7 | 101 | 7.50 ± 0.11 | 18.4 ± 0.9 | 2.83 ± 0.41 | 33.83 ± 3.07 |

**Table 4.1.:** Clinical planning cases for the performance evaluation of $\mu$KonRad on UMA and NUMA systems. In all cases the voxel size is $(2.62mm)^3$

account the definition of the kernel in equation 4.3 each process calculates a working unit $f_i\{i \in b_k\}$ individually while using the correct penalty factors $s_i$ for each VOI. The final result of the objective function is then retrieved by a similar broadcast and reduction operator as used before:

$$F = f_i\{i \in b_0\}^+ \dot{+} \cdots \dot{+} f_i\{i \in b_{n-1}\}^+ \tag{4.8}$$

Although, the final result of the objective function is broadcasted to every process only one of them needs to evaluate the stopping criteria of the optimization. Thus, the broadcasting would not be necessary. However, the communication effort of transferring the single valued result of the objective function once per iteration is negligible. If the objective functions reveals a significant improvement of the therapy plan so that the stopping criteria is not met, the calculation of new beam weights starts again.

## 4.5. Results

The concepts for a fast treatment planning realization were implemented into a new planning engine called $\mu$KonRad. The $\mu$KonRad framework provides the functionality to read clinical data (including the pre-calculated dose influence matrix) from the hard drive of the planning computer and perform a fast optimization using Newton's method and the L-BFGS approximation. A graphical user interface is not part of $\mu$KonRad.

In the following two sections the performance of the planning engine $\mu$KonRad will be described on shared memory systems (section 4.5.1) and on distributed (cluster) system (section 4.5.2). This investigation focuses on the computational performance of the proposed treatment planning concepts only. A discussion of the general clinical quality of the resulting plans will not be provided since the performance of the optimization is independent from the clinical parameters or the context of the data.

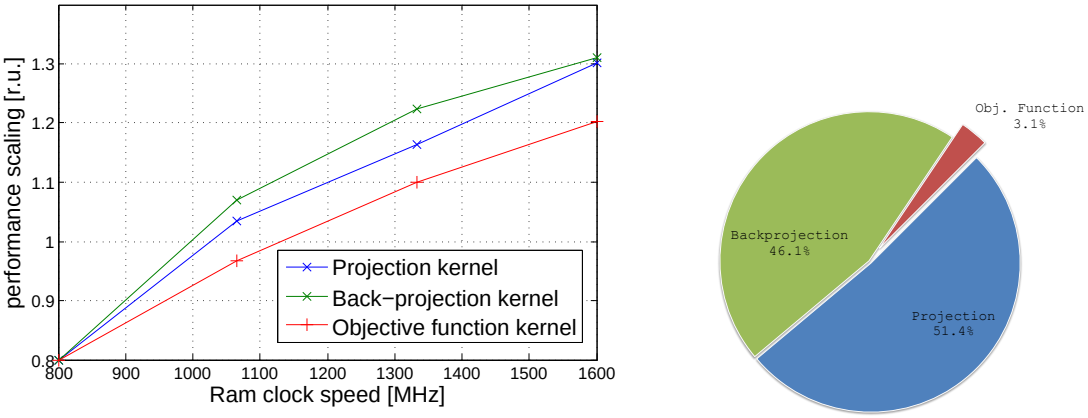
### 4.5.1. Performance of the IMRT planning algorithm on shared-memory systems

The performance of therapy planning in $\mu$KonRad on shared memory systems is shown in table 4.1 on the example of four clinical plans. The first plan (ID 1) describes a clivus chordoma case which is planned with photons. The second and the fourth plan are both prostate case which are also planned with photons (all at 6 MV). The third plan describes an abdominal case which is planned with protons (3D IMPT).

The performance of the treatment planning implementation is expressed by the over-all runtime of the optimization and by the achieved bandwidth of transporting data between the RAM and the CPU of the planning system. Since the plan generation algorithm is a memory-bound problem, the achieved bandwidth is a quality indicator of how well the resources of the planning system is used. The runtime value stated in table 4.1 was acquired as a mean value over 10 plan optimization runs. The standard deviation is denoted after the ± sign. The bandwidth was recorded for the physical dose calculation (equation 2.2) in each iteration. The given values

are the mean bandwidth over all iterations and the standard deviation.

To illustrate the dependency of the planning algorithm of the memory bandwidth, the relative speed of the computational kernels defined in section 4.2 for the example of plan 2 is recorded in figure 4.6(a) for various clock speeds of the systems memory. As a representative for the UMA system a configuration containing the




(a) (b)

**Figure 4.6.:** Performance of the optimization of plan 2. a) The relative calculation speed of the computational kernels is shown on plan 2 depending on the clock frequency of the system memory. b) The relative portion of the kernel runtimes relative to the total planning time.

low-cost desktop processor Intel i5-2500K with 8 GB of DDR3 RAM was used. The NUMA system was a server configuration with 4 AMD Opteron 6168 processors and 128 GB of DDR3 RAM.
The results on the UMA system have already been published in Ziegenhein et al. (2013).

### 4.5.2. Performance scaling of the IMRT planning algorithm on cluster systems

To answer the question whether or not the presented IMRT planning algorithm is suitable for clusters and scales on a high number of distributed computing cores, two aspects of the algorithm must be discussed: The inter-process communication between the nodes of the cluster on the one hand and the scheduling problem of defining appropriate dose cube segment sizes on the other hand. This section will present results on both aspects. Runtimes for IMRT planning on clusters will be shown at the end of this section.
Chapter 4.4 revealed that a communication between the concurrent processes is only necessary twice in each iteration: after computing the working unit result $u_{ij}\{i \in b_k, j\}^+$ and $v_{ij}\{i \in b_k, j\}^+$ for the calculation of new bixel weights (projection kernel) and after computing the working unit result $f_i\{i \in b_k\}^+$ for evaluating the current value of the objective function. The effort of transferring one result of the latter kernels is very low since only one single value has to be transmitted. The reduction of the former working units requires each process to send two arrays into the network. The number of elements in each array equals the number of bixels which are used in the treatment plan setup. Assuming a total number of 1000 bixels and 10 concurrent processes, the amount of data which has to be handled by the interconnection during this communication is $2 * 1000 * 8\ Byte * 10 = 0,16\ MByte$. Compared to modern data bus performances and network speeds this effort is relatively low and is not expected to be a bottleneck.
Besides transferring data, in the presented implementation a communication process always implicitly triggers a synchronization barrier for all processes. Thus, the parallel algorithm has to wait for the slowest process to send its data into the network. This is necessary because the algorithm can continue to the next module not before each process finishes the current working unit.
In order to prevent delays due to unbalanced runtime all processes should be able to execute their working units

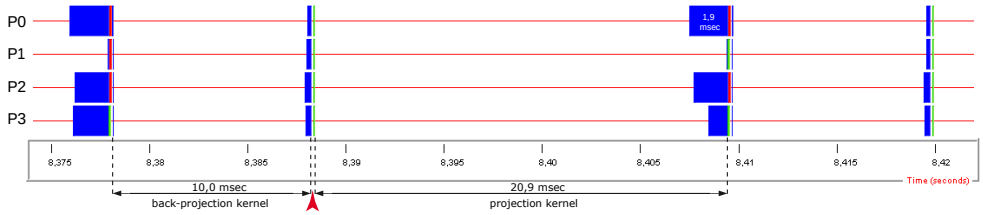



**Figure 4.7.:** A time line diagram for one iteration of the optimization of a typical IMRT therapy plan showing 4 concurrent processes.

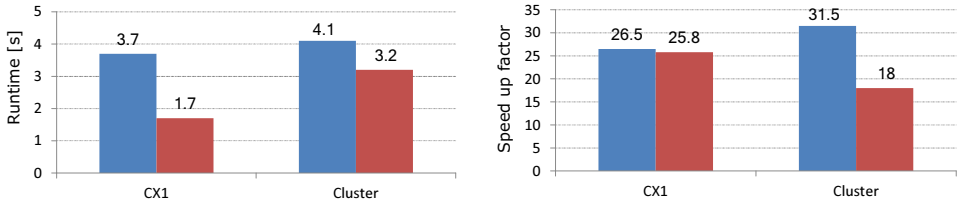



**Figure 4.8.:** Therapy planning time on cluster computers for plan 1(blue) and plan 2(red). The speed up factors are relative to the original KonRad framework.

in the same amount of time. This is achieved by an appropriated segmenting of the dose cube. As it turned out the solution to this *scheduling problem* is not obvious: For central slices of the dose cube the density of dose contribution elements (i.e. $D_{ij}$ elements) is much higher than for slices located at the borders of the dose cube. This is because the radiation target point for the coplanar beam setup is supposed to be near the cube center. Thus, segments located in the center of the dose cube should contain fewer slices than other segments. A simple method of finding an appropriate set of segments for each concurrent process would be to simulate for instance equation 4.7 on every dose cube slice prior to the actual optimization in a non-parallel environment. The segmentation for the parallel environment could then be chosen based on the recorded runtime for each slice. This is an accurate and fast estimation, since the additional runtime overhead would not exceed the runtime of one iteration of the planning algorithm without parallelization.
A time line diagram of a typical IMRT head and neck therapy plan is shown in figure 4.7. The diagram exhibits the state of the parallel algorithm for 4 concurrent processes at a certain time frame [2]. The four processes marked by P0 to P3 are arranged as horizontal time lines (red line). White spaces in this graph denotes the time in which the kernels are actually executed. During the blue boxes the process waits for a synchronization between them to happen. Green fractions of time are spend for the communication over the network. The first and third vertical group of boxes describes the communication to broadcast the result of projection kernels. The second and fourth communication is designed to exchange the results of the objective function kernels. Please note, that the graph is just a small portion of whole time line which was recorded during the optimization. It only shows one complete iteration step (marked by the arrows beneath the time lines).
The runtime of the IMRT planning algorithm was evaluated on two different cluster configurations (figure 4.8) Two clinical therapy plans were optimized on each system with a various number of concurrent processes. The processes were managed using the parallel programming interfaces OpenMP and MPI. The choice of which API is appropriate for which parallel system is based on technical issues and should not be discussed in this paper. If MPI was used a concurrent process is called *MPI process*. Concurrent processes which are managed by OpenMP are called *threads*. The first clinical case is the treatment of a clivus chordoma comprising 7

[2]The time line diagram was taken from a screen-shot using the performance analysis program Jumpshot (http://www.mcs.anl.gov/research/projects/perfvis)

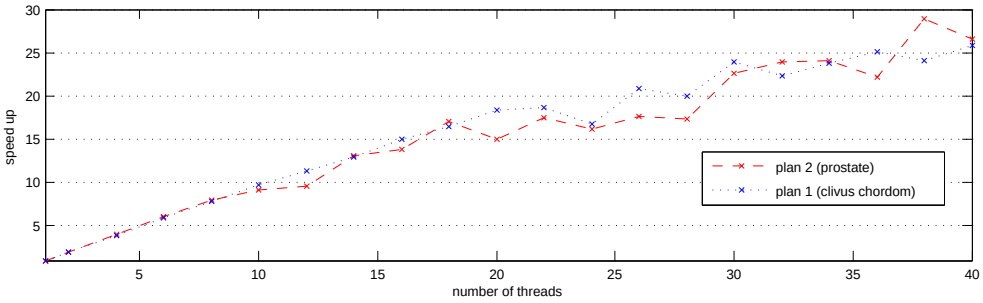



**Figure 4.9.:** Relative performance of the parallel therapy planning algorithm on Cray's *CX1* personal supercomputer. For details about the planning system and the clinical cases see table 4.3 and 4.2. The reference speedup of 1 equals 101,4 seconds and 45,2 seconds of optimization runtime for the prostate case and the clivus chordoma case, respectively.

incident beams. The second plan describes a common prostate treatment with 5 beams. Both patients were planed with photons. Further details of the plans can be found in table 4.2. A brief characteristic of the parallel systems used could be found in table 4.3.

| Plan | Beams | Dose Cube size (x,y,z) | Voxels [Mio] | Bixels | $D_{ij}$ elements [Mio] |
|---|---|---|---|---|---|
| 1 | 7 | (241,256,60) | 3.7 | 456 | 26.0 |
| 2 | 5 | (256,256,76) | 4.9 | 400 | 54.7 |

**Table 4.2.:** Clinical cases for the performance evaluation on cluster computers.

| ID | CPU | Cores per CPU | Network | API |
|---|---|---|---|---|
| CX1 | 10× Intel Xeon | 4 | Infiniband | MPI |
| BQ | 24× Intel Xeon | 4 | Ethernet | MPI+OpenMP |

**Table 4.3.:** Description of the cluster platforms used for the performance evaluation of therapy planning.

The CX1 is one of Cray's *personal supercomputers*. The system is equipped with 5 blades each carrying two Intel Xeon Quad-Core processors. A ulta-low latency Infiniband network was used to connect the blades. The relative performance of a complete therapy planning carried out using the parallel algorithm on this system is shown in figure 4.9. The graph shows the performance speed up scaling which could be achieved. The speed up is calculated relative to the runtime of the algorithm while using only one process (non-parallel) on that system. The second system on which we tested our parallel treatment planning algorithm was a small scaled research cluster (BQ). The cluster consists of 12 individual computational units each providing 2 Intel Xeon Quad-Core CPUs connected via Ethernet. Here, we used a combination of MPI and OpenMP a so called hybrid parallelization model for our planning algorithm. OpenMP was applied within one computational unit while MPI was used to communicate via the Ethernet between the units. The performance achieved on the BQ cluster is shown in figure 4.10. For the prostate case a speed up of about 30 can be achieved using all the resources available. The clivus case could not gain such a high speed up because the amount of clinical data was too small to take advantage of the cluster architecture.

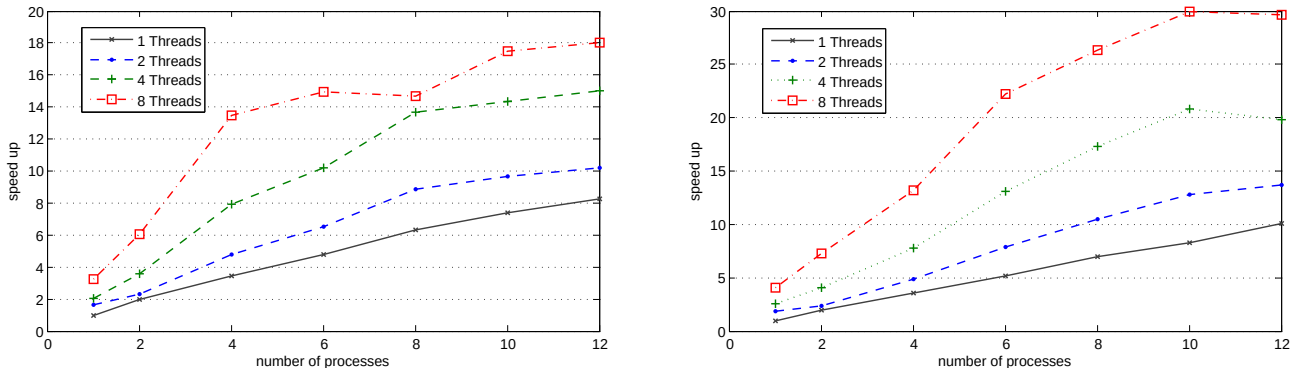



**Figure 4.10.:** Relative performance of the parallel therapy planning algorithm on the *BQ* research cluster for (left side) the clivus chordoma case and for (right side) the prostate case. Details about the planning system and the clinical cases can be found in table 4.3 and 4.2. The reference speedup of 1 equals 130,9 seconds and 59,1 seconds of optimization runtime for the prostate case and the clivus chordoma case, respectively.

## 4.6. Discussion

A typical clinical IMRT plan can be calculated within a few seconds through the use of Newton's method and a set of pre-calculated dose influence data on CPU based systems. This result is an important step towards adaptive radiation therapy (ART). In ART the treatment plan is modified during the course of treatment to compensate for any change on the patient geometry. With a real-time IMRT planning module it is possible to update the current treatment plan according to images which are acquired during a treatment fraction or between them. In addition to ART where the runtime of the plan optimization is in the main focus, also large scale planning problems like the beam angle selection (BAS) become feasible for clinical routine by using the fast concept introduced in this chapter. The fast plan optimization on NUMA system presented section 4.3 has been utilized recently for an extensive BAS study (Bangert et al. 2012) and (Bangert et al. 2013). From a computational point of view this study proves that the concepts for fast planning introduced in this work scale with the amount and size of clinical planning data. A planning time of about one second could be achieved for plans which had several thousand candidate beams. The size of the $D_{ij}$ data set in these cases was in the order of 100 GB.

The results in table 4.1 show that the IMRT planning algorithm as proposed in this work exploits all the hardware resources available on the shared-memory platforms. According to the discussion in section 4.1 solving the inverse planning problem based on pre-calculated dose data is a memory bound problem. Thus, the aim is to design an algorithm with uses all the available memory bandwidth of the computational system. On the UMA system a transfer rate of about 19 $GB/s$ for reading the $D_{ij}$ data was achieved. Since, the $D_{ij}$ matrix is only the largest data type among others involved in this problem, the proposes planning algorithm exploits at least 19 $GB/s$ of memory bandwidth. The theoretical possible bandwidth over all data of the UMA platform used to derive the results is 21 GB/s[3]. That means, over 90% of the hardware capabilities is used. The actual number of cores in the UMA processor is not important for the runtime. Although, it is important to use all available CPU cores for the following reason: The data transportation between RAM and CPU is controlled by the cache memories of the system. Each CPU core comes with its own cache memory unit each of them having a certain bandwidth to the RAM. In order to exploit the full bandwidth of the system, the data transportation has to use all cache memories which means that all CPU cores have to be used. So, it is not important that the arithmetic workload is shared by the core, but that each core helps to transport the data by providing the resources of its own cache memory. All desktop processors of this generation irrespective of the number of cores have the same

[3]http://ark.intel.com/products/52210

theoretical peak bandwidth when all cores are used.
The peak bandwidth of the NUMA system is not trivially calculated and depends also on the mainboard and the configuration of the main memory used in this system. To estimate this value, the performance measurement tool Stream was used [4]. It is a commonly accepted benchmark in the computer-science community to measure the maximal practical bandwidth achieved on a computational system. On the NUMA platform the maximal bandwidth in Stream was about 90 $GB/s$. This is very close to the transfer rate of reading the $D_{ij}$ data by the optimization algorithm on NUMA systems. This shows, that the concepts introduced in section 4.2 do scale on shared memory systems using more than one processor. Also on NUMA systems, the number of cores is not important for the bandwidth, however the number of processors is. Each processor has its own memory domain and therefore its own dedicated connection to them. Adding more processors in the NUMA system would result in a runtime improvement of the planning algorithm. Unfortunately, the NUMA concept does not scale easily to a high number of processors. Commonly available systems cannot be configurated with more than 8 processors.
The bandwidths stated in table 4.1 are not the same for every plan. This is not due to the clinical context of a specific case, but due to the structure of the pre-calculated dose influence data ($D_{ij}$). Especially on the NUMA system it can be seen that the bandwidth is higher for photon plans containing a larger $D_{ij}$ matrix. The achieved bandwidth for the proton plan (plan 4) is significantly smaller than for for the photon plan. This is can be explained by the concept of data handling in $\mu$KonRad. According to listing A.2 the two inner loop of the dose calculation runs over the elements of each bixel in the plan. For a high data throughput it is important that the number of operations within the most inner loop is as high as possible and the amount of changing the computational context by a new loop is as low as possible. For the proton plan the number of bixel is approximately as high as for plan 3 but the number of dose influence elements is only 1/10th compared to plan3. Thus the ration between processing data and the overhead for changing the operational context is disadvantageous for the proton case. Figuratively speaking, the proton $D_{ij}$ is stretched too far by a high number of columns (bixels) while only having relatively few lines (number of influenced voxels).

The analysis of the runtime characteristic on clusters shows conceptual problem with the scaling of the IMRT planning concept.
The time line graph in figure 4.7 reveals two interesting aspects. First, the calculation of the projection kernel takes approximately twice the time of calculating the physical dose (back-projection) kernel. That was expected because calculating a beam weight kernel involves data from the dose cube segment and from the $D_{ij}$ matrix. The physical dose calculation only operates on the $D_{ij}$ matrix and a much smaller beam weights array. The time for finishing the objective function kernel (between the back-projection kernel and the projection kernel, marked by a red arrow) is very low compared to the other two kernels. Second, the communication process of the projection kernel seems to be slower than the communication of the objective function kernels. A closer look to the graph reveals that the first (blue) boxes of the beam weight kernel communication (e.g. the third communication process) are larger than the blue boxes of the other communication process (e.g. the second communication process). These boxes indicate the time in which one process is waiting for another slower process to provide its results. In our example process P0 finishes its calculation first while P0,P2 and P3 are waiting for the slowest process P1 to finish. The actual communication is marked by the following green and red boxes which only takes about 0.13 msec compared to 1.9 msec of waiting in P0. This example demonstrates that the communication itself is quite fast while the synchronization wastes a lot of time due to unbalanced work loads. Despite using the scheduling strategy described above a difference in the work load cannot be avoided in our parallel model. This is due to two reasons. First, the smallest unit of a segment is a whole dose cube slice. For a high number of concurrent processes, the dose segments only comprise a few slices and the segmentation is not fine enough to balance the work load. Second, the optimization algorithm may close or open a certain bixel from one iteration to another. This would change the work load for a small segments dramatically which can be seen in the time line of P4 between the first and third communication.

[4] http://www.cs.virginia.edu/stream

The graph in figure 4.9 reveals that a satisfying performance scaling of the algorithm can be achieved for less than 20 processes. If more processes are used (i.e. more CPU cores are involved) the speed up improvement gets smaller and more unstable. This is due to two reasons. First, the scheduling problem described in section 4.4 becomes important at a high number of processes. By having for instance 40 concurrent processes and 60 dose cube slices (plan 1 in table 4.2) each dose segment only consists of one or two slices. That dearth of fine-tuning results in an unbalanced work load among the processes. Second, the CPUs are designed to have four processing cores but only two L2 caches. So, two cores have to share one cache memory. Since treatment planning with pre-calculated dose influence matrices is a memory intensive endeavor this leads to a significant performance degradation. If less than 20 concurrent processes are used, the system can distribute the processes in a way that only 2 of them are assigned to one processor. That way each process could use its own cache memory and the performance scaling is as expected.
It must be concluded that the IMRT planning problem based on pre-calculated dose influence data does only scale for a relatively small number (below 100) of parallel processing unit. The communication overhead and the load balancing problem prevents this problem from scaling well on massively parallel CPU-based systems.

Other works investigate the performance of the planning problem on GPUs. Men et al. (2009) presented a GPU-based ultra-fast planning algorithm utilizing a gradient descent method in combination with Armijo's rule. It is known as one of the fastest implementation of the inverse planning problem on GPUs in the community. The authors report a runtime of 2.79 s for a clinical plan which is comparable to plan Nr. 1. Ziegenhein et al. (2013) concluded that the runtime of Men's GPU-based algorithm is at least in the same ball park as the runtimes achieved for the CPU-based implementation on a UMA processor as presented in section 4.2. For a multi-processor system the performance is even than achieved on a GPU.

## 4.7. Outlook

In the first part of this work, the conventional IMRT planning problem and a performance orientated state-of-the-art solution on modern CPU-based systems was introduced. The runtime of typical clinical treatment plans is discussed on three classes of computational systems architectures (UMA, NUMA and cluster systems). Nearly every commonly available computer based on Intel or AMD processors can be assigned to one of these classes. The performance of a GPU-based implementation was mentioned as an alternative platform to the CPU. The analysis of the planning problems shows two major findings. On the one hand, the solution presented in this work is a bandwidth limited algorithm. Thus, the runtime depends on how fast the clinical data can be transfered from the main memory of the planning system to the processing units. On the other hand, the problem does not scale on a larger number of processors. The combination of these two results leads to the conclusion that the performance of the conventional inverse planning problem has come to an end on modern computers. On multi-processor systems the full bandwidth of the data transportation system has been exploited. More bandwidth is only achieved by multiple computer systems which are organized as a cluster. Unfortunately, the problem does not scale with the high number of concurrent processes, so that the expected performance gain does not appear. Thus, the multi-processor NUMA system is indeed the most suited architecture to efficiently solve the inverse planning problem and it is not expected that the bandwidth of future shared-memory hardware increases significantly.
In order to achieve a significantly higher performance a new therapy planning paradigm has to be proposed which uses on the one hand more of the arithmetic capabilities of modern computer hardware and on the other hand reduces the amount of clinical data as much as possible to avoid the von Neumann bottleneck. Modern CPUs and GPU provide a huge potential of arithmetic performance which is mainly achieved by the parallel processing of data on several levels. However, this is only possible if the ratio of arithmetic operations to memory operations is sufficiently high enough. Modern multi-processors shared memory systems can deliver an arithmetic performance in the order of 1 tera floating point operations per second while the transportation of data is limited to about 100 giga bytes per second. Thus, an algorithm has to perform at least 10 basic

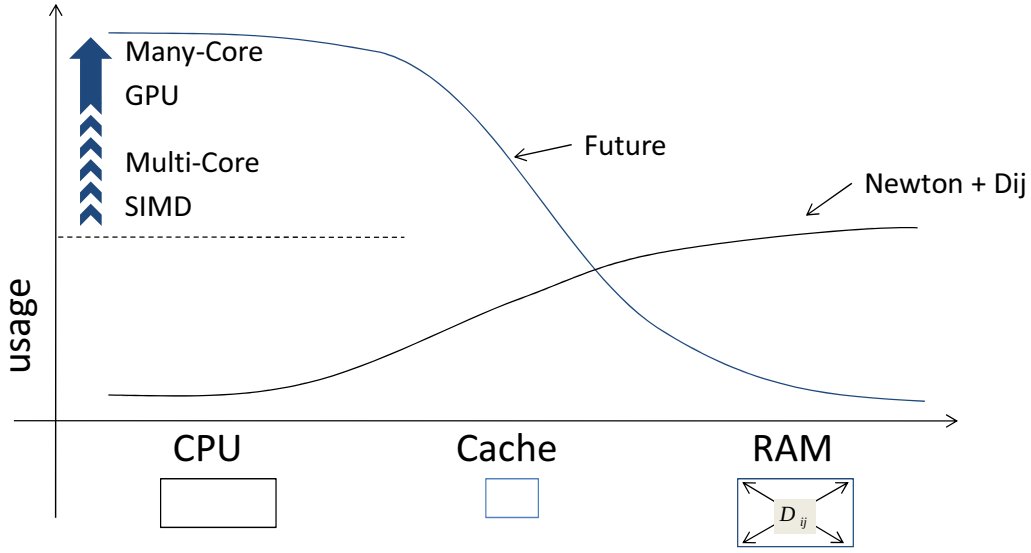



**Figure 4.11.:** Two opposing trends of using the hardware resources on modern algorithms.

arithmetic operations on a data word in order to exploit the potential of modern computer systems. According to the discussion in section 4.1 the realization of the therapy plan optimization based on Newton's method and a set of pre-calculated dose data only executes two basic arithmetic operations while processing 5 memory related operations. That is a very disadvantageous ratio. In order to achieve a higher computational performance, a new therapy planning paradigm has to be introduced which relies more on arithmetic operations than on memory operations. Figure 4.11 illustrates this demand. The conventional planning concept which is described in this chapter makes extensive use of the planning system's memory (RAM) and only uses a very small fraction of the arithmetic performance. The new planning paradigm which is presented in the next chapter only uses a small amount of clinical data and performs extensive arithmetic operations on them. To accomplish that, the concept of the pre-calculated dose influence matrix $D_{ij}$ has to be abolished. It is no longer feasible to pre-calculate physical dose data and store it in the memory. This concept was developed in the early days of IMRT planning when CPUs very slow and calculating data on the fly was not possible. The new planning concept features a new on the fly dose calculation method which is introduced in section 5.3.1. It relies on a relatively small set of data which is kept in higher cache memory for fast access. The dose is calculated using a convolution in spatial domain which performs a large amount of basic arithmetic operations in parallel on the small set of readily kept data. This would correspond the second curve marked "'Future"' in figure 4.11. The SIMD technology stands for the implementation of data parallelism on the instruction level which is extensively used for the new dose calculation. The multi-core architecture of modern computers enables a genuine thread-level parallelism which allows to process multiple instructions at the same time. The new planning paradigm introduces in the next chapter also makes used of massively parallel architectures like the GPU. The many-core architecture which features many integrated CPU-like cores on an additional device was unfortunately not readily available during the time of this work. However, this technology emphasizes the need for more arithmetic load in modern algorithms. The new planning concept is already compatible with this new technology.
Apart from the aforementioned drawbacks from a technical point of view, there are also some major drawback of the conventional planning method from the clinical point of view. These are discussed in section 5.1 to motivate the new planning paradigm any further. The extra performance potential which is gained through the paradigm shift on the computational side can be exploited to improve the planning concept from the clinical point of view as the next chapter demonstrates.

# 5. The Interactive Dose Shaping Paradigm

## 5.1. Why developing a new treatment planning paradigm?

The conventional method of deriving a treatment plan as described in chapter 4 has been used for more than 15 years in clinical routine. It relies on an interative (quasi-)Newton optimization procedure. The otimization is driven by a single- or multi-criterial objective function consisting of dose-constraints which are defined for a set of pre-segmented, organ specific volumes of interest (VOI). The optimal treatment plan is found by minimizing the objective function. This indirect approach of deriving a treatment plan suffers from various shortcomings which are unavoidable using this method. There are three main disadvantages which will be discussed in the following.

First, the objective function (equation 4.3) comprises a set of parameters which are clinically not intuitive. While the prescribed dose is expressed in the physical unit *Gy*, the trade-off between the planning goals is described by a numerical penalty factor which denotes the relevance of the corresponding prescription for the planning process. An example of the objective function parameters for a clinical prostate is shown in figure 5.1.

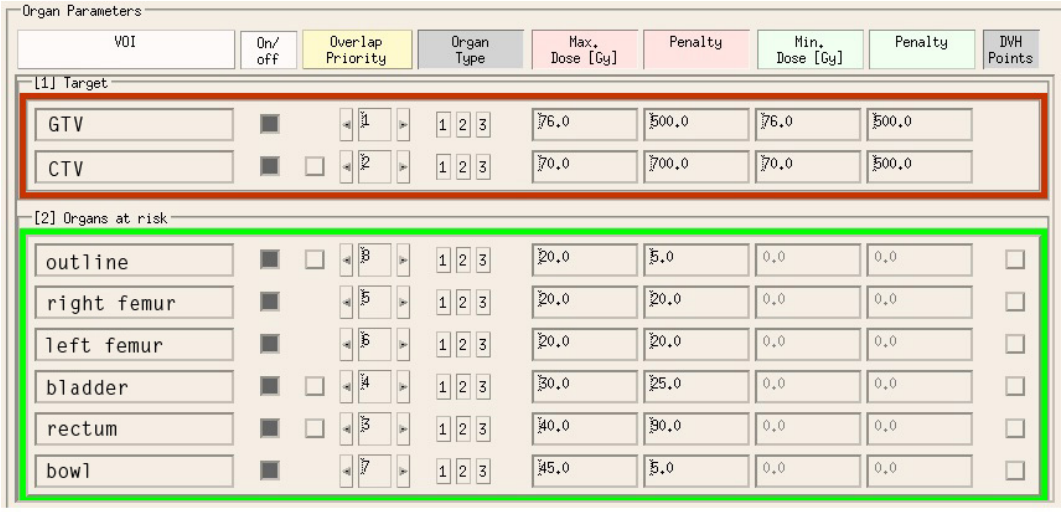



**Figure 5.1.:** The parameter list to define the objective function as presented to the user in the conventional treatment planning framework KonRadXP.

It shows a typical interface of a treatment planning framework presented to the user to control the parameter of the objective function. The organs are class-divided into targets (red box), organs at risk (green box) and unclassified tissue(black box). For each organ a minimal and a maximal dose can be specified. For target volumes, these two values are usually set to an equal number since it is believed to be advantageous to have a homogeneous dose distribution in the target. The minimal dose of an OAR is zero, since it is desirable to have no dose at all in these organs in the ideal case. The dose in the unclassified volume is not included in the objective function and therefore not controlled by the optimization. It is obvious that not all constraints can be fulfilled in a clinical treatment plan. For the prostate case shown in figure C.2(a) for instance (which is the same plan as loaded in KonRadXP to specify the parameters in figure 5.1) it is physically not possible to cover

the target volume with more than 70 Gy and at the same time having absolutely no dose to the rectum at all. The position of the dose gradient between these two organs is indirectly controlled by the penalty factors which can be assigned to each prescribed dose constraint. The dose prescription in Gy is a meaningful parameter to both, the therapist and the radiation physicist. Although, the penalty factor is not intuitive and has no direct clinical meaning. There is no obvious connection with the value of a penalty factor and the clinical outcome. The therapist finds a good set of parameter which result in a clinical acceptable plan through a trial and error approach: He starts with a first set of parameters which are derived from his experience. After a complete optimization process, the therapist evaluates the dose distribution and refines the parameters until an acceptable plan is found. This is a time consuming endeavor. One optimization process usually runs in the order of minutes. Several trials have to be made until the final plan satisfies the the clinical specifications.
Therapist cannot be sure what a refinement of the parameters changes to the plan until the optimization is finished. The indirect approach of using the objective function produces plans which are optimal due to the mathematical formulation of the problem based on dose constraints and penalty factors. However, it is not guaranteed that a mathematically optimal plan is also clinically optimal or even clinical acceptable.

Second, the control of local plan features is limited to the pre-defined volumes of interest. Dose constraints are defined for all voxels of a volume equally. It is for instance not possible to define a desired dose for only a part of the target volume or limit the dose exposure to only a part of an organ at risk. Having this limitation, the appearance of hot spots in the normal tissue is especially difficult to control in clinical practice. Hot spots are local dose escalation which only affect a few voxels of the plan. The normal tissue which comprises the whole patient except for the target volumes and the OARs contains a high number of voxels. A large deviation of a few of these voxels (as in a hotspot) only gives a small contribution to the objective function, so that the hotspot will not be eradicated by the optimization. From a clinical point of view these hotspots are a severe thread to the patient and may increase the toxicity to the healthy tissue. In an attempt to control the hotspots it is common clinical practice to define an artificial volume of interest within the region where the hotspot occurs. Doing that, it is possible to assign an individual prescribed dose and penalty factor to this region. Although this approach is very time consuming since the artificial volume has to be defined on every ct slice. Furthermore it increases the complexity of the plan.

Third, using the conventional concept it is difficult to adapt an already optimized therapy plan to a changed patient geometry as detected by IGRT methods (Jaffray et al. 2002). This is especially interesting in the field of adaptive radiation therapy (ART) (e.g. (Wu et al. 2002)). The adaption usually does not affect the whole, but only a certain part of the plan. For instance through the use of IGRT methods the locality of the prostate can be detected prior to the treatment. Here, the plan adaption will consist of a re-location of the target volume while the other treatment parameters do not change. With the conventional planning concepts the whole treatment has to be re-planned to incorporate the shift of the prostate. There is no means for applying partial changes to the therapy plan directly.

In addition to the three main drawbacks from the clinical side, there are several drawbacks from the computational point of view as described in section 4.1. The Newton optimization approach relies on the transportation of a huge amount of data in each iteration. This leads to a memory-bound problem that only shows a moderate performance on modern computers. In addition to that, the iterations are not independent from each and the results need to be synchronized after each step which makes a massively parallel implementation not feasible.

Several alternatives to the conventional planning method have already been propose. One of the most popular approaches is the multi-criteria-optimization(MCO) method (Craft et al. 2006, Halabi et al. 2006, Ogryczak 2002). MCO pre-calculates a Pareto surface of a given planning setup and lets the therapist determine the planning goal trade-off through the use of sliders for each dose prescription. This concept has been implemented into clinical planning frameworks. Although, it still relies on the concept of the objective function. It is not

guaranteed that the desired clinical plan is on the Pareto front.
Another work of Cotrutz and Xing (2003) proposed a dose shaping mechanism which allows to defined penalties not only to segmented VOIs but on individual voxels. This approach is an extension of the conventional therapy planning method and still relies on an iterative optimization and the definition of an objective function. A comparable direct planning approach as described in this chapter has not been introduced yet.

## 5.2. The concept of Interactive Dose Shaping (IDS)

In the last section it was discussed that the conventional approach to generate a radiation treatment plan has several unavoidable drawbacks which hamper the current and future advanced applications of radiation therapy. in this section the concept of the Interactive Dose Shaping (IDS) is presented. Some of these ideas have already been introduced in (Ziegenhein 2008). Nevertheless, a comprehensive description of the concept for a three-dimensional planning framework is originally presented in this work.

The development of the IDS paradigm is based on three main desirable features.
First, the new planning approach is designed to directly involve the clinical knowledge of a therapist. Using the conventional approach the therapist defines numerical parameters like target dose and penalty factors and the iterative optimization derives a plan which is optimal according to the objective function. In contrast to that, the IDS method should involve the therapist during the process of plan optimization. The automated optimization block of the conventional optimization which is guided by a mathematical notion is split up into a sequence of sub-steps. The therapist interactively defines the sequence of sub-steps and controls the impact of each step on the therapy plan. To make this approach feasible, the modifications made in each sub-step must be implemented instantly into the plan, so that the therapist can evaluate its effect and propose a a next action step based on this evaluation.
Second, it should be possible to adapt existing, pre-optimized therapy plans for small changes. This is a desirable feature in the field of adaptive radiation therapy. Using the conventional planning method, this is not easily possible. Even small variations to the geometry or to the dose distribution of the plan change the objective function and another iterative optimization process needs to be executed. With the strategies for a fast plan generation presented in chapter 4 a new plan can be optimized within seconds, so the runtime is no longer an argument here. Although, a plan optimization via the conventional method is a global optimization which affects all planning features. Thus, it is possible that already established features elsewhere in the plan change, too. All these unintentional changes have to be re-assessed by a clinician, which is a time consuming process. Thus, it is an important requirement that the plan modification made in one sub-step has an effect only on a limited locality of the plan.
Third, the implementation of the IDS concept must be designed to make use of modern computational hardware. In section 4.7 it was concluded that the conventional optimization is a strong memory-bound problem and does not exploit the capabilities of a modern CPU or GPU. The IDS concept should rely on algorithms which scale well on parallel architectures and ideally are compute-bound methods. A high performance implementation of the IDS concept is necessary to guarantee the interactive character of the concept. A local planning feature should be imposed instantly, i.e. in real-time, so that the therapist can immediately evaluate the impact of his action. As discussed in section 4.7 neither the quasi-Newton optimization method, nor the physical dose calculation based on a pre-populated dose influence matrix is able to achieve a high computational performance on modern hardware. So both methods have to be replace.

The Interactive Dose Shaping method introduced in this work describes the realization of a local plan modification as it is carried out within one interactive sub-step. The conceptual challenge of realizing a local plan modification is to constrain the impact of it to a certain locality in the treatment plan. Just imposing the modification itself is relatively easy, since the effect of changing the radiation fluence amplitudes to the dose distribution in the patient is well understood. Using the formulation of equation 2.2 the dose in the patient is

simply derived as a matrix multiplication between the dose influence values and the bixel weights of a radiation fluence. In contrast to that, constraining a dose modification to a certain locality is a difficult endeavor. It requires to understand the correlation between the amplitudes of the radiation fluence and the dose distribution in both direction. The dose pattern $d_i$ would be the input parameter set of this algorithm and the output would be the bixel weights describing the desired radiation fluence. A mathematical notion of this problem would be derived by solving equation 2.2 for the weights $w_j$. This implies the construction of the inverse dose influence matrix $D_{ij}^{-1}$. Generally this is not feasible due to several reasons: First, the dose influence matrix is sparsely populated and generally not of full rank. Second, the matrix can be very huge. The dimension of this matrix equals the number of bixels raised to the second power. The number of bixels used in a treatment plan can easily be in the order of 10000. Third, negative bixel weights are not allowed. This would correspond to the existence of a radiation modality which erases already applied dose from the patient. Thus, even if an inverse of the dose influence matrix is calculated, a post-processing step is needed to eliminate the negative bixel weights. Every modification to the plan which is intended to affect only a certain locality will inevitably have a more global influence on the plan. In order to avoid that, a two-step approach for IDS is introduced: dose Variation and Recover[1]. The dose Variation imposes the local planning feature which is desired by the therapist. The inevitable, implicit global changes to other localities in the plan is compensated in a subsequent Recovery step. The Recovery is an automated algorithm which first identifies locations in the therapy plan which have been affected unintentionally and second tries to eradicate these deviations. The process of one interactive dose shaping step is pictured in figure 5.2. The blue boxes indicate an interaction between the IDS engine and the

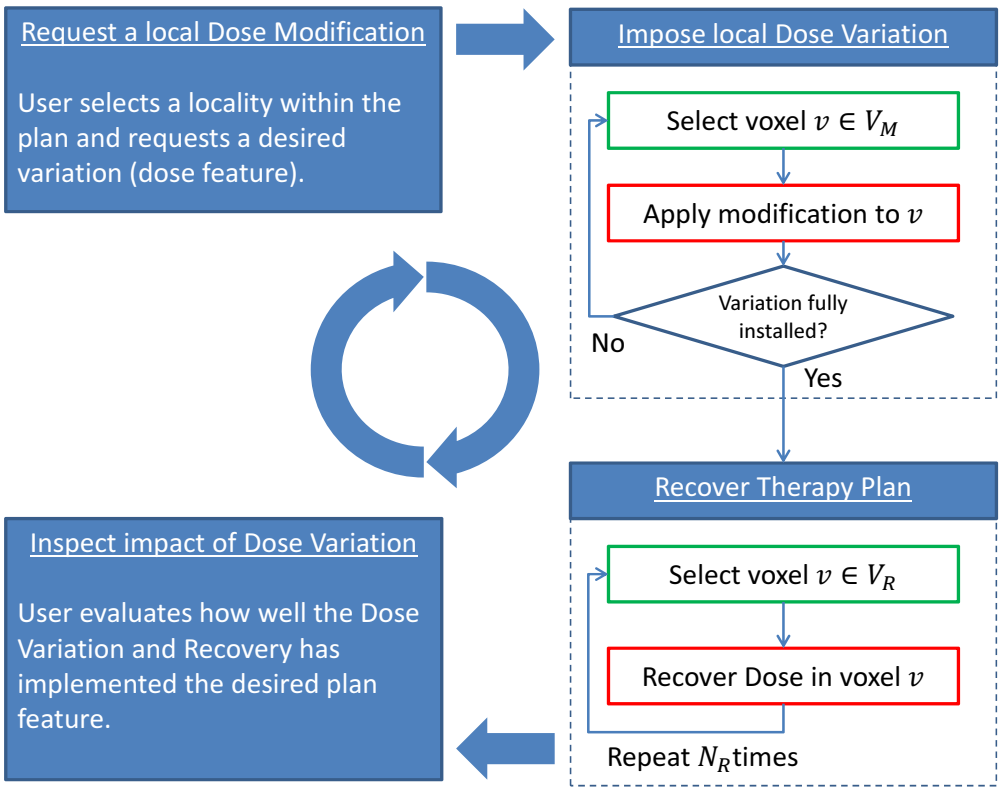



**Figure 5.2.:** Workflow of the Interactive Dose Shaping paradigm. The left two boxes in blue define the user interface to the IDS engine (right side) containing the Variation and Recovery module.

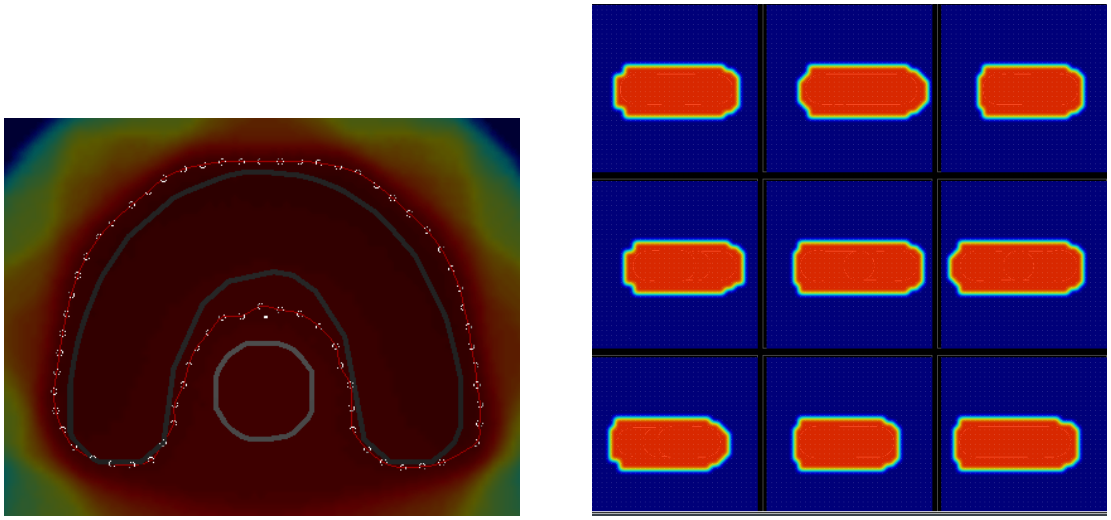
therapist (user). First, the user selects a locality of the plan to which a desired dose feature should be imposed

[1] Variaton and Recover denote the two-step process in IDS and are therefore written with a capital letter.

to. Usually the user requests a higher or lower dose in a fraction of a volume of interest. The two boxes on the right side describe the basic Variation and Recover strategy of IDS.
The IDS process starts with a Variation to realize the desired modification. This is achieved by a strategy which operates on individual voxels of the plan. An algorithm identifies the set of voxel $V_M$ which comprises the desired Variation locality. All voxels in $V_M$ do have a dose value which is in conflict with the desired feature to be imposed. The IDS Variation strategy selects one voxel $v$ from $V_M$ and applies a modification to this voxel in an attempt to resolve this conflict. This is achieved through a Manipulation[2] of radiation fluence amplitudes which contribute dose to $v$. Due to lateral scattering of the incident radiation there will be more than only one amplitude which causes a significant dose deposition to the selected voxel $v$. Consequently, adapting the dose in voxel $v$ is an ambiguous task which can be realized by various fluence Manipulations. The effect of the Manipulation is back-projected instantly into the dose pattern of the patient. It will also change the dose value of voxels which are in close proximity to the selected voxel $v$. Usually this is in the interest of the user who wants to impose a dose feature to a certain region of the plan and not just to one voxel. If the region of Variation is large it may be necessary to pick more than one voxel from $V_M$ and perform another fluence Manipulation to fully install the desired dose feature. The voxels are modified sequentially while the dose is updated after each modification.
The Manipulation of fluence amplitudes has a global effect on the therapy plan. It does not only changes the dose contribution to voxels in the locality of Variation (which was intended), but the dose of every voxel which is influenced by the manipulated fluence amplitudes will change inevitably. In order to keep the dose modulation to the desired locality and preserve already established dose features these unintentional changes have to be compensated. This is done by the IDS Recovery module (see figure 5.2). The Recovery identifies voxels $V_R$ of the therapy plan which have been affected unintentionally by the Variation process. A limited number of $N_R$ voxels are recovered by applying a Manipulation of fluence amplitudes which on the one hand restore the original dose of voxels in $V_R$ and on the other hand preserve the Variation which has been just installed moments ago. The recovery of one voxel also comprises a dose update calculation based on the radiation fluence Manipulation. After the Recovery process is done, the user can immediately inspect the new plan (last blue box in figure 5.2). Based on the results, other dose shaping processes can be launched to impose new dose features elsewhere in the plan.

To demonstrate the paradigm depicted in figure 5.2 an basic example for a Dose Variation and Recovery (DVR) process is visualized in the following. In addition to that a brief mathematical formulation of the dose-fluence relation is provided to illustrate the process. As discussed earlier the IDS concept does not use the pre-calculated dose influence matrix $D_{ij}$. However, the $D_{ij}$ approach is well suited to describe the relation between fluence amplitudes and dose in the patient. For that reason a conceptual dose influence matrix $\mathcal{D}_{ij}$ will be used i the following. It describes the dose influence of a discrete fluence amplitude $j$ to a voxel $i$ in the plan. Although, in contrast to the matrix $D_{ij}$ it is not used for the actual physical dose calculation in the IDS planning system.
The DVR example is taken out on a simple geometrical phantom (see appendix C.1). It consists of a symmetric horse-shoe shaped target volume and a spherical organ at risk which is situated in the emargination of the target. The placement and shape of these volumes is a geometrical exaggeration of a real prostate tumor case or a difficult to irradiate clivus chordom case.
The starting point for the DVR process is an unoptimized three-dimensional plan as shown in figure 5.3(a). The color coding in this picture shows the physical dose of the plan. The prescribed dose (i.e. the therapeutic dose which is desired to deliver to the target) is highlighted with the red iso-dose line in the transversal slice of the plan. As the image shows, in this initial state the target volume is fully covered by the prescribed dose. This comes from the radiation fluence maps which are arranged to be conformal with the target volume. In figure 5.3(b) the fluences are shown for all nine incident beams. The color coding of the fluence amplitudes show, that the target is fully irradiated but no fluence modulation is made to spare the spherical organ at risk.

[2]The Manipulation of fluence amplitudes as a result of Variation or Recovery is also denoted with a capital letter.



(a) (b)

**Figure 5.3.:** The initial planning state of the phantom case. a)A transversal slice through the phantom geometry which shows a homogeneous dose coverage of the target. The iso-dose line denotes the desired prescribed dose to the horse-shoe shaped target volume. b) The homogeneous dose is achieved by nine beams having a conformal but unmodulated fluence map

The dose $d_i$ in voxel $i$ can be described by:

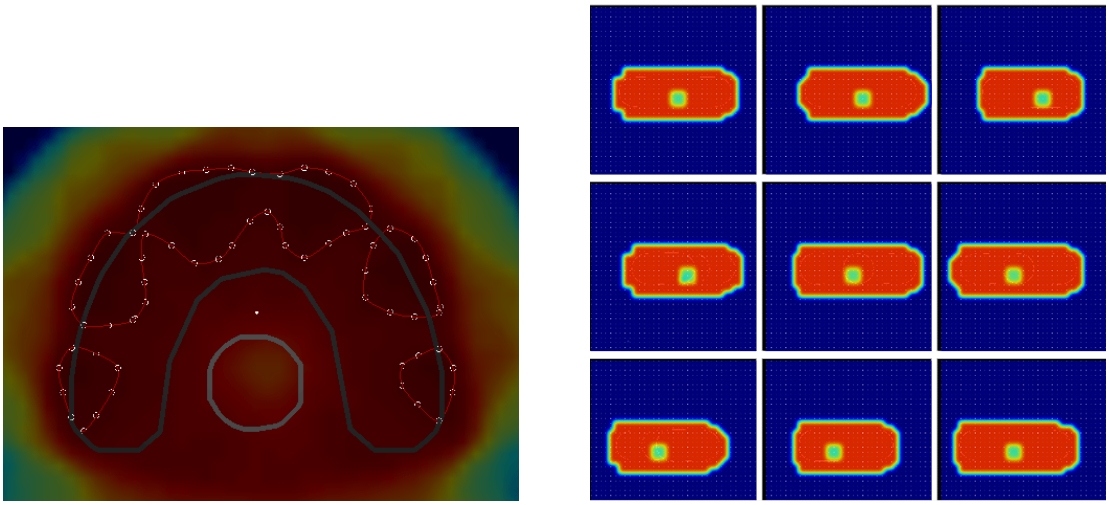
$$d_i = \sum_{k=1}^{9} \sum_{j_k \in I_k} \mathcal{D}_{i j_k} f_{j_k} \tag{5.1}$$

The first summation denotes the superposition of the incident beams $k$. The phantom is planned with 9 beams. The second sum is taken out over all fluence amplitudes $I_k$ of beam $k$. The value of the amplitudes $f_{j_k}$ are multiplied with the conceptual dose influence matrix to derive the physical dose distribution $d$. The conformal non-modulated fluence maps in figure 5.3(b) are achieved by assigning a constant value to all fluence amplitudes which are situated in the projection of the target geometry onto the fluence map. All other value are zero. A similar starting point is chosen for the conventional Newton plan optimization, too.

The simples form to impose a modification on this plan is to request a dose change in only one voxel. Since the OAR receives too much dose, an improvement of the plan would be achieved by requesting a lower dose to a voxel $c$ in the OAR. The user request is imposed by the Variation module and the result is shown in figure 5.4(a) The picture clearly exhibits the requested cold spot in the upper right region of the organ at risk. According to the diagram in figure 5.2 the Variation was achieved by selecting a series of voxels from the set $V_M$ and performing a Manipulation of fluence amplitudes associated with these voxels. Since the user request was limited a priori to one voxel, the process of selecting a voxel from $V_M$ is trivial ($V_M$ only contains $c$) and the loop over multiple voxels does not exist. The Variation $\Delta d_c$ is installed by applying a Manipulation $\Delta f_{i_k}$ to a set $L_k$ of fluence amplitudes which contribute dose to voxel $c$:

$$(d_c + \Delta d_c) = \sum_{k=1}^{9} \left[ \sum_{j_k \in I_k} \mathcal{D}_{i j_k} f_{j_k} + \sum_{j_k \in L_k} \mathcal{D}_{i j_k} \Delta f_{j_k} \right] \tag{5.2}$$

The resulting dose $\tilde{d}_c = d_c + \Delta d_c$ is supposed to install a part or all of the dose shaping request. The amplitudes in $L_k$ are situated in the area of the projection of voxel $c$ to the fluence map of radiation beam $k$. In figure 5.4(b)



(a) (b)

**Figure 5.4.:** A Variation was imposed to the phantom geometry plan which results in a cold spot in the OAR (slightly green area in the upper right section of the OAR). b) The cold spot is installed by reducing the value of fluence amplitudes which are situated in the projection of the Variation onto the fluence map.

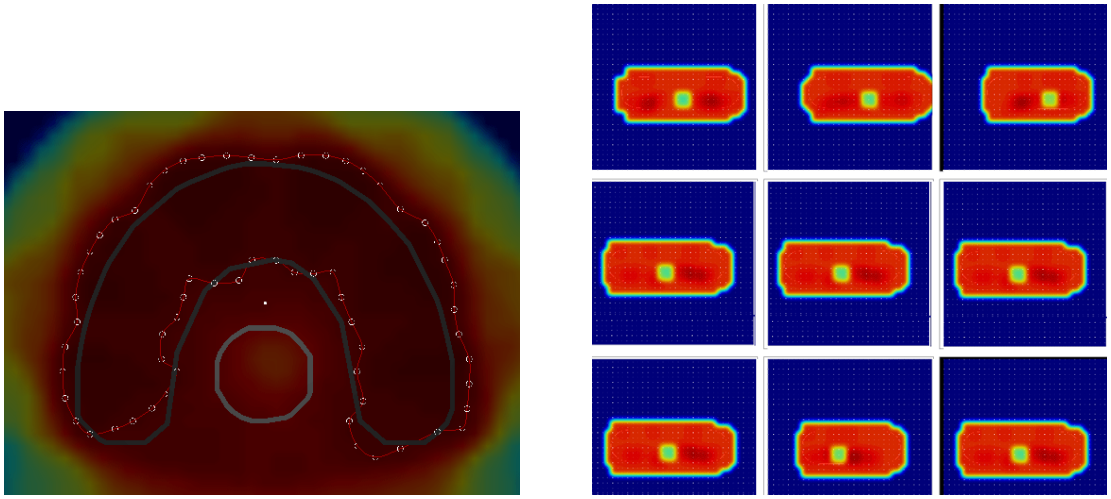
this area creates a cold spot in the fluence map. This is expected since the user requested a Variation to a lower dose value in voxel $c$. The size and shape of the fluence Manipulation dictates the form of the dose Variation in the selected voxel $c$. For this example the Variation is supposed to be limited to a very small locality (only one voxel) so only a few amplitudes are manipulated. Even though, the Manipulation of the fluence maps is very small, it not only installs the dose shaping request, but also has a global effect on the dose distribution of the plan. Along the pathways of the incident radiation beam, voxels get under-exposed due to the cold spots in the fluence maps. This can be see in figure 5.4(a) on the shape of the prescribed dose iso-line. It does not cover the whole target volume anymore (as in figure 5.3(a)). Dose exposure is missing in voxels which are situated in the projection of the fluence cold spots (i.e. the direct pathway of the beam) and on the inner boundary of the target. The effect of the Variation on the inner boundary of the target is especially high, because the iso-center where the beams are focused at is situated just above the OAR (white dot). Due to the inverse square law which describes the intensity of the dose deposition in this beam geometry, voxels near the iso-center are significantly more affected than voxels farther away.

The under-exposure of the target as a side-effect of imposing the dose shaping request is compensated by the dose Recovery process. The Recovery process first identifies a set of voxels which are subject to be recovered and subsequently manipulates the fluence maps to re-install the original dose value (see figure 5.2). For the phantom case discussed here, the set $V_R$ comprises all the voxels in the target volume which are not enclosed by the iso-line shown in figure 5.4(a). All these voxels have been unintentionally under-exposed by the Variation process before. The Recovery selects a voxel $r$ from $V_R$ and performs an *inverse Variation* on it. That means, the fluence amplitudes are manipulated so that the original dose in $r$ is re-installed:

$$(d_r + \Delta d_r) = \sum_{k=1}^{9} \left[ \sum_{j_k \in I_k} \mathcal{D}_{i j_k} f_{j_k} + \sum_{j_k \in O_k} \mathcal{D}_{i j_k} \Delta f_{j_k} \right] \tag{5.3}$$

Consequently, in order to compensate the cold spot created by the Variation process of this example, the Manipulation in the Recovery process leads to higher fluence amplitude values (dark red spots in figure 5.5(b)). Ideally the intersection of the set of fluence amplitudes $O_k$ which is used for the Recovery with the set $I_k$ is empty,

so that the dose feature imposed to voxel $c$ by the Variation is not revoked. However, even if $L_k \bigcap O_k = \emptyset$, it cannot be concluded in general that the dose in voxel $c$ remains completely unchanged, since the dose in voxel $c$ is constructed by more fluence amplitudes than used for the Variation. It is $L_k \subseteq I_k$ and it is possible that $L_k \bigcap O_k = \emptyset$ but $O_k \bigcap I_k \neq \emptyset$.
The Recovery process is applied to as many as $N_R$ voxels subsequently, while the dose distribution is updated after recovering each voxel. The number $N_R$ of voxel to be recovered is chosen, so that the response time to the shaping request is low enough to allow an interactive planning in real-time. The resulting dose distribution is presented to the user immediately after the Recovery process is finished. For the example chosen to demonstrate the DVR process, the dose distribution after 32 Recovery steps is show in figure 5.5(a). The iso-line of



(a) (b)

**Figure 5.5.:** The target volume of the phantom case was recovered for unintentional changes of the dose distribution. a) The Recovery almost re-installs the iso-line to cover the whole target. b) The Recovery is realized by various fluence Manipulation which can be recognized by the dark red areas.

the prescribed dose reveals some cold spots in the target which could not be recovered. Based on this result, the therapist can assess if the dose distribution is still clinically acceptable so that he can continue planning by requesting another local dose feature. If the therapist decides that the dose distribution is not acceptable, the last request can be reversed and the therapist has the possibility to request another dose feature or the same feature but with a less strong Variation on the plan.

The realization of the IDS concept poses three main challenges:
First, adequate strategies have to be introduced for the Variation and Recovery module as shown in figure 5.2. Rules have to be defined to drive the selection of a voxel for the Variation or Recovery process on the one hand and control the Manipulation of the fluence amplitudes on the other hand. Both aspects are combined to the *IDS engine* which is discussed in section 5.4 and 5.5
The whole concept of Interactive Dose Shaping is based on the assumption that local dose features which are requested by the user can be implemented within an interactive time frame. This is vital, since each IDS step must be manually triggered by the user. The planning process which consist of multiple related IDS steps is only feasible if the user does not have to wait an annoying amount of time between them. From a computational point of view, the response to a dose shaping request must be done in real-time. The real-time constraint in IDS means, that the whole process of retrieving the dose shaping request, imposing the Variation and Recover the unintentional global changes must be done in less than 1 second. This is a challenging endeavor. To accomplish that, all modules of the IDS engine must be designed to operate as efficient as possible on modern computa-

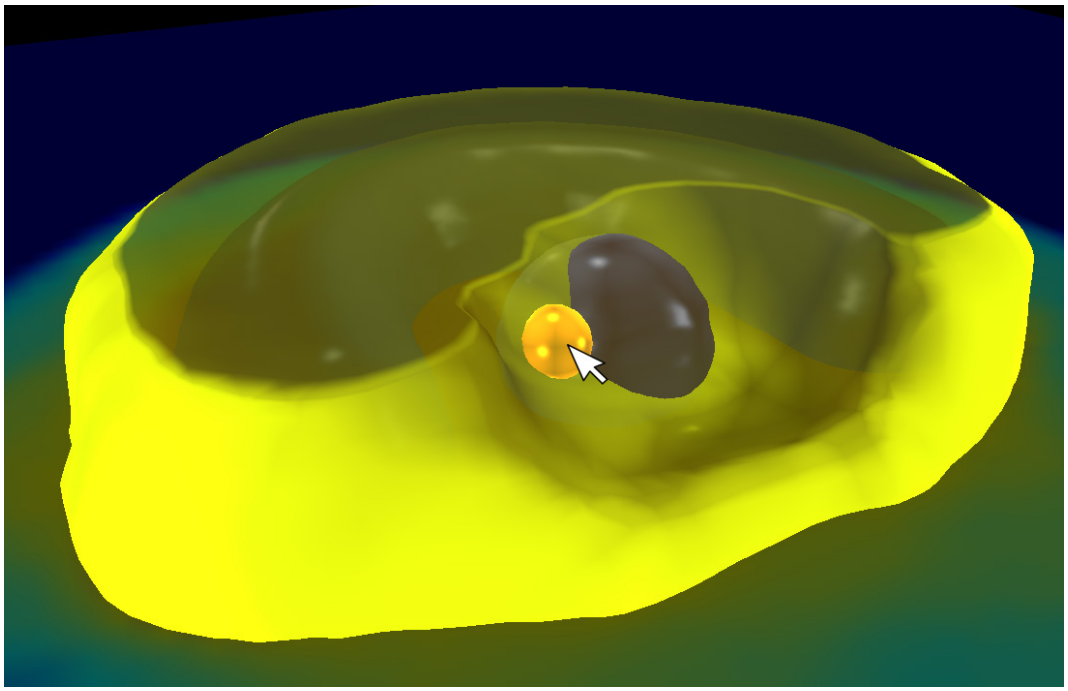


**Figure 5.6.:** Interface of the iso-surface shaping tool in IDS. An arbitrarily chosen iso-surface is deformed by a hard ball (golden sphere).

tional hardware to complete their tasks in the order of milliseconds. The module which presumingly takes the most time is the dose calculation. It is triggered after imposing the dose feature on each voxel and after each Recovery step (see figure 5.2). In section 4.7 it was concluded that the dose calculation method based on the pre-calculated dose influence matrix is not suited to use the capabilities of modern HPC devices. Furthermore, it relies on the assumption that the underlying patient geometry does not change during the course of planning, an assumption which would not allow dynamic re-planning. Consequently, a new dose calculation method has to be introduced which is on the one hand optimized to implemented localized Modulation of the fluence map efficiently and on the other hand exploits modern computational hardware features as much as possible to produce low runtimes. Developing such a dose update calculation method is the second main challenge in IDS and is described in section 5.3.1

The IDS engine and the dose calculation module are placed within an IDS planning framework. This framework was especially designed for the Interactive Dose Shaping method. It has the ability to handle clinical data, provides powerful three-dimensional plan visualization tools and enables the user to impose dose shaping requests via an intuitive to use graphical user interface. For the basic DVR example explained earlier in this chapter the Variation was expressed as *impose dose to a voxel* (see figure 5.4). As the name suggest, this Variation method imposes an arbitrary dose to just one voxel in the plan. For planning real clinical case this method is not well suite, since it would take too long designing a dose distribution by imposing dose to individual voxels. For the clinical planning scenarios presented in section 5.7 an *impose dose to a VOI* technique is used which selects a number of voxels from a VOI and imposes a uniform dose to them. The selection of the voxels is realized by the voxel selection module described in section 5.4.
Other Variation tools which are currently investigated are: the manipulation of the DVH curve and the direct shaping of an arbitrarily chosen iso-surface. The interface for the latter tool is depicted in figure 5.6. It shows the three dimensional phantom geometry which is overlayed by a semi-transparent iso-dose surface. In the initial state (figure 5.3) the iso-surface covers the whole target volume and the OAR. Via a surface manipulation

tool, the therapist can impose a Variation by shaping the iso-surface directly. To realize that, a hard ball (golden sphere in the figure) which is dipped with one hemisphere into the iso-surface can be dragged around the hull. By a mouse click from the user, the voxels which intersect the covered hemisphere of the hard ball are subject to a Variation which imposes to them a dose lower than the iso-value. The result is a new iso-surface which excludes the area of the hard ball. Using this tool, a therapist can intuitively "dig out" the OAR from a certain iso-value.
Further descriptions of such sophisticated Variation tools are explicitly excluded from this work since it is the topic of another ongoing PhD project. The Variation tools belong to the interface depicted as the first blue box in figure 5.2. The focus of this work is the design and the implementation of the dose Variation and Recovery strategy which is shown on the right hand side of figure 5.2. However, the basic structure of this framework is briefly introduced in section 5.6
Before the IDS Variation and Recovery is discussed, a concept for a fast dose update calculation is introduced in the next section.

## 5.3. The IDS dose update calculation

### 5.3.1. A concept for low-latency, real-time, ultra-fast local dose update calculations

To implement the concept of the local dose shaping paradigm introduced in section 5.2 a fast dose calculation algorithm is needed. Small local Manipulations of the fluence amplitudes proposed by the IDS concept must be translated to the dose distribution very quickly. This is important to evaluate the modification steps interactively, in real-time. The conventional dose calculation methods are not well suited for that task. That is why a new dose calculation method is proposed which is explicitly designed for the new planning paradigm.

The dose calculation algorithm for the IDS concept has three major differences to conventional dose calculation methods as for instance introduced in section 3. First, the fluence Manipulations proposed by IDS will always be spatially limited to a certain locality of the whole radiation field. Thus, the dose pattern will only change for a subset of voxel and not for the whole plan. A complete re-calcuation of the dose distribution is not needed. It is merely considered as a dose update calculation which refines the existing dose distribution according to the imposed fluence Modulation. Second, the dose update calculation has to be done very quickly. According to the estimation in section 5.2 approximately 20 subsequent Recovery steps are assumed to compensate unwanted dose changes outside the considered local area of the IDS Variation. The whole process of one dose Variation and approximately 20 Recovery steps must be carried out in real-time (about 1 second). Since, each Variation or Recovery process requires a dose update, the dose calculation must be done in less that 50 milliseconds (assuming 20 steps and still achieve real-time response). That is more than one order of magnitude faster than the pencil beam algorithm introduced in section 3 can provide a dose update for the whole patient. So, the locality of the fluence Modification must be exploited. Third, the quality of the dose update calculation does not need to be as accurate as the dose calculation which is used for preparing a clinical deliverable plan. Within the process of Variation and Recovery the updates are used to evaluate the fluence Modifications and propose the parameters for the next step. In this intermediate planning state, it is not necessary to provide dose at clinical quality. Although, the accuracy must be high enough, so that the IDS engine can propose a meaningful next step.

Taken into account all these requirements, the dose update calculation for IDS must be based on the pencil beam method. Using a method which provides a higher accuracy, like the Monte Carlo method (Jia et al. 2011; 2012) or the superposition method (Keall and Hoban 1996, Hissoiny et al. 2010) does not seem reasonable for two reasons. First, the calculation time would be too long. Second, the high quality is not needed in the intermediate steps of Variation and Recovery. Although, after a series of local dose shaping processes and before the final plan gets delivered, it is necessary to re-calculate the whole dose distribution using a superposition or Monte Carlo method. However, this is not part of the IDS engine and needs to be done in an independent

step prior to the delivery of the plan. According to the discussion in chapter 4.7 it is not reasonable to use a pre-calculated dose influence matrix approach for the IDS paradigm. This concept generates a memory-bound problem which threatens the scalability of the IDS concept and the real-time character (see figure 4.11).
In section 3 the equation for calculating the dose distribution based on a set of measured singular value decomposed pencil beam kernels according to Bortfeld et al. (1993) was introduced:

$$D'_{irreg}(x_p, y_p, d) = \sum_{i=1}^{3} D'_i(d) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Psi'(x,y)F(x,y) \times w_i(x - x_p, y - y_p)dxdy \tag{5.4}$$

For a practical realization the convolution terms are computed in frequency domain using the fast Hartley transform (FHT). Anyway, the convolution even for small fields takes up to a few seconds on modern CPUs. This is because a large number of transformations between frequency and spacial domain has to be computed. A convolution is realized by transforming the two dimensional material ($\Psi'(x,y)F(x,y)$ and $w_i(x,y)$) into frequency domain, then multiply the result and afterwards transform the result back into spatial domain:

$$C(x_p, y_p) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Psi'(x,y)F(x,y) \times w_i(x - x_p, y - y_p)dxdy = \mathfrak{F}\left\{\mathfrak{F}\left\{\Psi'(x,y)F(x,y)\right\} \times \mathfrak{F}\left\{w_i(x,y)\right\}\right\}(x_p, y_p) \tag{5.5}$$

with the operator $\mathfrak{F}\{\}$ being the fast Hartley transform. For a nine beam plan a total number of 63 transformations [3] have to be calculated.
Using the FHT for the convolution, the integral is solved for every point $(x_p, y_p)$ at once. This is beneficial if the whole dose distribution needs to be calculated. For a local dose update calculation as it should be used in the IDS engine, only a subset of all voxels in the patient is affected. The small fluence Modifications proposed by the Variation and Recovery steps are spatially limited and it is generally not necessary to re-calculate the convolution for every point $(x_p, y_p)$.

This observation is exploited to achieve the required runtime improvement over one order of magnitude for the dose update calculation. Instead of using the fast Hartley transform, the convolution $C(x_p, y_p)$ is solved directly in discretized spatial domain:

$$C(x_p, y_p) = \sum_x \sum_y \psi(x,y) w_i(x - x_p, y - y_p) \tag{5.6}$$

The terms $w_i(x - x_p, y - y_p)$ (which will be refereed to as *kernel weights* in the following discussion) are discretized according to Bortfeld et al. (1993). A two dimensional array with a fixed size (typically 256 x 256) is allocated. The point of origin is set to the center of the material (point $(128, 128)$) and each pixel gets a value from the function shown in figure 3.2 according to its distance from the point of origin. The term $\psi(x,y)$ is the product of the primary fluence $\Psi'(x,y)$ and the transmission function $F(x,y)$ which is discretized on an array of the same dimension as the kernel weights. The advantage of solving the problem in spatial domain is that small changes in the fluence $\psi(x,y)$ can be implemented directly in the convolution without rebuilding the complete material $C(x,y)$. Figure 5.7 shows an example on a transversal plane of an abstract phantom geometry. The figure shows the intensity modulation of one radiation beam which delivers dose to a patient geometry outline. The patient geometry consists of a target volume (red area) and an organ at risk (OAR, green area). Distance and size of the geometry is not to scale. The original fluence $\psi(x,y)$ (for which the convolution is already calculated) is modulated in the region that projects to the OAR (blue area). This effects the convolution material $\Delta C$ on the blue scale at the bottom of the figure. Not only the locality which corresponds to the fluence Manipulation directly changes (between the blue dotted lines) but also a area around this locality has to be recalculated. This is due to the finite range of the convolution kernels which transport scattered particles to the surrounding material. Nevertheless, only a relatively small part of the whole convolution needs to be

[3] for each beam 1 FHT to transform $\Psi'(x,y)F(x,y)$ into frequency domain, 3 FHT to transform $w_i(x,y)$ and another 3 transformation to bring the result back to spatial domain

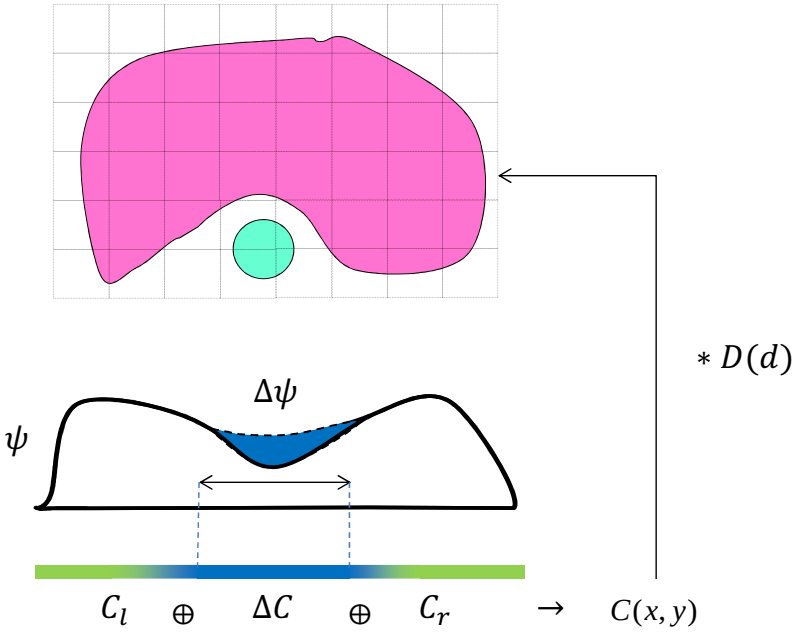



**Figure 5.7.:** A transversal slice showing a phantom geometry (red target with green organ at risk) and the radiation fluence $\psi(x,y)$ of one beam. A local fluence Manipulation $\Delta\psi(x,y)$ is imposed which requires a re-calculation of the corresponding amplitudes in the convolution material (blue).

recalculated and the areas $C_l$ and $C_r$ can be reused to updated the new material $C(x,y)$. The dose in the patient geometry is then derived by multiplying the appropriate amplitudes from the convolution materials with the depth dose factors $D_i(d)$ (Figure 3.2). For a spatially limited Manipulation of the fluence this reduces the dose calculation time significantly. Instead of re-calculating the complete convolution only a small area $\Delta C$ of the convolution material must be updated:

$$\Delta C = \left\{ C(x_p, y_p) \; with \; x_0 - h \leq x_p \leq x_1 + h \; and \; y_0 - h \leq y_p \leq y_1 + h \right\} \tag{5.7}$$

while the area of the Manipulation in a three-dimensional planning setup is the rectangle between the points $(x_0, y_0)$ and $(x_1, y_1)$. To account for the range of the dose kernels, a margin $h$ is taken into account for the re-calculation. The margin $h$ corresponds to the area between the blue and green scale at the bottom of figure 5.7. The area $\Delta C_l$ and $\Delta C_r$ of the convolution material is not affected by $\Delta\psi(x,y)$ and can be re-used from the previous calculation. Choosing a good value for the margin $h$ is very important for the runtime and accuracy of the algorithm. If the margin is too small, a reasonable amount of dose scattering is neglected especially on the edges of the Manipulation area. However, if the margin is too broad, the runtime of the dose update calculation will be too high.

Despite exploiting the locality of the dose update problem, the algorithm turned out to be too slow to fulfill the real-time constraint within the IDS engine. In order to further reduce the runtime, only a small set of points is re-convoluted and the full convolution material $C(x,y)$ is then interpolated using a B-spline method. Figure 5.8 shows an example. Instead of updating every point of the convolution material which is affected from the Manipulation (as shown in figure 5.7) the convolution is re-calculated only for a few points (blue arrows) from the modified part of the fluence. Another set of sample points is taken from the areas of the convolution material which is not affected by the Manipulation (dotted, black arrows). The effort of taking these points is

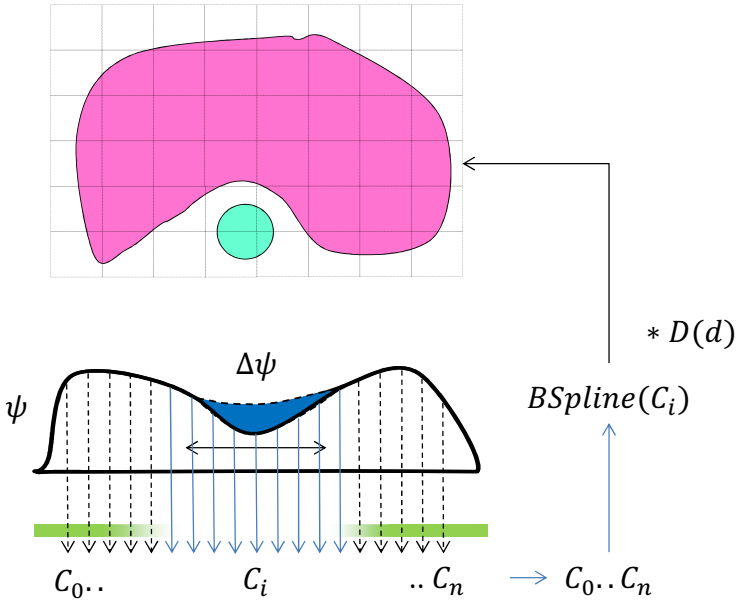



**Figure 5.8.:** Dose calculation using the B-spline model: Only a limited number of points $C_i$ of the convolution is re-calculated. The full material is then interpolated using a two-dimensional B-spline method.

very low, since they do not have to be calculated but only copied from the current convolution material. The complete set of points $C_0..C_n$ contains the re-calculated convolution points from the area of the fluence Manipulation (red arrows) and the copied sample points from the unaffected areas (black, dotted arrows). A new convolution material $C(x, y)$ is then generated from these sample points with the help of a B-spline interpolation.

For the use in the IDS engine, both the radiation fluence and the convolution material are represented by a two-dimensional bi-cubic B-spline. The B-spline interpolation has to be calculated in two-dimensional space, since the radiation beams are two-dimensional arrays of fluence amplitudes for a real clinical planning setup. The sample points are arrange on a static, equidistant grid. The size and structure of the grid is constant throughout the planning process. Figure 5.9 shows a fraction of the rectangular grid over a discretized two-dimensional radiation fluence. Solid dots represent the fluence amplitudes. Encircled dots are control points of the B-spline representation which are usually set after each 5 fluence amplitudes. The control points define a value list $(x_i, y_j, f_{ij})i = 1, 2, ..., nj = 1, 2, .., l$ with $f_{ij}$ being the fluence value at the control point $(x_i, y_j)$. The grid for the representation of the convolution material $C(x, y)$ has the same discretization, the same dimension and the location of the control points coincide with these from the fluence representation. Please note that the picture only shows an arbitrary fraction of the grid. Thus, the first control point in the top-left corner does not have the coordinates $(x_1, y_1)$.

The data between the control points (marked as dots in figure 5.9) is interpolated by a two-dimensional B-spline function $S(x, y)$:

$$S(x, y) = \sum_{\nu=1}^{n} \sum_{\mu=1}^{l} c_{\nu\mu} B_\nu(x) B_\mu(y) \tag{5.8}$$

with the constraint that the interpolation reproduces the control points exactly: $S(x_i, y_j) = f_{ij}$ for all $i = 1, 2, ..., n$ and $j = 1, 2, .., l$. The functions $B_\nu(x)$ and $B_\mu(y)$ are the one-dimensional cubic B-splines $B_i(x)$ over two sets of

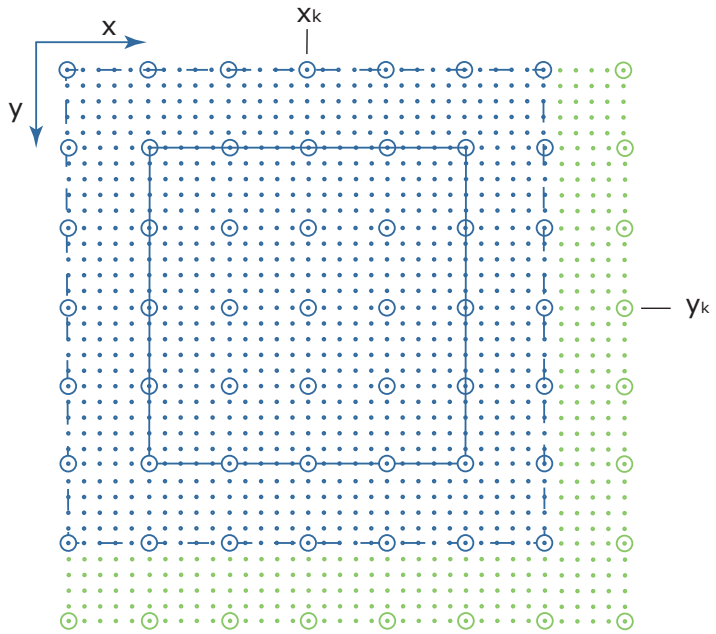



**Figure 5.9.:** Amplitudes (dots) and control points of a grid to discretize the radiation fluence and the convolution material.

control points $x$ and $y$ which are defined according to (Schwarz and Köckler 2006)

$$B_i(x) = \frac{1}{6h^3} \cdot \begin{cases} (x - x_{i-2})^3 & x \in [x_{i-2}, x_{i-1}] \\ h^3 + 3h^3(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3 & x \in [x_{i-1}, x_i] \\ h^3 + 3h^3(x_{i+1} - x) + 3h(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3 & x \in [x_i, x_{i+1}] \\ (x_{i+2} - x)^3 & x \in [x_{i+1}, x_{i+2}] \\ 0 & else \end{cases} \tag{5.9}$$

The additional index $i$ is introduced on purpose to distinguish the one dimensional definition from the two-dimensional functions $B_\nu(x)$ and $B_\mu(y)$. This definition is for equally spaced grids only. It describes the spline function for control points ($i > 3$) which are located within the material. The base functions for the boundary area are not considered, since the size of the radiation fluence is assumed to be infinite. The value $h$ denotes the distance between two control points. The two-dimensional bi-cubic B-spline function $B_{\nu\mu}(x, y) = B_\nu(x)B_\mu(y)$ is shown in figure 5.10

Throughout the plan generation, the IDS engine frequently requests certain Manipulations of the radiation beam fluences. To implement these modification very quickly into the plan, the algorithm has to perform three steps. First, the Manipulation is integrated into the fluence representation using the B-spline model introduced above. Second, the value of certain control points of the convolution material are re-calculated using the discretized method of equation 5.6. Third, the convolution material is reconstructed from the control points by a spline interpolation. All three steps will be discussed in the following.
The Manipulation itself is expressed as a relative change of the fluence at one or more control points. This should be demonstrated on an example: Assume that an increase of 10% dose in only one control point $(x_a, y_b)$ is requested. This is realized by increasing the coefficent of the affected control point for 10% ($\Delta c_{ab} = 0.1$). According to equation 5.9 the corresponding B-spline has a support of 4 intervals. Thus, the fluence amplitudes change within an area comprising 5 control points in each dimension. This area is called a Manipulation *patch* and is bordered by solid blue lines in figure 5.9. The amplitudes within the blue rectangle are re-interpolated

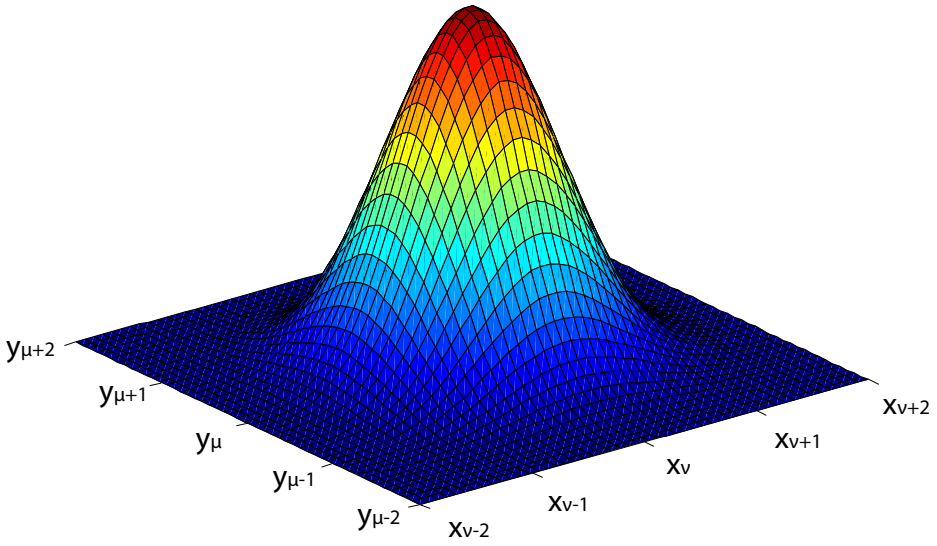



**Figure 5.10.:** The two-dimensional bi-cubic B-spline function $B_{\nu\mu}(x, y)$

by using equation 5.15 considering the coefficient $c'_{ab} = (1 + \Delta c_{ab})c_{ab}$ for control point $(a, b)$.
In order to update the convolution, the dose calculation decides for each control point (node) if it has to be re-calculated according to equation 5.6 or if it can be just copied from the current convolution material. For the example presented in figure 5.9 all nodes within the dotted line have to be re-calculated. This area comprises the patch itself and a margin of one control point at each side to account for scattered particles. The values of the control point from the green area are assumed to remain unchanged and therefore are just copied. At the end of this operation, the new convolution material is described only by the values $g_{ij}$ of the convolution in its control points. To retrieve the full material, the coefficients $e_{\nu\mu}$ of the interpolation problem have to be found by solving the linear equation:

$$\sum_{\nu=1}^{n}\sum_{\mu=1}^{l} e_{\nu\mu}B_{\nu}(x_i)B_{\mu}(y_j) = g_{ij} \tag{5.10}$$

This system has $n \times l$ equations and the same number of unknown variables $e_{\nu\mu}$. Thus, there exists one unique solution which can be found using a QR decomposition method (Schwarz and Köckler 2006). Since IDS only uses the inner B-spline function 5.9 the solution to this problem can be found directly by matrix inversion. Equation 5.10 can be reformulated to:

$$\tilde{B}_{\nu}E\tilde{B}_{\mu} = G \tag{5.11}$$

with G and E being the matrix representation of the values of the convolution at the control points $g_{ij}$ and of the coefficients $e_{\nu\mu}$ of the B-spline representation, respectively. The tensor $\tilde{B}_{\nu}$ and $\tilde{B}_{\mu}$ contains the values of the B-spline function (equation 5.9) at its control point $x_{\nu}$ and $y_{\mu}$, respectively:

$$\tilde{B}_{\nu} = \tilde{B}_{\mu} = \begin{pmatrix} 2/3 & 1/6 & 0 & 0 & 0 & \dots & 0 \\ 1/6 & 2/3 & 1/6 & 0 & 0 & \dots & 0 \\ 0 & 1/6 & 2/3 & 1/6 & 0 & \dots & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & \dots & 0 & 1/6 & 2/3 & 1/6 & 0 \\ 0 & \dots & 0 & 0 & 1/6 & 2/3 & 1/6 \\ 0 & \dots & 0 & 0 & 0 & 1/6 & 2/3 \end{pmatrix} \tag{5.12}$$

It can be shown that $\tilde{B}_\nu$ and $\tilde{B}_\mu$ are symmetric full rank matrices. Thus the inverse can easily be calculated and the coefficients of the B-spline representation of the new convolution material are retrieved by

$$E = \tilde{B}_\nu^{-1} G \tilde{B}_\mu^{-1} \tag{5.13}$$

All matrices involved in equation 5.13 are quadratic and the number of elements equals the number of control points on the grid. With the coefficient matrix $E$ the full convolution material can be interpolated by

$$C(x,y) = \sum_{\nu=1}^{n} \sum_{\mu=1}^{l} e_{\nu\mu} B_\nu(x) B_\mu(y) \tag{5.14}$$

For the calculation of the final physical dose three convolution materials have to be updated, one for each depth dose component (see equation 5.5). Thus, the dose $D^{'}$ at a point $(x_p, y_p)$ relative to the central axes and at a radiological depth $d$ is then derived as a superposition of three convolution materials multiplied with the corresponding depth dose components:

$$D(x_p, y_p, d) = \sum_{i=1}^{3} C_i(x_p, y_p) * D_i(d) \tag{5.15}$$

The dose update calculation method which is described in this section may seem a little arbitrary to the reader. However, it was chosen that way because there is a very efficient realization of this method on modern computers which is described in the next section. The next section will also reveal the reason why it is beneficial to go back to discrete convolutions in spacial domain and using the B-spline method from a computational point of view.

### 5.3.2. Computational aspects of local dose update calculation method

In section 5.3.1 a method for implementing localized dose Manipulations (patches) proposed by the IDS engine was introduced. The method is based on Bortfeld's singular value decomposition of pencil beam kernels for high energy photon (Bortfeld et al. 1993). For the dose update calculation described in this work, the convolution between kernel and radiation fluence is taken out in a discretized spatial domain. It has been discussed that in order to reduce runtime, the convolution is only calculated for certain control point of the radiation fluence from which the full material is then interpolated by a two-dimensional bi-cubic B-spline method. The purpose of this section is to demonstrate that the dose update calculation can efficiently be implemented on modern computer systems. A few vital technical details of the implementation will be provided.

The advantage of solving the integral 5.6 directly in spatial domain is that a single point of the convolution can be easily calculated by only executing a number of multiply-add operations. Multiply-add operations are highly optimized in modern computational hardware and yield a great arithmetic performance. Modern CPUs are equipped with an instruction-level parallelism[4] which calculates 8 of the so called FMA (fused multiply add) operations on single precision float point numbers per clock cycle. Equation 5.6 can be realized by a FMA operation. It consists of the multiplication of a kernel weight value with a fluence value and adding the result to the corresponding convolution value. Furthermore, the convolutions in two different points $(x_p, y_p)$ are totally independent from each other. Therefore, it is straight forward to also implement a thread-level parallelism by assigning different locations of the convolution material to different cores of a processor. A modern server based processor comprises about 10 cores each running at approximately 3 GHz. Thus, a total number of $2.4e^{11}$ fused multiply add operations can be calculated per second. The basic algorithm for the fast convolution in spacial domain is outlines in listing B.1. listing B.1 demonstrates how the fused multiply add instructions of a modern CPU can be used. The three kernel weight functions $w_i(x, y)$ (figure 3.2) are discretized over a two-dimensional

[4] http://software.intel.com/en-us/avx/

array which are referenced in line 4-6 by `kernel0_p`, `kernel1_p` and `kernel2_p`, respectively. The amplitudes of the radiation fluence are stored as a one dimensional array that is referenced by the `fluenceSample_p` pointer (line 2). The convolution itself for all three pencil beam kernels takes place within the two for loops (line 15 to 24). The intrinsic command `_mm_loadu_ps` loads 8 consecutive single precision values from the address provides as a formal parameter. For example in line 15 and 17 the algorithm retrieves 8 amplitudes from the radiation fluence and the first kernel and stores it in the register variables `fluenceSample_m256` and `kernel_m256`. In line 18 the values in both register variables are multiplied element wise and added to the existing `conv0_m256` register which is pre-populated with the convolution results of previous loops. The final result of this FMA operation is stored again in the convolution register. This procedure is repeated for the second and third kernel. Please note that each intrinsic operates on 8 values at once. The actual coordinates of the fluence amplitudes and kernel values in the computer memory are calculated by the init functions in line 8,11 and 12. The operations performed in line 15,17 and 18 are pictured in figure 5.11 in detail.

This basic algorithm for the convolution of the pencil beam kernel is very fast and very effective. It directly

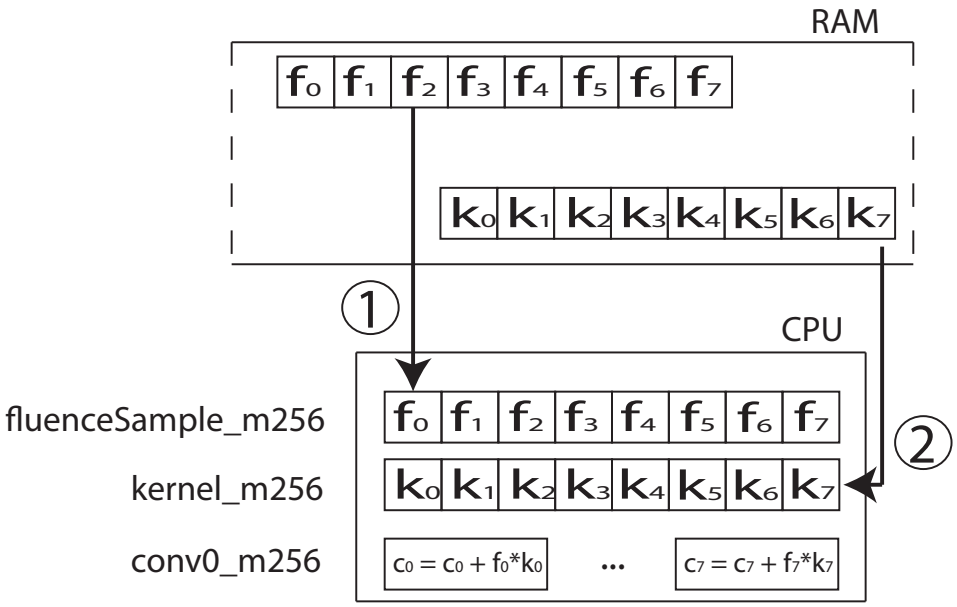



**Figure 5.11.:** Basic operations of the convolution algorithm. (1) 8 consecutive fluence amplitudes are retrieved from the main memory of the computer and stored into the CPU register `fluenceSample_m256`. (2) the corresponding 8 kernel weight values are retrieved and stores in the register `fluenceSample_m256`. using the FMA functionality of modern CPUs the convolution of all values can be performed in one clock cycle.

exploits the instruction level parallelism of modern CPUs. In contrast to the conventional dose calculation using the pre-calculated dose influence matrix (listing A.2) it is not expected that the load operations in line 15,17,20 and 23 form a von-Neuman bottleneck. The kernels and the fluence amplitudes are accessed in a consecutive order and therefore can be cached in higher memory for fast access. Thus, the convolution as outlined is a compute bound problem and fits very well in the strategy for developing fast and scaling applications (see figure 4.11).

Apart from calculating the discrete convolution in spatial domain, solving the linear equation system equation 5.15 is another performance critical part of the dose update calculation. As described in the previous

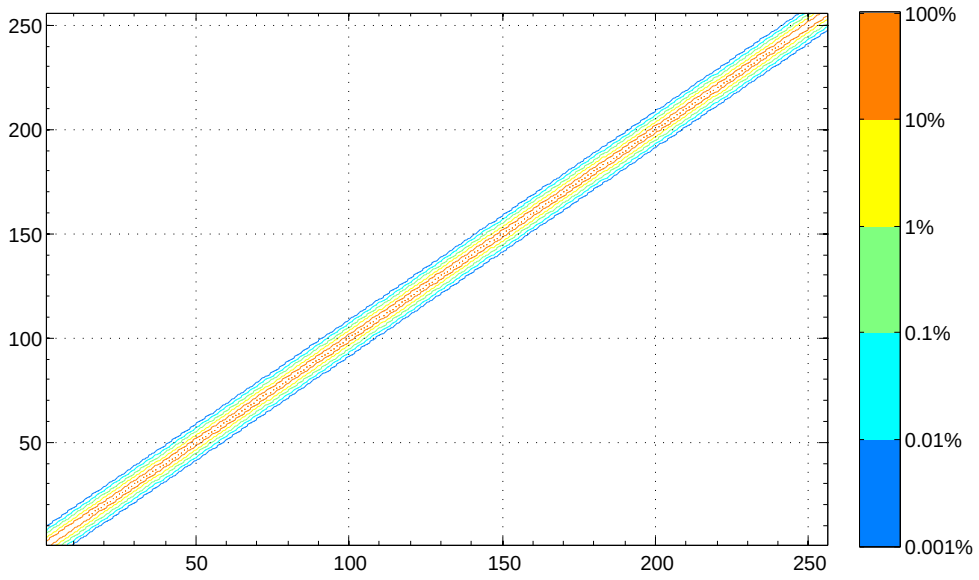



**Figure 5.12.:** The relative values of the inverse B-spline matrix. The matrix has significant values only in close proximity to the diagonal.

section, the solution can be directly found by a matrix multiplication according to equation 5.13:

$$E = \tilde{B}_{\nu}^{-1} G \tilde{B}_{\mu}^{-1} \tag{5.16}$$

A matrix multiplication based on single precision (SGEMM) or double precision float point numbers (DGEMM) is a common problem in computer science. However, a deep understanding of the computational hardware is needed to implement a near-peak performance solution (Goto and Geijn 2008). By now, there are many libraries on the market that provide optimized algorithms for handling matrices. Among the most known are: Intel Math Kernel Library (MKL)[5], Basic Linear Algebra Library (uBLAS)[6] and ATLAS (Whaley and Dongarra 1998). For arbitrary matrices involved in implementation of equation 5.13 it would be reasonable to use one of these libraries. For the dose update calculation in IDS an own implementation for the matrix multiplication is provided which is adapted to the structure of the problem. Compared to general SGEMM task, the calculation of the B-spline coefficients has two main differences. First, all four matrices are quadratic and the dimension equals the number of control points defined for describing the radiation fluence and the convolution material. Typically, the number of B-spline control points is not more than 256 in each dimension. So, the memory usage of matrix $G$, $E$ and $\tilde{B}_{\nu}^{-1}$ is about 786.4 (3 matrices with the size of $256^2$ elements of single precision float point format that occupies 4 byte). This amount of data can easily be stored permanently in the cache of a CPU so that issues of data transportation from the main memory does not have to be considered (as it is usually done for larger matrices). Second, the spline matrix $\tilde{B}_{\nu}^{-1}$ (equation 5.12) is a symmetric sparse matrix which only holds elements near the diagonal entries. It is supposed that the inverse of this band matrix is sparse, too. For a full size B-spline matrix of $256 \times 256$ elements which is build according to the scheme in equation 5.12 the inverse only holds 37% of nonzero elements. Since the B-spline fit for the dose update calculation can tolerate some uncertainties, it is interesting to take a closer look to the elements of the inverse B-spline matrix in figure 5.12 The graphic shows a typical band structure of sparse matrices. White areas in this matrix are populated with entries below 0.001% of the largest absolute value on the diagonal of this matrix. In numbers, only 3.5% of the

[5] http://software.intel.com/enus/intelmkl
[6] www.boost.org/doc/libs/1_53_0/libs/numeric/ublas/

inverse matrix entries are within the blue iso-line in figure 5.12. A second look at equation 5.13 reveals that the sparse character will propergate to the result matrix $E$, since every element in $E$ is a factor involving the multiplication of two elements from the inverse B-spline matrices $\tilde{B}_{\nu}^{-1}$ and $\tilde{B}_{\mu}^{-1}$. Because a low runtime is an important factor for the dose update calculation, it is not reasonable to perform a full matrix multiplication by considering a SGEMM problem.
The algorithm for solving equation 5.13 was designed to deliver a fast answer by exploiting overhead-free instruction-level parallelism. Furthermore, it is desired to implement a means for controlling the distance from the diagonal which is considered for the multiplication. Doing that, the matrix band with the smallest considerable value is chosen. listing B.2 shows the main components of this algorithm. The algorithm implements a matrix multiplication $C+ = A \times B$ with a value check on A. Therefore, the order of element wise multiplication was changed to allow an efficient value check on the one hand and to implement a simple cache optimized reading of the matrix entries on the other hand. According to (Goto and Geijn 2008) there are additional sophisticated methods for implementing the matrix multiplication on modern CPUs. Also, instead of using the wide AVX registers this algorithm only exploits the smaller SSE registers because during the time of the implementation the AVX technology was not yet fully available.
The two most outer for loops in line 5 and 7 refer the elements of matrix A in a row major order. The function `rangeCheck()` decides if the matrix element is large enough to be considered for the matrix multiplication. If the value is accepted it is broadcasted to the 128 bit register `aik_m128` which now holds 4 copies of this value (line 12). The inner for loop (line 16) runs over all elements of matrix $C$ in row $i$ and of matrix $B$ in row $k$. The multiplication of the elements is performed in line 20, also in 128 bit register. Please note that line 20 is again a typical multiply-add operation. Although using SSE, there is no possibility to combined them to one single fused multiply add instruction.
The thread-level parallelism is not shown in listing B.2. As for the convolution algorithm, the overhead for creating and synchronizing threads would be too high to see the benefit of running the the matrix multiplication on different cores. The thread-level parallelism is integrated at a higher level in the dose calculation at a much coarser grain.

This section briefly introduced the two main operations for the dose update calculation: First, the convolution of the pencil beam kernels with the fluence amplitude equation 5.6 in listing B.1 and second the fitting operation of the B-spline model to the control point data which is described by equation 5.13 and realized in listing B.2.

### 5.3.3. Performance and accuracy of the IDS dose update calculation

In this section the runtime and the accuracy of the IDS dose calculation engine introduced in section 5.3.1 is discussed. A low runtime is important, since the dose calculation is part of the time critical dose Variation and Recover (DVR) process.
Figure 5.13 shows the runtime of the IDS dose calculation engine for four different dose calculation scenarios on the optimized fluence modulation of the prostate case (see figure C.2). The figure depicts the calculation process as a time bar plot which shows the time consumption of the integral parts of the dose calculation. The first time bar entitled $D_{exact}$ denotes the conventional dose calculation:

$$D'_{exact}(x_p, y_p, d) = \sum_{i=1}^{3} D'_i(d) \sum_x \sum_y \Psi'(x, y) F(x, y) \times w_i(x - x_p, y - y_p) \tag{5.17}$$

It bluntly calculates the convolution between the fluence and the kernel weights in a discretized spatial domain for every voxel and every beam of the therapy plan. Even with the highly efficient convolution algorithm B.1 it takes 12.7 s to derive a complete dose distribution for the prostate case.
To improve the runtime of the calculation, an alternative method has been introduce in section 5.3.1 which only calculates the convolution on certain control points and interpolates the full material using a B-spline model.

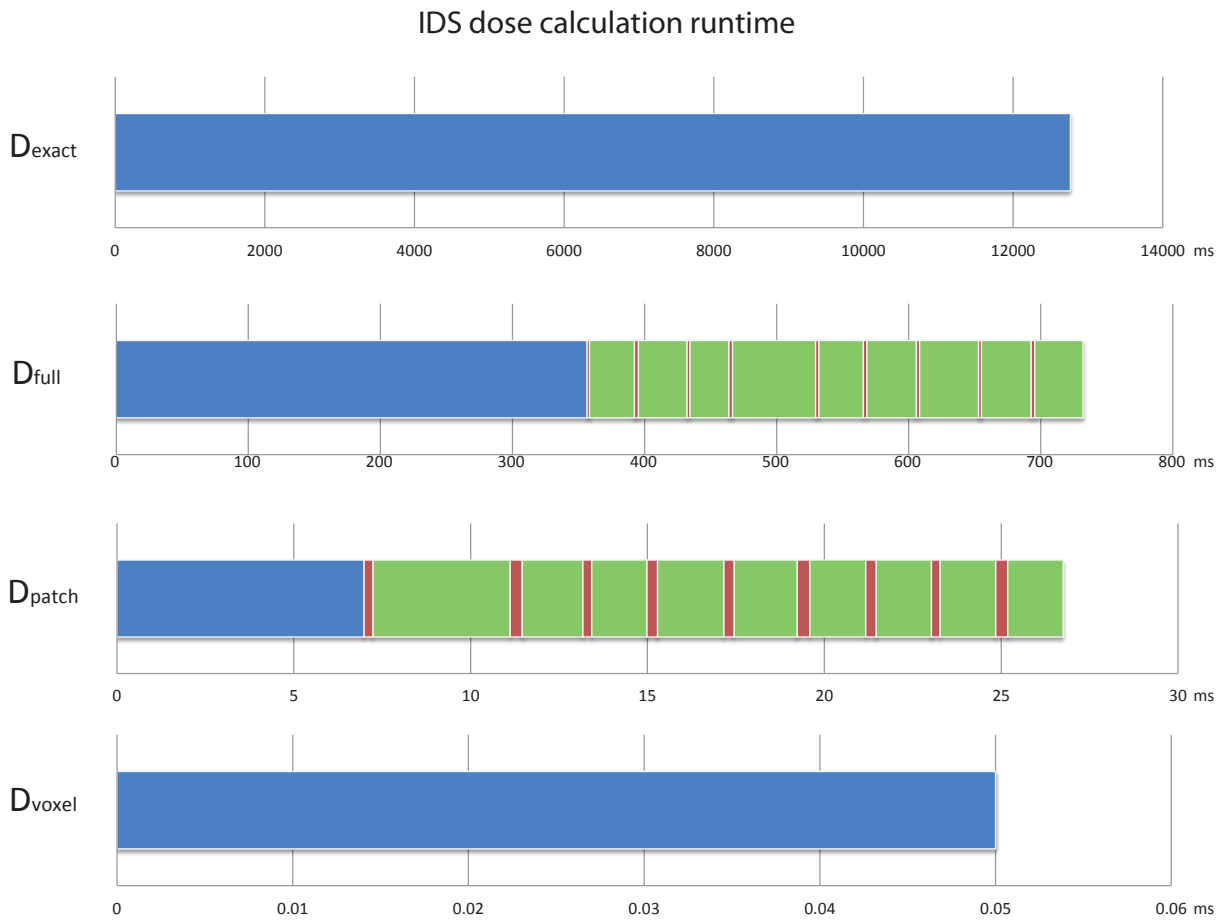
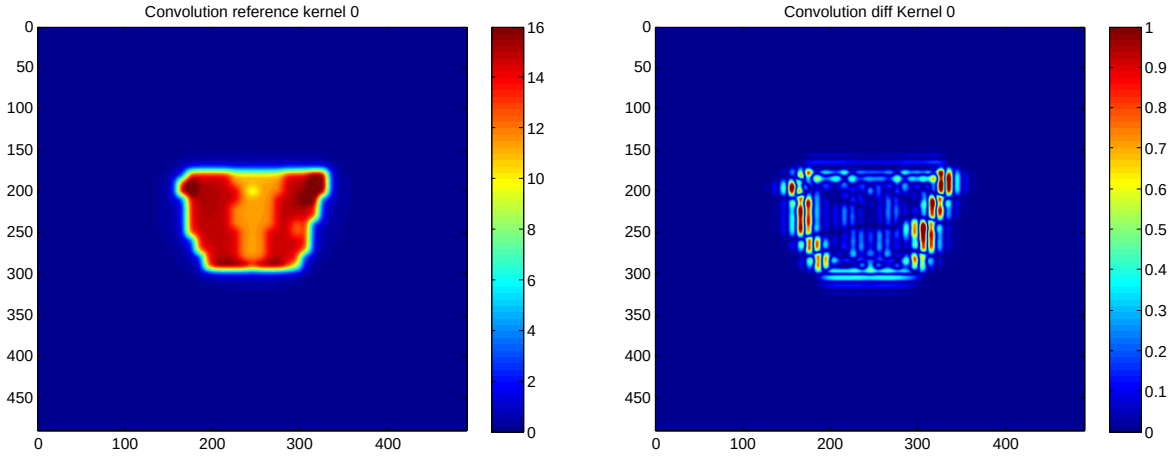



**Figure 5.13.:** Typical runtime of the IDS dose calculation engine for four different scenarios on the clinical nine beam prostate case.

The implementation of this method $D_{full}$ consists of three parts. First, calculating the convolution at the control points for all beams of the plan (blue bar, 365 ms) according to equation 5.6. Second, calculating the spline parameters (red bar ≈ 2.5 ms) according to equation 5.13 and third interpolating the amplitudes of the convolution material (green bar, ≈ 32 ms) according to equation 5.15. The convolution at the control points is taken out for all beams in one block for performance reasons. The fitting of the spline parameters and the interpolation is drawn as individual blocks for all nine beams of the plan. Both cases, $D_{exact}$ and $D_{full}$ assume that the whole fluence map changes or rather no dose has been calculated before.
For the implementation of a patch as it is used frequently in the Variation and Recovery module of IDS, the dose update calculation is shown in the third bar plot ($D_{patch}$). Within less than 30 ms the dose of the whole treatment plan can be updated. The fraction of the convolution for all beams is smaller, since there is only a limited number of control points involved in the patch. The spline parameter solver (red bars) takes about 0.3 ms while the interpolation costs about 1.8 ms in average.
The dose update calculation $D_{patch}$ includes all voxels of the plan which are influence by the Manipulation patch. In some cases, for instance for the dose probing in algorithm 5.4 it is only necessary to re-calculate the dose in one specific voxel $i$. This is achieved by method $D_{voxel}$ which performs the conventional dose calculation in equation 5.17 for only one point, i.e. for one set of $(x_p, y_p, d)$. For all nine beams of the prostate plan the single point dose calculation can be done in 0.05 ms.

The runtime of the dose calculation does not depend on the content of the clinical data used, but only on the size of the data. The prostate case is a typical example of an IMRT plan and most treatment plans use clinical data of a similar size. Thus, it is not necessary to examine another planning case to verify the performance of the dose calculation method.

The key feature of the fast dose calculation introduced in this work, is the approximation of the convolution by using a B-spline model. The following discussion is intended to compare the dose accuracy of using the conventional convolution on the one hand ($D_{exact}$) and the B-spline interpolation ($D_{full}$) on the other hand. The determination of the dose components $D_i$ and the radiological depth is the same for both methods. Thus, the comparison is focused on the differences in the convolution materials and its effect on the dose distribution. Figure 5.14 draws a comparison for the first kernel component of beam 0 of the optimized plan for the clinical prostate case. The left column shows the exact convolution (performed by $D_{exact}$) while the right column shows the absolute differences to the convolution retrieved by using the B-spline model. The primary dose deposition is described by the convolution of the radiation fluence with the first kernel weight $w_i(x - x_p, y - y_p)$ shown in figure 5.14(a). The basic structure of the fluence (see figure C.2(f)) for the first beam can still be recognized. The absolute difference shown in figure 5.14(b) reveals that the spline model of the convolution re-produces the inner high dose area of the convolution well. Here, the convolution amplitudes are relatively smooth and homogeneous. However, in the gradient regions at the edges of the material, the spline model diverges from the exact method. Large difference occur in that regions for amplitudes which have to be interpolated between the control points of the spline model. The comparison of the other two components draws a similar picture (see figure 5.15: On the edges of the convolution material the gradient region is only poorly reproduced by the spline model.
The effect of the differences in the convolution material on the dose distribution is shown in figure 5.16 for the prostate case and for the intracranial case (see appendix C.3) Each of the curves characterize the relative number of voxels (y-axis) according to there reference dose (x-axis) which show a dose deviation of not more than 1%. The meaning of this graph is explained for one example on the prostate case: The curve reveals a remarkable ditch of 99.2% at 8 Gy reference dose. That means, that the dose deviation of the B-spline method compared to the exact reference method (equation 5.17) is smaller than 1% for 99.2% of all voxel in the therapy plan which have a reference dose (calculated by $D_{exact}$) between 7.5 and 8.5 Gy. Further, the figure shows that all voxels which have a dose of more than 8.5% fulfill the 1% accuracy criteria. The curve for the intracranial plan showns a similar behaviour. Here, all voxel which have at least 4.5 Gy or higher show a dose deviation of




(a) (b)

**Figure 5.14.:** Comparison of the convolution material for the first kernel component involved in the prostate case. The left column shows the absolute convolution amplitudes derived by the exact method $D_{exact}$. The right column shown the absolute difference between the convolution materials calculated by $D_{exact}$ and $D_{full}$ which used a B-spline interpolation to reduce the convolution time.

less than 1%.

### 5.3.4. Discussion

The fast dose calculation method introduced in this section is a vital part for the realization of the IDS paradigm. A fluence Manipulation patch as it is requested by the Variation or Recovery module can be implemented within less than 30 ms for a nine beam plan. With that, the real-time constraint of the whole DVR process can be fulfilled. The dose calculation method is based on the theory of the singular value decomposition of pencil beam kernels for high energy photon. The convolution between the fluence amplitudes and the kernel components is calculated in spatial domain and modeled by a B-spline interpolation.
The original method proposed by (Bortfeld et al. 1993) used a discrete Hartley transform for calculating the convolution material in frequency domain. For the application of IDS this is not suitable, since the dose Variation and Recovery relies on a fast incorporation of localized fluence changes into the radiation map. In frequency domain this cannot be done efficiently so that a switch to spatial domain seems reasonable. The huge arithmetic workload of the convolution is reduced by interpolating the convolution of a few control points using a two-dimensional B-spline function. The B-spline function was chosen because it has minimal support which is especially beneficial for the re-convolution of only a localized patch of the two-dimensional material. Furthermore, the fitting of the B-spline parameter to the values of the control points and the interpolation only involve basic arithmetic operations which can be executed very fast on modern processors.The performance results show that for a typical nine beam plan a DVR patch can be implemented within less than 30 ms into the dose distribution. Using the original pencil beam method even the highly optimized version presented by (Siggel et al. 2012) took 599 ms for a nine beam plan on an eight core workstation. Thus, a speedup factor of almost 20 could be achieved by using the B-spline interpolation method presented in this work.
The differences of the dose calculation between using the slow, exact convolution in spatial domain and the B-spline interpolation is acceptable for the high dose regions of the plan. The large deviation for voxels below 10 Gy (see figure 5.16) only plays a minor roll, because the interesting regions which characterize a treatment plan are the coverage of the target (which is at the upper end of the dose scale) and the sparing of the organs at risk which usually take more than 10 Gy. The good agreement of the interpolation model for high dose areas

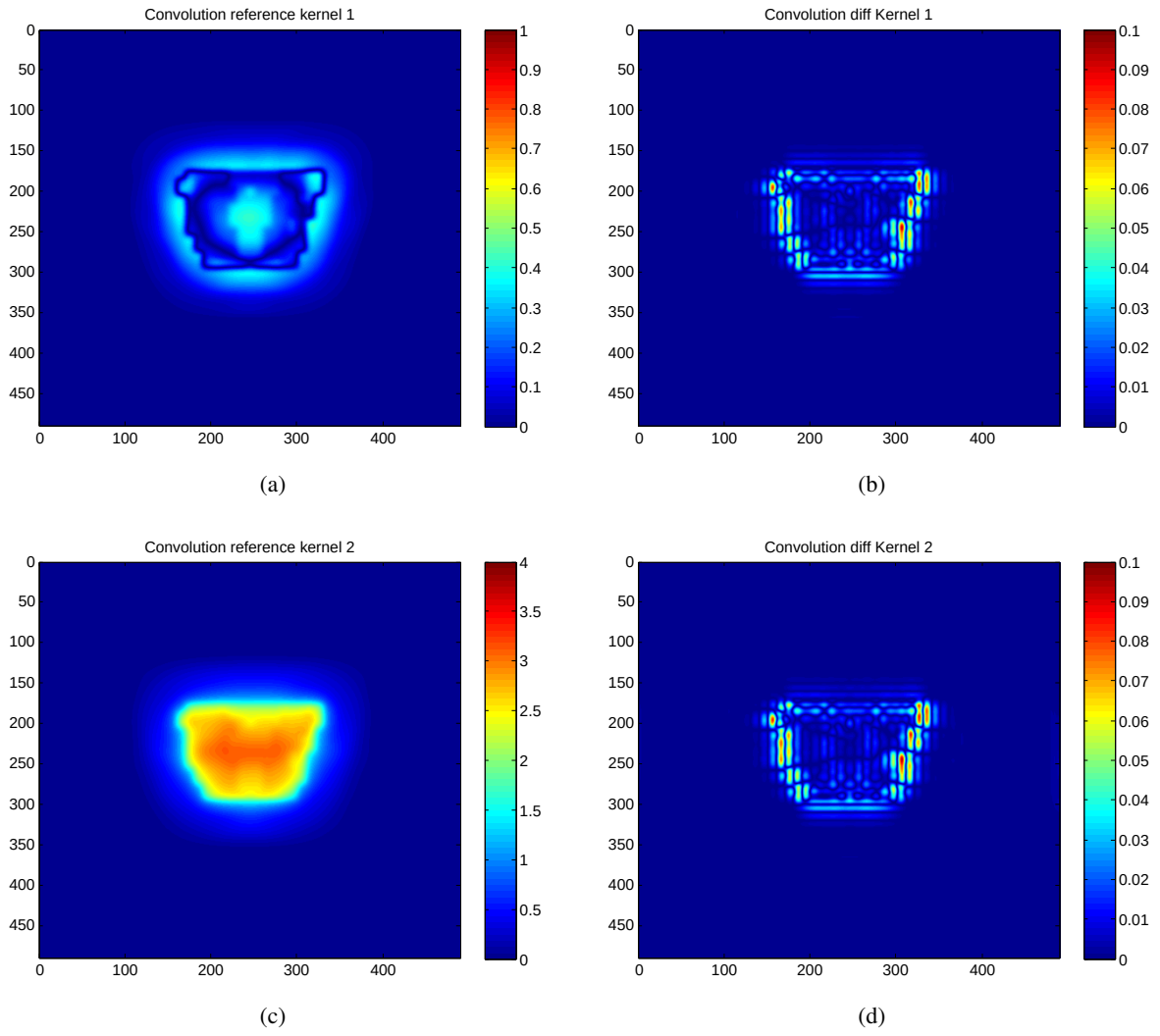
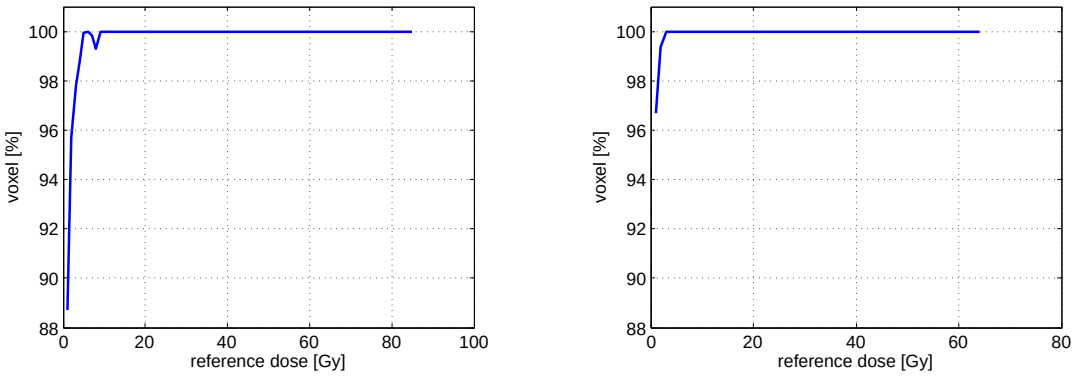



**Figure 5.15.:** Comparison of the convolution material of the scattering kernel components involved in the prostate case. The left column shows the absolute convolution amplitudes derived by the exact method $D_{exact}$. The right column shown the absolute difference between the convolution materials calculated by $D_{exact}$ and $D_{full}$ which used a B-spline interpolation to reduce the convolution time.




(a) (b)

**Figure 5.16.:** Effective accuracy of the dose distribution calculated using the B-spline interpolation ($D_{full}$) compared to the exact convolution model $D_{exact}$ for a) the prostate case and b) the intracranial case.

is due to two reasons. First, the target coverage is designed by the therapist to be as homogeneous as possible. That means, usually the high fluence areas do not contain a steep gradient which results in a smooth convolution material (see figure 5.14(a)) that can be approximated very accurately by the B-spline interpolation (see figure 5.14(b)). The relatively large deviation on the edge of the target projection area do not have a significant influence on high dose voxels. This can be explained by the synergistic character of an IMRT plan: The voxels which project onto the convolution area of one specific beam showing a large deviation, also receive dose contributions from the other beams. In general the projection onto these other beams will fall into a high dose area which is approximated well. So the punctual high deviation of the convolution on one beam is diminished by the other beams. Only the voxels which project to the gradient region on all beams will show a significant dose variation. These voxels, in turn, receive a small amount of dose and are not in the focus of the therapist.

Despite all the advantage of the pencil beam method, the general quality of this method is limited. It is surely sufficient for calculating the dose of an intermediate planning step as it is generated by the DVR process to propose the next dose Variation. For a final dose calculation prior to treatment delivery the plan has to be re-evaluated by a more accurate method. Nowadays, fast superposition methods or even Monte Carlo methods (Jia et al. 2011; 2012) are used for an accurate clinical application. The final dose calculation is independent from the IDS planning and not part of this work.

## 5.4. Selecting voxels for Dose Variation and Recover

This section introduces strategies of how to chose a sequence of voxels for the IDS engine. Within the process of a dose shaping step, the task of selecting a voxel appears two times. First, in the Variation process which imposes the dose shaping request and second in the Recovery process which compensates global unintentional plan modifications. In figure 5.2 the selection module is marked by a green box.

By choosing a voxel, the location of the plan is specified on which a Variation or Recovery step is carried out. This is vital for the success of the IDS step. Especially for the Recovery process the degree of how well the original dose distribution could be re-installed throughout the plan depends on the selection of the recovered locations. An obvious criteria for selecting a voxel to be recovered is its dose. The larger the dose deviation of a voxel to the desired value is, the more eligible it seems to recover this voxel. For the phantom example shown in figure 5.4(a) this would cause the algorithm to start recovering the voxels which are situated in the inner

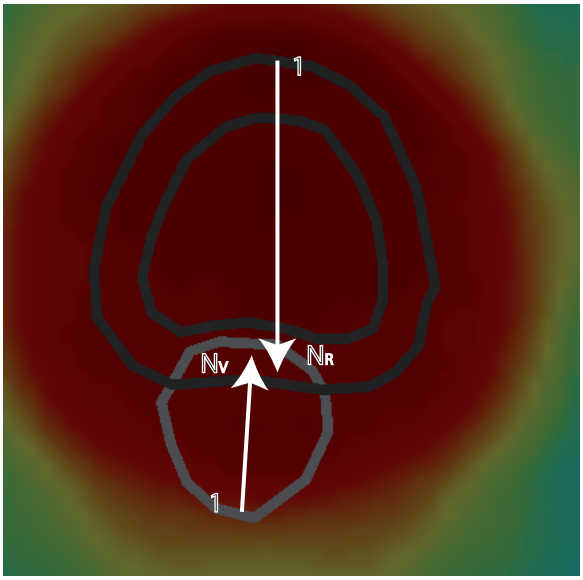
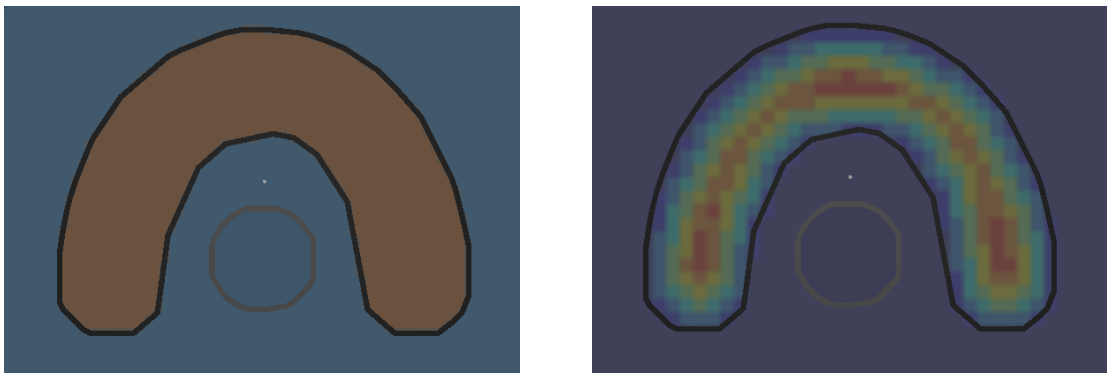



**Figure 5.17.:** The course of selecting voxels for the Variation and Recovery process on the prostate case.

boundary of the target volume where the under-exposure of the target is highest due to the inverse square law of the incident radiation. This is generally not a good strategy. If the target and the OAR are in closer proximity as in the phantom case, the selection by highest dose will fail for the Recovery. A real patient geometry which is abstracted by the phantom case is shown in figure C.2. The figure describes a clinical prostate case where the therapist attempts to install a dose gradient directly between the rectum (organ at risk) and the tumor (target volume). Due to the geometry of this case the dose gradient has to be very steep and large deviations will always occur in the region where rectum and prostate are close together. By selecting only voxels from this proximity region, the Recovery (which attempts to push the gradient out of the target) and the Variation (which attempts to push the gradient into the target) will produce a stalemate situation which results in an alternating dose distribution throughout the DVR process.
A superior strategy would start recovering voxels from locations which are farther away from the gradient region. That way, the dose can be build up towards the proximity of OAR and target which produces a homogeneous dose pattern (see figure 5.17). The same strategy seems reasonable for the Variation process. Assume the user requests the dose impose shaping modality which prescribes a maximal dose for the whole organ at risk. Here, it is also recommended to start imposing the maximal dose from the region which has to highest distance to the gradient area. Imposing dose on regions which are far away from the target is easier and more degrees of freedom can be used. Furthermore, imposing dose to a OAR region which is far away from the target also has an influence on the proximity region (due to particle scattering). Thus, Variations which concern eventually the voxels near the target volume will only cause a small (the remaining) fluence Manipulation.

In order to implement this selection strategy the IDS engine has to define a notion of distance between an arbitrary voxel and the proximity of other volumes of interest. The conventional clinical data set does not provide such distance information, so it has to be calculated by the IDS engine before the first Variation starts. The notion of distance is generated by the distance transform which is explained in the next section. The idea of the selection strategy that was developed qualitatively in this section is then quantified in section 5.4.2.



(a) (b)

**Figure 5.18.:** A transversal slice through the phantom geometry showing a) the binary VOI grid of the target b) the distance map of each target voxel to the boundary of the target.

### 5.4.1. Distance transform

The IDS engine proposes a dose variation based on the current dose distribution of the treatment plan. For proposing the next planning step, the IDS engine has to know how the dose pattern is correlated to the patient geometry. In conventional planning this information is provided by a VOI cube (volume of interest cube). The VOI cube has the same dimensions as the dose cube. It is a binary data set indicating if a voxel belongs to a certain volume of interest or not. On the example of the horse-shoe shaped phantom the voi cube of the target volume is shown on the left side of figure 5.18. figure 5.18(a) shows a transversal slice of the phantom geometry. It is a binary image with an overlay of the contours of the horse-shoe shaped target and the circular OAR in the emargination. Blue voxels[7] denote 0 and red voxels denote the value binary 1. Thus, every voxel which belongs to the target volume is marked by 1. Conventional planning systems store voi cubes for every volume of interest, so that the algorithm is informed about the correlation between voi structures and voxel.
For the IDS engine this spatial information is not enough. For the local modification of the plan the algorithm must know where the voxel is situated within the voi structure. A metric has to be defined which specifies a distance to the borderline of the voi. Thus, the image is not just binary, but contains a distance relative to a certain structure. In figure 5.18(b) the distance of every voxel in the target structure to the borderline is shown. Blue denotes a low distance and red denotes a high distance. Voxels outside of the target are not considered and set to $-1$. The algorithm which generates the distance map (distance transform) was first described in (Rosenfeld and Pfaltz 1968) and is briefly introduced in this chapter

Let $I = \{(i, j), 0 \leq i, j \prec dim\}$ be the binary image as shown in figure 5.18(a). The voxels of the object (in this case the target) have a value $v(i, j) = 1$ and form a subset $S = \{(i, j), v(i, j) = 1\}$ of the image. The voxel of the background of the image $\bar{S} = I/S$ not containing the object is set to zero. According to (Danielsson 1980) the distance map $L(S)$ is

$$L(i, j) = min\left(d\left[(i, j), \bar{S}\right]\right) \tag{5.18}$$

which assigns a label to every voxel in $S$ that defines the shortest distance to the background $\bar{S}$. There are many ways to define the distance functions $d$. In this work, the so called chessboard distance $d_8$ is chosen:

$$d_8((i, j), (h, k)) = max\left(|i - h|, |j - k|\right) \tag{5.19}$$

where $(i, j)$ and $(h, k)$ are the coordinates of voxels in the image. It is called chessboard distance, because it resembles the number of steps a king in the board game chess has to take to reach the adjacent 8 fields. For

[7] Although, an image technically contains pixels, the term voxel will be used in this context because the transversal slice intersects the center of one layer of voxels here.

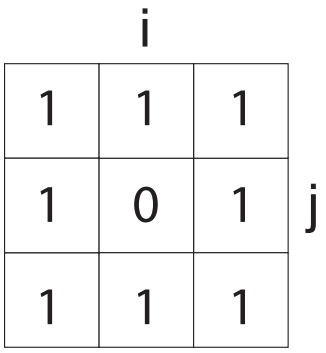



**Figure 5.19.:** The definition of the chessboard distance $d_8$ relative to the center element of the quadratic field.

an arbitrary voxel $(i, j)$ the distance to the next neighbors is shown in figure 5.19 The chessboard distance was chosen because it is the closest integer relaxation of the euclidean norm which represents an intuitive notation of distance between two points. On the one hand, the euclidean norm is too expensive to calculate for every voxel in the planning setup. Furthermore, it may not be necessary to calculate the exact distance between two voxels since the IDS engine just needs a rough idea where the voxel is situated within the object. On the other hand, there is a very fast algorithm to generate the distance map based on the chessboard distance according to (Sonka et al. 1999). The algorithm is shown in listing 5.1. It performs two passes over the voxels of the

**Algorithm 5.1** Distance Transformation

1: **function** DISTANCETRANSFORM(*Image I*)

**input:** Image $I$, a binary image, where the object voxels are marked 1 and background 0
**output:** distance map $L$, the distance map according to equation 5.18

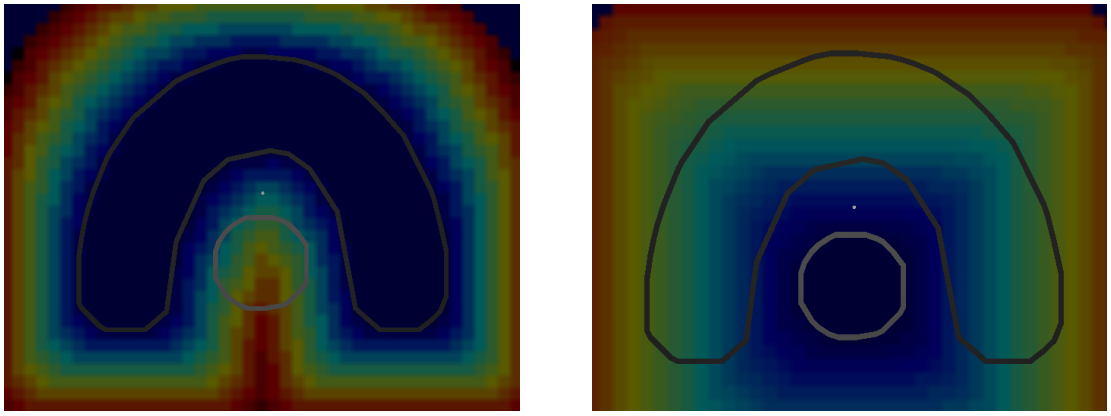
**local variables:** $i$, index of the image from left to right
$j$, index of the image from top to bottom
$dim$, number of voxels in each dimension of the quadratic image
$v(h, k)$, value of the image at voxel $(h, k)$
$A, B$, set of neighboring voxels according to figure 5.20

2: **for** $i = 1..(dim - 2)$ **do** ▷ First pass
3: **for** $j = 1..(dim - 2)$ **do**
4: **if** $(i, j) \in S$ **then** ▷ process only voxels within the object
5: $L(i, j) \leftarrow \min\{v(h, k) + 1 \; with \; (h, k) \in A\}$
6: **for** $i = (dim - 2)..1$ **do** ▷ Second pass
7: **for** $j = (dim - 2)..1$ **do** ▷ process all voxels of the image
8: $L(i, j) \leftarrow \min\{L(i, j), v(h, k) + 1 \; with \; (h, k) \in B\}$

binary image $I$. In the first pass the image is traversed from left to right and from top to the bottom of the image (line 2,3). The second pass runs from right to left in the bottom up direction (line 6,7). In each pass the value of the distance map at voxel $(i, j)$ is adjusted based on the values of the neighbors. For the first pass only 4 neighbors out of 8 which are marked by $A$ in figure 5.20 are considered (line 5). The value of the distance map is derived from the minimum of the neighboring value increased by 1. In the second pass the remaining 4 neighbors marked by $B$ in figure 5.20 are considered. The final distance map at point $(i, j)$ is

| | | |
|---|---|---|
| A | A | A |
| A | (i,j) | B |
| B | B | B |

**Figure 5.20.:** Definition of the neighbor sets *A* and *B* for the use in algorithm 5.1



(a) (b)

**Figure 5.21.:** A transversal slice through the phantom geometry showing a) the inverted metric of the target and b) the inverted metric according to the spherical shaped OAR.

then determined from the minimum of the neighboring values increased by 1 and the distance from the first pass.

The algorithm transforms a digital image in which the object is marked with 1 (figure 5.18(a)), into a distance map (figure 5.18(b)). It is easy to modulate the algorithm so that an inverse distance of the objects is calculated as shown in figure 5.21(a) and figure 5.21(b). The inverse distance assigns to every voxel outside of the object the nearest distance to the object itself. That can be used to define a whole variety of distance maps which expresses a correlation between the voi structures of the treatment plan. For clinical setups containing more than 2 OARs, it is possible for instance to calculate for every target voxel the distance to the nearest OAR borderline. This information can be used to effectively drive the IDS voxel selection process, as it is explained later.

From a computational point of view, the generation of a distance map is very pleasing. The algorithm is based on deriving the minimum out of a fixed number of values from the neighboring voxel. This operation is highly optimized in the SSE and AVX instruction set of modern computational hardware[8]. The SSE and AVX registers can hold up to 4, respectively 8 data values so that the determination of the minimum in line 5 and 8 can be done in one clock cycle. Since the distance map is only generated within one slice of voxel, the amount of data which is accessed is relatively small. One slice of a common treatment plan can reside in higher memory while the operation is executed, so that a memory bottleneck is not expected. Furthermore, the algorithm is well suited for parallel execution. The distance maps of each slice are independent from each other, so that

[8] http://software.intel.com/en-us/avx/

each core can operate on an individual slice without the need for synchronization during the execution.

The distance map only relates voxel within one slice. It is not a full three dimensional information set which compares arbitrary voxel. This issue is discussed later in this work in detail. The chessboard distance is easy to calculate, although it does not give the absolute spatial distance between the centers of two voxels. For the purpose of this work this is supposed to be sufficient. It provides a relative measure of distance between locations which is used by the IDS engine to chose an appropriate voxel from a possible set of candidates for the dose modification.

### 5.4.2. Strategies

With the help of the distance transform which creates a notion of proximity between voxels in the plan, two indicators for the selection can be used: the dose of a voxel and the distance between a voxel and the closest borderline of targets or OARs in the plan. The selection strategy which was motivated in section 5.4 is implemented for the Variation and Recovery process as shown in algorithm 5.2.

**Algorithm 5.2** Voxel selection for Variation and Recovery

1: **function** SELECTVOXEL(*VOI*, *gtMod*, *selectionMod*)

**input:** *VOI*, specifies the volume of interest for which a voxel is selected
*gtMod*, comparator which specifies the rank between voxels
*selectionMod*, selection modality

**output:** *selectedVoxel* the voxel which is selected

**local variables:** $i$, index of a voxel
$d_i$, dose of voxel $i$
$V$, set of candidate voxel, for Variation $V = V_M$ and for Recovery $V = V_R$
*presDose*, desired Dose in voxel $i$
$Y$, vector of voxels sorted by the comparator *gtMod*

2: **for all** $i \in VOI$ **do** ▷ look at all voxels i in VOI
3: **if** $d_i \neq presDose$ **then** ▷ dose in i deviates from prescribed dose
4: $V+ \leftarrow i$ ▷ add i to set $V = V_M$ for the Variation or $V = V_R$ for Recovery
5: $Y \leftarrow sort(V, gtMod)$ ▷ sort $V$ according to the comparator *gtMod*
6: $index \leftarrow generateIndex(selectionMod)$
7: $selectedVoxel \leftarrow Y[index]$ ▷ select one voxel from sorted $Y$

The pseudo code reveals that the basic selection algorithm is the same for Variation and Recovery. For selecting a voxel for the Variation, the *VOI* parameter is set to an organ at risk to which the dose shaping request was made. For the selection within the Recovery process the *VOI* parameter is set to the target of the therapy plan. The selection begins with running through the specified *VOI* and adding every voxel which violates the prescribed dose to a vector $V$ (line 4) which equals $V_M$ for Variation or $V_R$ for Recovery. The voxels added to $V$ are sorted according to the rule *gtMod* which is specified as a comparator between two elements of a vector. For the use in IDS the following four comparators can be used:

$$gtMetric(lhs, rhs) \Leftrightarrow lhs.metric > rhs.metric \tag{5.20}$$

$$gtDoseAbs(lhs, rhs) \Leftrightarrow |lhs.dose - lhs.targetDose| > |rhs.dose - rhs.targetDose| \tag{5.21}$$

$$gtDoseMetricWeighted(lhs, rhs) \Leftrightarrow \tag{5.22}$$
$$(\alpha * \frac{lhs.metric}{maxMetric} + \beta * \frac{|lhs.dose - lhs.targetDose|}{|maxDeltaDose|}) > (\alpha * \frac{rhs.metric}{maxMetric} + \beta * \frac{|rhs.dose - rhs.targetDose|}{|maxDeltaDose|})$$

$$gtDoseMetricAbs(lhs, rhs) \Leftrightarrow \tag{5.23}$$
$$((lhs.metric = rhs.metric \wedge gtDoseAbs(lhs, rhs)) \vee (lhs.metric \neq rhs.metric \wedge lhs.metric > rhs.metric))$$

The comparator establishes a rank between the elements of a vector. It compares two arbitrary voxels (one on the left-hand side (lhs) with another voxel on the right-hand side) and returns true if the voxel on the left-hand side has a higher relevance than the other voxel. The comparators in equation 5.20 and equation 5.21 are used to sort the canditate voxels $V$ according to the metric and the dose deviation to the prescribed value, respectively. Comparator 5.22 and 5.23 take both, the metric and the dose of a voxel into account to establish an order of the candidate voxels.
The index generation in line 6 of algorithm 5.2 choses one voxel from the pre-sorted vector of candidates. The most intuitive index is 1 which selects the head element of the candidate vector. It is the voxel which has the highest relevance according to the selected comparator. In addition to that, stochastic index generating methods has been implemented. It turned out that sampling a voxel according to a normal distribution is an efficient way to relax the order imposed by the comparator.

### 5.4.3. Performance of the voxel selection process for Variation and Recovery

The runtime of the selection process for dose Variation and Recovery is shown in figure 5.22 on the example of the prostate case (appendix figure C.2)

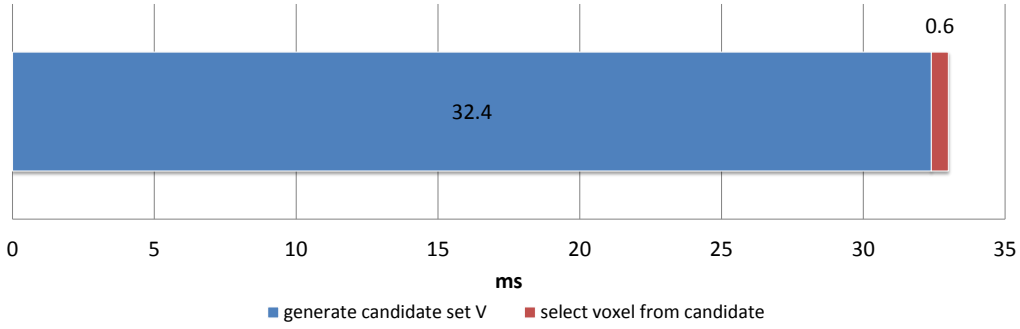



**Figure 5.22.:** Absolute runtime of the IDS voxel selection process

The runtime is illustrated as a time bar plot containing the generation of the candidate voxel set $V$ (line 2 to 4 of algorithm 5.2) and the final selection of the most eligible voxel (line 5 to 7 of algorithm 5.2). The generation of the candidate set takes 32.4 ms and is therefore one of the most time consuming processes throughout the IDS modules. Furthermore, it is a memory bound operation since the generation of $V$ has to visit every voxel of certain VOIs in the plan to check if a certain criterion is met. However, the absolute runtime is low enough to fulfill the real-time requirement in the context of the whole DVR module.
The final selection of the most eligible voxel can be done very fast (0.6 ms). This includes the sorting process of the candiate set $V$ and the generation of an index which points the final voxel in $Y$ (line 7 in algorithm 5.2). Sorting the candidate set $V$ is done by the C++ Standard Template Library (STL) which consumes the most of the 0.6 ms.

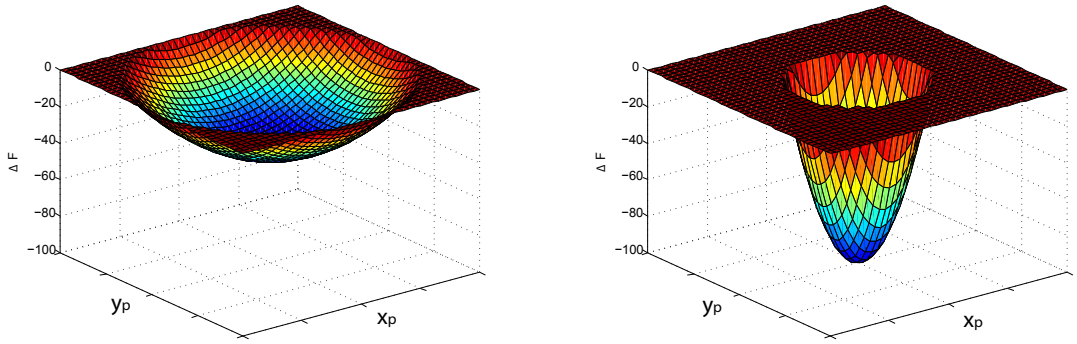



**Figure 5.23.:** Two possible shapes of a fluence Manipulation patch. a) a dose Variation is imposed by a large patch of a relatively mild modulation b) the same Variation effect is achieved by a small patch of larger amplitude modulations.

## 5.5. Fluence Manipulation for dose Variation and Recovery

The fluence Manipulation is a vital process for the IDS engine. It is triggered during the Variation to impose the requested dose feature on a voxel and in the Recovery to compensate for an unintentional change of dose. The fluence Manipulation module (marked by a red box in figure 5.2) operates on a single voxel which was chosen by the selection module before (see section 5.4).

Establishing a desired dose feature within one voxel is an ambiguous task for two reasons. First, the IMRT relies on a synergistic effect between the incident beams. Only one radiation beam could not generate the typical conformal dose distribution for which IMRT is known. The spatial conformity of the dose pattern to the patient geometry is established by several incident beams coming from different angles. For installing a Variation or Recovery within a voxel the mulit-beam setup provides an additional degree of freedom: The fluence Manipulation can be done on all beams equally or only on a subset of beams. The second reason which creates ambiguity is the occurrence of lateral scattered particle. The lateral dose profile allows to vary the number of fluence amplitudes which are involved in the Manipulation process. For instance a dose Variation can be implemented by applying a large Manipulation of a few fluence amplitudes around the projection of the selected voxel or by imposing a relatively mild Manipulation on a widespread area around the projection. Both cases are shown for the fluence of one radiation beam in figure 5.23. The surface denotes the Manipulation of the fluence map in order to install a desired dose feature in voxel $i$. The projection of voxel $i$ onto the radiation fluence intersects the amplitudes at the point $(x_p, y_p)$ which is the center of the patch $\Delta F$.

Creating an appropriate fluence modulation for the Variation and Recovery poses two challenges. On the one hand, size and shape of the modulation has to be determined. On the other hand, the IDS engine has to decide which beams have to be involved in the modulation. For the Variation of the phantom case example in figure 5.4(b) the IDS engine proposed a relatively small fluence patch while the intensity of the Manipulation is distributed equally over all incident beams. Since the shaping request was limited to only one voxel, it is reasonable to modulate as few amplitudes as possible. Please not, that it is generally not possible to only modulate the one fluence amplitude to which the voxel projects. Due to scattering the amplitudes in the proximity also contribute significant dose to the Variation voxel, so that there is a minimal size of the fluence Manipulation patch depending on the intensity of the imposed dose Variation. Due to the symmetrical shape of the target volume it is expected that no incident beam angle is preferential for imposing the Variation.
The situation changes for real clinical cases as for instance the prostate case in figure C.2. Here, to install a

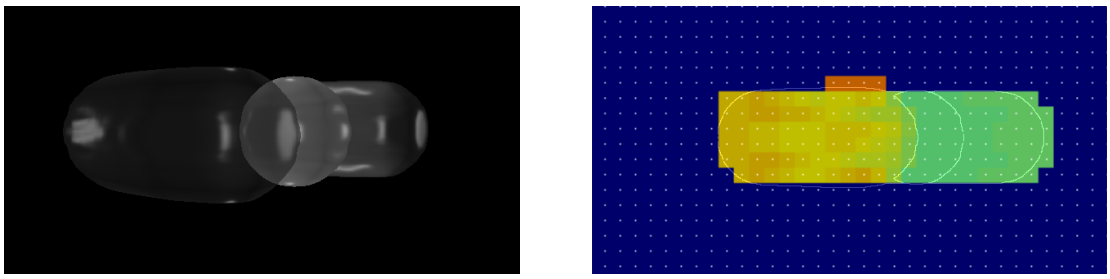


**Figure 5.24.:** a)The beam's eye view for beam 3 onto the phantom geometry. b) The TPI data set according to the target calculated for the control points of the fluence map of beam 3.

Variation in the rectum the algorithm would prefer for instance beam 2 and 7 over beam 0. Modulating the fluence of beam 0 is not reasonable, since it does not only influence the dose in the rectum (as intended for a Variation) but also influence a large number of voxels in the target which shrouds the rectum seen from that angle. The situation is inversed for a Recovery scenario. Since beam 0 has the shortest and direct exposure to the target it will be preferred to re-install a prescribed dose. Since the OAR lies relatively deep within the patient, the rectum will only get a mild exposure from beam 0 because of the exponential decrease of the primary photon fluence (see figure 3.2). The size and shape of a modulation patch depends on the requested Variation or Recovery and also on the geometry of the patient. If the shaping request relates to a larger volume than a larger size of the fluence Manipulation can be chosen. For the Recovery, usually a relatively small patch is used, since there is a number of Recovery processes which all install a adiabatic dose change to the selected voxels.
In order to generate appropriate Manipulation patches for Variation and Recovery, the IDS engine has to know the effect of changing a fluence amplitude on the resulting dose pattern. This relation is established conceptually by the dose calculation. Although, calculating dose for creating the fluence Manipulation patches is way too time consuming. Instead a two-dimensional data set called beam travel path information (TPI) is created. One TPI data word is assigned to each fluence amplitude. It specifies how much influence a Manipulation of this amplitude has on a give VOI. With that information, the IDS engine can decide which of the fluence amplitudes are beneficial for impose a Variation or re-installing a desired dose distribution. The generation of the TPI value is explained in the next section while the creation of the fluence Manipulation patches based on TPI is introduced in section 5.5.2.

### 5.5.1. Beam travel path information

In the last section the concept of the distance map for IDS was introduced. It extends the voi grid of a conventional treatment plan by providing a notion of distance to a specific voi boundary for each voxel. With that, the IDS engine has access to a geometrical information base which describes the patient geometry independent from the beam setup of the treatment. In order to define a relationship between the geometry of the patient and the radiation fluence, a second set of data called TPI (*beam travel path information*) is introduced. The TPI data set denotes an *intensity of influence* on target and OAR structures for each radiation fluence amplitude. An example for the horse shoe phantom is shown in figure 5.24. Figure 5.24(a) shows the phantom case (figure C.1) through the beam-eye view for radiation beam nr. 3 coming from behind and to the left side. The OAR is partially covered by the target volume. However, in this image the OAR is fully visible since the target is rendered transparent. Figure 5.24(b) shows the same beam's eye view which is overlaid with the TPI values generated for that radiation beam. The white lines in this picture mark the edges of the phantom geometry projected onto the beam's eye view. From right to left 4 segments of a circle are drawn as white lines. The first denotes the right wing of the target volume, the second marks the visible edge of the OAR and third and fourth circle border the left side of the target which is in front for that beam direction. The color map represents the intensity of the TPI which is calculated at the resolution of the control points for the B-spline representation

(white points) of the fluence. At each control point a virtual beam is cast through the phantom geometry. The amount of influence each beam has on the target volume is recorded and exhibited as a TPI color code. Light green denotes a low influence while yellow and red indicate a high influence. The absolute scale of the color are not mentioned since this example is only designed to demonstrate the main principle of TPI. The color pattern can be explained as follows: Assume each control point is the source of a small beam (pencil beam) which is sent towards the plan with an angle relative to the phantom as shown in figure 5.24(a). The set of pencil beams which intersect the left wing of the target will have an influence on that wing and pass through the right wing of the horse shoe in the rear area of the phantom (high influence, high TPI marked by darker colors). In contrast to that, the set of small beams which miss the foremost left wing will only traverse part of the rear right wing of the phantom (low influence, low TPI marked by lighter colors). The color coding in Figure 5.24(b) reflects this behavior. Apart from the fact, that the left group of pencil beams influences a larger number of target voxels, there is another reason for their high TPI: The intensity of a photon beam decreases exponentially in matter (see figure 3.2). That means, the intensity of influence is higher for voxels which are closer to the surface of the patient.

In order to quantify the intensity of influence, the TPI value is defined over the set of control points $(x_p, y_p$ on the fluence map for each beam $b$:

$$TPI(VOI, b, x_p, y_p) = \sum_{i \in P(VOI,b,x_p,y_p)} D_1^{'}(d(i)) \tag{5.24}$$

$D_1^{'}(d(i))$ is the depth dose component on the central axis according to figure 3.2. It only considers the primary radiation of a small circular field (pencil beam) and neglects scattered particle. The transformation function $d(i)$ assigns a radiological depth to each voxel $i$. This information has already been received through the ray-casting method which is a prerequisite for the fast dose calculation described in section 5.3.1. The sum in equation 5.24 runs over voxels $i$ which are member of the travel path $P$ of the virtual pencil beam that is emitted by control point $(x_p, y_p)$. The beam path $P$ contains all the voxels which are directly hit by the pencil beam. A closer look to equation 5.24 reveals that the TPI-value is the accumulated primary dose over all influenced voxels in a certain VOI which is imposed by a pencil beam that intersects the location of the control point $(x_p, y_p)$ on the radiation fluence. As shown later, this information is valuable for the IDS engine to propose a Manipulation of the fluence amplitudes in order to impose a certain change in the dose pattern.

From a computational point of view, the generation of the TPI data set seems to be a costly operation. In order to generate the path $P(VOI, b, x_p, y_p)$ for every pencil beam, another ray-casting operation has to be taken out. The number of rays which have to be casted is at least equal to the number of control points on all radiation beams of the planning setup. To avoid the that, the TPI values are determined by algorithm 5.3 which is not intuitively identical to equation 5.24. The TPI data set is usually generated individually for the target volumes and for the OARs which is controlled by the parameter $C$ in line 3. For each voxel in $C$ the projection onto the fluence map for all beams is calculated. The projection determines the nearest control point to the straight line which stands perpendicular on the fluence map and intersects the center of voxel $i$ (see figure 5.25). In line 6 the TPI value is increased by the value of the depth dose component $D_1{}'(d(i))$ by taking the radiological depth of voxel $i$ into account. After processing all voxels of a certain class of voi (target or OAR) the algorithm 5.3 generates the TPI data set according to equation 5.24.
The time critical projection function in line 5 transforms the three-dimensional coordinates of a voxel (patient system) to a point on the radiation fluence (collimator system). The transformation is carried out in homogeneous coordinates, so that the intersection point on the fluence is found by a multiplication with a $4x4$ matrix:

$$(x_p, y_p, z_p, 1)^T = M \cdot (x_v, y_v, z_v, 1)^T \tag{5.25}$$

To derive the matrix M one has to take the geometry of the therapy setup into account. A detailed description of the transformation matrix can be found in (Siggel 2008, p.27). The coordinate transformation and the calculation of a correction term according to the beam divergence are efficiently realized by listing B.3.

**Algorithm 5.3** GenerateTPI

1: **function** GENERATETPI(Patient and beam geometry, *VOI C*)

**input:** Patient and beam geometry and a certain VOI (target or OAR)
**output:** $TPI(C, b, x_p, y_p)$, TPI value of beam $b$ at control point $(x_p, y_p)$ (equation 5.24)

**local variables:** $\#voxel$, the total number of voxels in the treatment plan
$\#beams$, number of beams
$x_p, y_p$, coordinates of a control point on the radiation fluence
$D_1^{'}(d(i))$, central axis depth dose component according to figure 3.2
$d(i)$, radiological depth of voxel $i$ determined by ray-casting method (Siggel et al. 2012)

2: **for** $i = 1..\#voxel$ **do** ▷ loops over the entire patient volume
3: **if** $i \in C$ **then** ▷ voxel $i$ is member of VOI $C$
4: **for** $b = 1..\#beams$ **do** ▷ loops over the entire patient volume
5: $\{x_p, y_p\} \leftarrow projectToFluence(b, i)$
6: $TPI(C, b, x_p, y_p) + \leftarrow D_1^{'}(d(i))$ ▷ add depth dose to TPI value

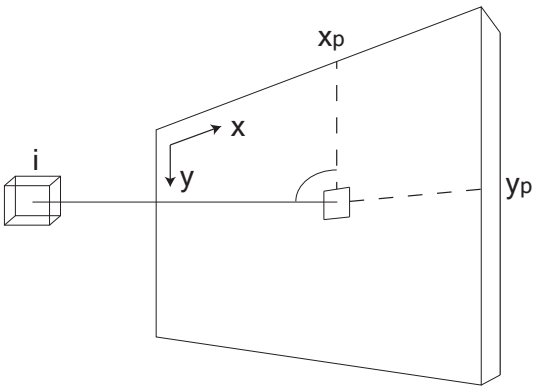



**Figure 5.25.:** Projection of the center of a voxel onto a fluence map. The coordinates $(x_p, y_p)$ are defined as the intersection point of a line which stands perpendicular on the fluence map and hits the center of voxel $i$

### 5.5.2. Strategies

With the help of the TPI data set which has been introduced in the last section, the creation of a fluence Manipulation patch is described by algorithm 5.4

**Algorithm 5.4** createPatch

1: **function** CREATEPATCH(voxel $i$, coreSize $c$, lateralSize $s$, $targetDose$)

**input:** voxel $i$, the voxel which has been selected for Variation or Recovery
coreSize $c$, the inner range of the Manipulation patch (see figure 5.26 )
lateralSize $s$, the range which is taken into account for lateral scattering (see figure 5.26 )
targetDose, the requested dose in voxel $i$

**output:** Patch $\Delta F$, the resulting fluence Manipulation

**local variables:** $A$, the fluence Manipulation patch
$\#beams$, number of beams
$TPI$, two-dimensional TPI data set per beam for $A$
$K$, two-dimensional kernel data set per beam for $A$
$k$, lateral kernel factor at a specific amplitude $(x, y)$ for beam k
$v$, TPI value for pixel $(x, y)$ for beam k
$t$, factor specifying the intensity of the fluence amplitude Manipulation

2: **for** $b = 1..\#beams$ **do** ▷ loops over all beams
3: $A(b).box \leftarrow boundingBox(b, i, c, s)$ ▷ determine the geometrical limit of the patch
4: $TPI(b) \leftarrow createTPI(b, A.box, OAR(i))$ ▷ create TPI data set on patch
5: $K(b) \leftarrow createKernelData(A.box)$ ▷ create kernel data set on patch
6: $d0 \leftarrow doseAt(i)$ ▷ save original dose in voxel $i$
7: **for** $b = 1..\#beams$ **do** ▷ create normalized patch
8: **for all** $x \in A.box$ **do**
9: **for all** $y \in A.box$ **do**
10: $k \leftarrow K(b, x, y)$
11: $v \leftarrow f(b, x, y, TPI)$
12: $A(b, x, y) \leftarrow k \otimes v \otimes \mathbb{1}$
13: $d1 \leftarrow doseCalc(i, A)$ ▷ re-calculate dose only in voxel $i$ given patch A
14: $t \leftarrow |targetDose - d0| \div |targetDose - d1|$ ▷ re-calculate dose only in voxel $i$
15: $\Delta F \leftarrow A(x, y) \otimes t$

The algorithm for creating a fluence Modification patch has three parts. In the first part (line 2 to 5) the basic data structures are defined which are necessary for the patch generation. In line 3 the geometrical limits of the patch are defined. Each patch forms a quadratic field. The center of a patch coincides with the projection of voxel $i$ which is subject to the fluence Manipulation. The total size $l$ of a patch is determined by the core size $c$ and the lateral size $s$ to $l = 2s + c$ (see also figure 5.26a). In line 4 the corresponding TPI data set is generated. For a dynamic patient setup where the geometry can change throughout the course of the treatment the TPI data set is calculated on the fly according to algorithm 5.3. For a static IMRT plan the TPI data is constant and line 4 just copies the TPI values from previously created data. Line 5 generates a set of kernel factors for each amplitude of the patch which will be explained momentarily.
In the second part of the algorithm (line 6 to 12) the actual patch $A$ is created at a normalized intensity. The amplitudes in $A$ are defined in line 12. Each data point is a factor of the terms $k$ and $v$ (line 12). The values for $k$ were defined in line 5. $k$ denotes the scatter contribution on voxel $i$ of a pencil beam coming from the

fluence amplitude $(x, y)$ of beam $b$. For that the pencil beam kernel value $w_1$ (see figure 3.2) is taken at a radius which equals the distance between the location $(x, y)$ and the nearest edge of the core in patch $A$. By using the scatter contribution of the pencil beam, the radiation modality gets involved in creating the fluence Manipulation patch. A large scatter contribution will create a large patch. This is reasonable because at a high degree of particle scattering a lot of neighboring amplitudes will contribute to the dose of voxel $i$. If there is only minor scattering, the term $k$ will approach to zero very rapidly with increasing distance, so that the Manipulation is limited to a relatively small area around the projection of voxel $i$. The term $v$ holds the TPI value which was created previously in line 4.
The factor $\mathbb{1}$ in line 12 denotes the creation of a normalized patch. So far, the requested dose in voxel $i$ was not taken into account. The effect of the normalized fluence Manipulation on the dose pattern is tested in line 13 for voxel $i$ only. Together with the original dose $d0$ which has been saved in line 6, a scaling factor for the patch amplitudes can be derived (line 14). Since the dose deposition is linear with respect to the fluence modulation, scaling the patch with $t$ is supposed to implement the desired dose feature in voxel $i$ in the final part of the algorithm.

An example for the creation of a patch on the horse shoe shaped target is shown in figure 5.26. A Manipulation is generated to realize a dose Variation which was requested for a voxel on the surface of the OAR (see figure 5.26e)). The example is illustrated for beam nr. 7 which approaches the phantom from the left side. (Please also revisit figure C.1(e) to verify the beam direction) figure 5.26e) shows the beam's-eye view of beam 7 on the patient. The patch dimension is shown in figure 5.26a). The projection of voxel $i$ is situated in the middle while the amplitudes are pre-populated with a constant factor $t$ at a range of $c+2s$ around the projection. Within the core region of the patch, the decrease of the Manipulation intensity due to scattering is neglected so that $k$ hold a maximal value for points in that region (see figure 5.26b). In this example the scattering factor approaches zero very quickly, so that only the inner region of the patch actually impose a non-zero fluence Manipulation.
The TPI data set which was created for this patch is shown in figure 5.26c). By comparing the (tilted) beam's eye view with the TPI value pattern, the relation to the phantom geometry can be seen: The TPI data set was created for a dose Variation process. It favors pencil beams which have a low influence on the target volume with a relatively high TPI value. Pencil beams on the left side of the patch should be preferred since they just miss the edge of the left wing of target. Pencil beams coming from amplitudes on the right side of the patch will primarily hit the target volume and impose a relatively high dose alteration in the target voxels of that region.
The final fluence modulation patch $\Delta F$ in algorithm 5.4 is shown in figure 5.26d). The amplitudes of the patch are derived from multiplying the TPI data with the radial symmetric kernel factors and the global scaling factor $t$. Not all amplitudes are modulated since the kernel factor quickly approaches zero at growing distance from the center point. However, the characteristic pattern imposed by the TPI data set can be recognized. The left side of the patch generates a higher Manipulation of the fluence map due to the beneficial geometrical arrangement of the phantom case.

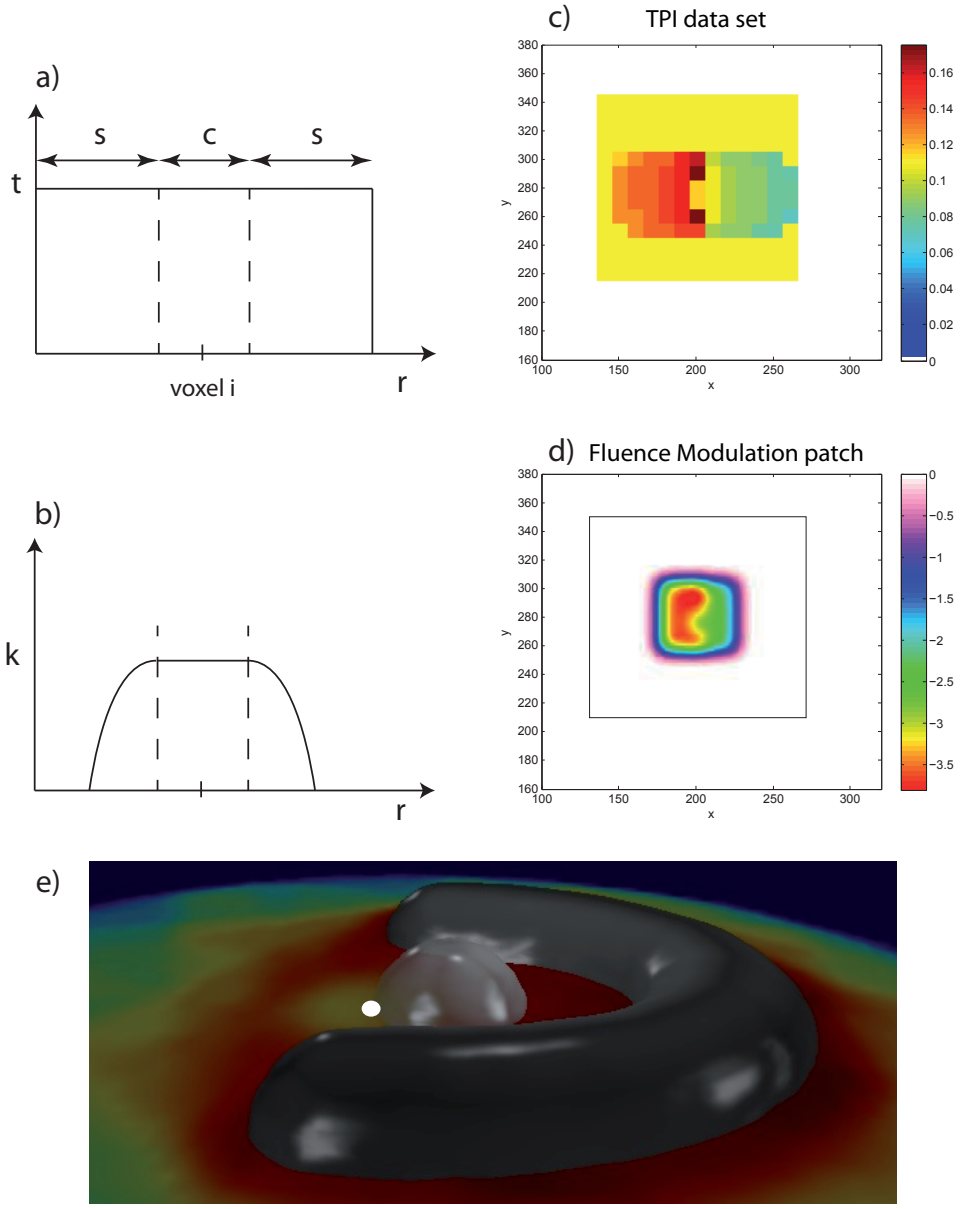



**Figure 5.26.:** Visualization of the patch generation process by example. a) The side length of the quadratic patch is defined to be $l = 2s + c$ b) the scatter kernel value is radial symmetric around the center of the quadratic patch c) The TPI data set over the patch is shown. d) The final patch amplitudes. e) The tilted beam's eye view of beam 7 is shown on the phantom geometry.

### 5.5.3. Performance of the Fluence Manipulation for dose Variation and Recovery

The performance of algorithm 5.4 is illustrated in figure 5.27.

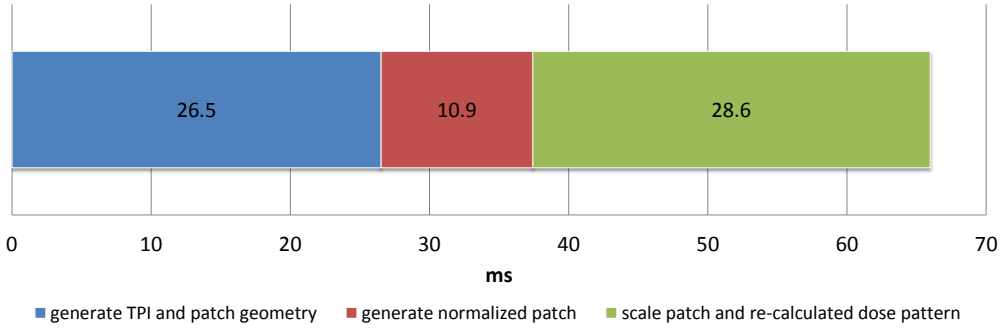



**Figure 5.27.:** Absolute runtime of the Manipulation patch creation process

It shows the runtime of the three main parts of the patch generation process which have been identified in the previous section. In this example a dynamic generation of the TPI values is assumed which pads out the first part of algorithm 5.4 (line 2 to 5) to 26.5 ms. The generation of the normalized patch with a single core control point $c = 0$ and scatter region of $l = 2$ control points takes 10.9 ms. This is a relatively localized fluence Manipulation patch as it is used for implementing a Recovery on the target volume. The last part which is depicted as a green bar in figure 5.27 includes the scaling of the Manipulation patch and a re-calculation of the dose distribution.

## 5.6. The IDS framework

To implement the Interactive Dose Shaping paradigm it was necessary to develop a dedicated prototype framework. Existing therapy planning frameworks which are also available for research purposes could not utilized. On the hand, these frameworks are designed for conventional planning methods based on the optimization of an objective function (e.g. (Nill 2001)). Most of these frameworks do not have a three-dimensional user interface which allows for the definition of local planning features. On the other hand, conventional planning frameworks which are accessible to research are not optimized for performance. They develop over time within the limits of master or PhD thesis in which providing the functionality of a concept is more important than optimizing the algorithms for performance. Since achieving a high performance and having the ability to interact with the user via an three-dimensional interface is crucial for the IDS paradigm, a completely new planning framework from scratch had to be designed. The basic structure and the key elements of this new IDS framework is introduced in this chapter.

The framework comprises two main libraries (see figure 5.28). One is the graphical user interface which also includes the handling of clinical data. The other is the Interactive Dose Shaping library which provides modules for the IDS engine and an ultra-fast dose update calculation method. The details of this framework are rather tecnically and should not be discussed in this work. Although, it should be emphasized that a complete clinical therapy planning framework including a powerful three-dimensional graphical user interface has been designed from scratch with two exceptions: On the interface side, basic routines for importing clinical plans and imaging data was re-used from previous works. On the IDS side the function for calculating the pencil beam kernel data was re-used from the work of (Siggel et al. 2012). However, the IDS engine alone comprises approximately 17000 lines of code.

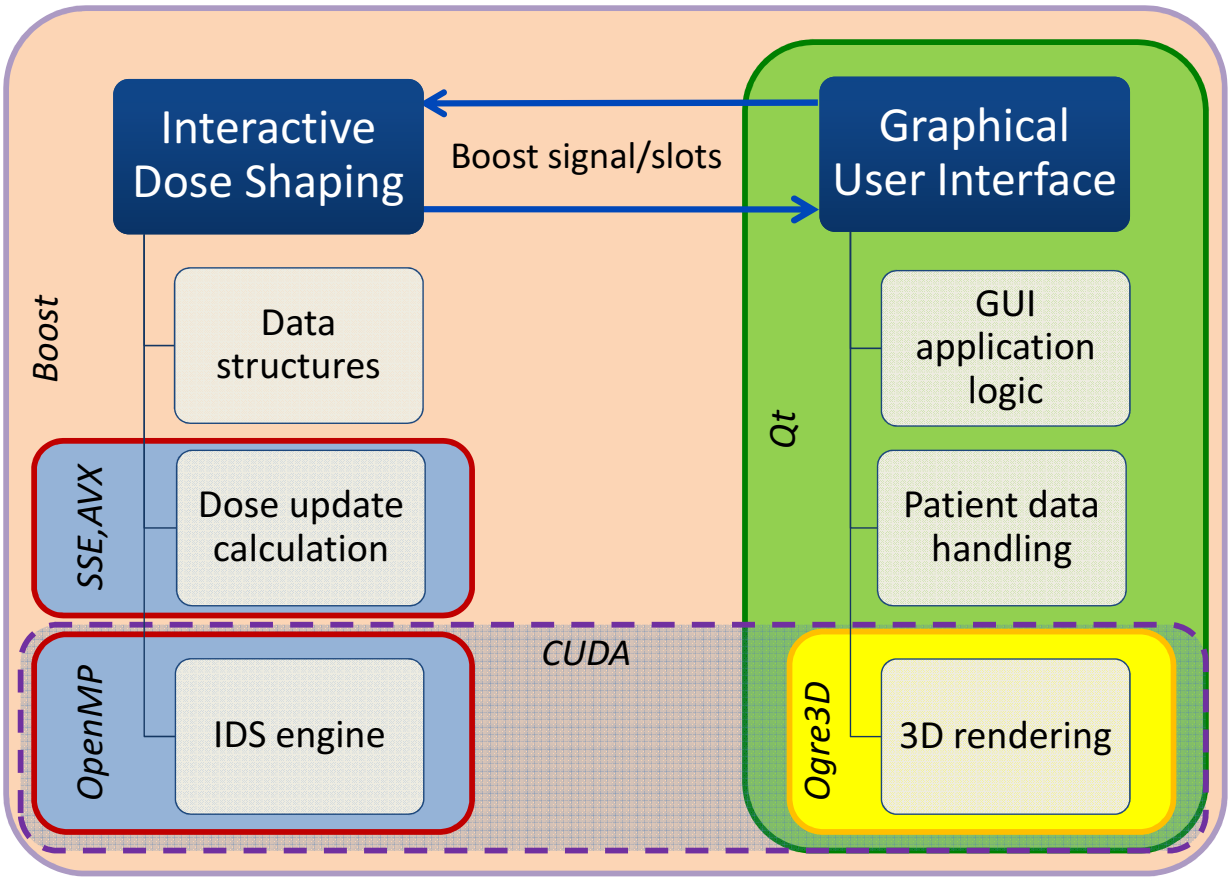



**Figure 5.28.:** Basic structure of the new IDS planning framework as a prototype to test the interface and the dose Variation and Recovery strategy. The left side comprises the key features of the IDS strategy while the right side comprises the graphical user interface and the handling of the patient data.

Several state-of-the-art technologies and libraries are used in the new framework. To exploit the performance of modern computational hardware the OpenMP API (Dagum and Menon 1998) is used to create concurrent algorithms which process data in parallel. This is necessary to distribute the work load onto multiple cores of modern CPUs (thread level parallelism). OpenMP is mainly used in the IDS engine, to calculate the Variation and Recovery Strategies for all beams of the treatment plan in parallel.
The high performance of the dose update calculation depends on the use of the SIMD (single instruction multiple data) (Patterson and Hennessy 2008) capability which is implemented in modern CPU and GPU hardware. Via SSE and AVX [9] several data words are processed at once in the registers of the processors. This is the most fine grained parallelism to be found in computer hardware and can be used for algorithm which have a very short runtime. A brief introduction of how SIMD operations are used for the calculation of dose can be found in section 5.3.2
The three-dimensional representation of the treatment plan is provided by a modern GPU (graphics processing unit). In addition to that there are some image processing task which are also done by the GPU. On the interface side this comprises the surface extraction module using a marching cube algorithm (Lorensen and Cline 1987). On the IDS side the ray-casting which is a preprocessing step for the dose calculation is also done on the GPU. The programming interface which utilizes the graphics card for general scientific calculation is called CUDA and is used for the interface and for the Interactive Dose Shaping modules (see figure 5.28). The CUDA algorithms itself should not be discussed in this work since they have been developed within another thesis.
The three-dimensional interface which allows for a full navigation in the therapy plan is realized using the Ogre3D [10] engine. Orge3D is a open-source graphics rendering engine used for games, simulators and scientific visualization. The interface of the planning framework itself was implemented using the Qt[11] libraries. Also these modules of the framework are not subject of this work and were developed mainly by another PhD project.
The whole framework uses the functionality of the Boost libraries for C++ [12] (Demming and Duffy 2010). Boost is a collection of free C++ libraries for a various range of application. In the new planning framework it is used for serializing the IDS data structures and the clinical data and for controlling the interface over multiple concurrent threads.

## 5.7. First planning results on the IDS framework

In this section the feasibility of generating therapy plans from scratch with the new IDS framework is demonstrated. By using only the impose dose Variation technique described in section 5.2 an attempt was made to reproduce the quality of conventionally optimized plans for the prostate case and the intracranial case (see appendix C.2 and C.3).

The planning result for the prostate case is shown in figure 5.29. The reference plan (dashed line) was optimized with the conventional Newton's method using clinical approved parameters for the objective function. The prescribed dose to the gross tumor volume (GTV) was set to 7600 cGy and the clinical target volume (CTV) was prescribed to 7000 cGy. The rectum which is the critical organ at risk in this case should receive no more than 4000 cGy. The DVH of the IDS plan which was received by the impose dose Variation and Recovery strategy is shown in continuous lines. The rectum was imposed to 6000 cGy in order to achieve a plan which resembles the reference plan. The DVH curves show that the IDS strategy achieved a comparable plan.

[9] http://software.intel.com/en-us/avx
[10] http://www.ogre3d.org/
[11] http://qt-project.org/
[12] http://www.boost.org/

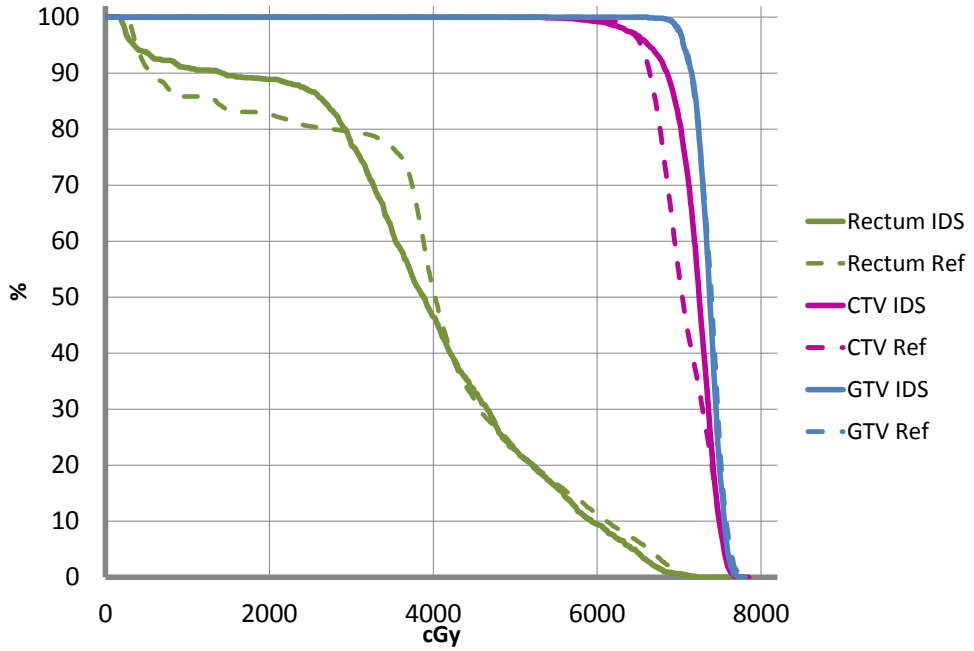



**Figure 5.29.:** DVH curve comparison of the prostate plan. The plan marked by dashed lines (reference plan) was optimized using the conventional Newton method based on clinical approved parameters for the objective function. The IDS plan (straight line) was generated by the dose impose tool with the objective to reproduce the reference plan as close as possible.

For the intracranial case which represents a more complex planning scenario the DVH curves are compared in figure 5.30. This case contains only one target volume but numerous organs at risk which are in the scope of the therapy planner. The dose to the eyes of the patient are especially important for a deliverable plan. The maximum dose to the left eye is subscribed to 1000 cGy while the maximum dose to the right eye can be as high as 2000 cGy due to the close proximity to the target volume.

Both IDS plans were generated by the framework which is described in section 5.6. Using only the impose dose Variation technique it took less than 2 minutes the generate these plans. The number of Variations in each dose impose step was set to 30. For each Variation 5 Recovery steps were triggered. The selection of the Variation and Recovery voxels was guided by the *gtDoseMetricAbs* sorting scheme defined in equation 5.23. The plan comparisons in this section are indented to demonstrate the feasibility of the IDS planning paradigm. The clinical quality of this method has to be yet investigated by a separate clinical study which is not the scope of this work.

## 5.8. Discussion

This chapter introduces the ideas and the basic concepts of implementing the Interactive Dose Shaping (IDS) paradigm. Three main parts of the IDS paradigm have been presented. First, strategies for the dose Variation and Recovery (DVR) module of the IDS planning paradigm. These modules are the key feature of the new planning method. Second, a new method for re-calculating the dose distribution of the plan according to localized Manipulations of the fluence maps as they are requested by the DVR module. Third, a brief introduction of the planning framework itself.

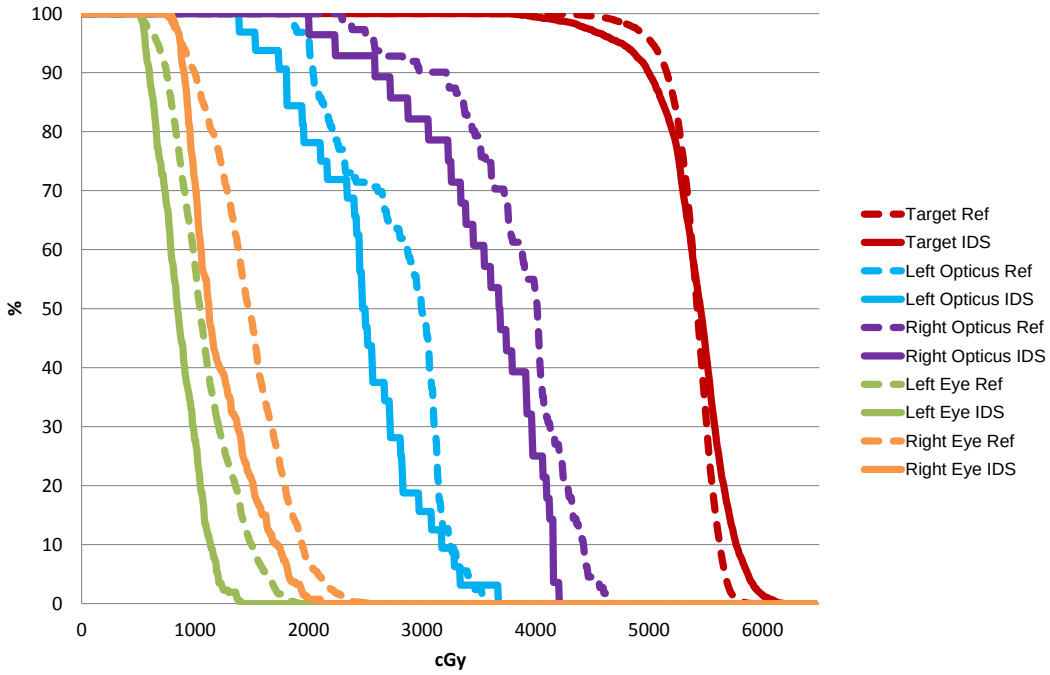



**Figure 5.30.:** DVH curve comparison of the intracranial plan. The reference plan optimized by the conventional Newton method based on clinical approved parameters for the objective function is marked by dashed lines. The IDS plan (straight line) was generated by the dose impose tool to resemble the reference plan as much as possible.

The strategies behind dose Variation and Recovery have to fulfill two demands. On the one hand, they have to be effective in order to implement the desired local dose featured which is requested by the therapist. One the other hand, the method which implement the DVR strategies must run efficiently on modern computational hardware, so that the real-time constraint of a dose feature request can be be fulfilled. Designing these strategies is a challenging endeavor for which a deep knowledge of high performance computing and medical physics is needed.

The two main components of the DVR are the selection of a modification voxel and the generation of a Manipulation patch to impose a desired dose change on the plan. In section 5.4.3 and 5.5.3 it could be shown that both operations can be executed on real clinical therapy plan in less than 35 ms and 70 ms, respectively. That means, a full Recover process can be completed in about 100 ms. The low runtime of these modules is due to the efficient algorithm presented in this chapter which mainly exploit the instruction level parallelism of the current hardware. The thread level parallelism which controls the distribution of the workload onto the cores of the CPU has not been described explicitly in this chapter. This is due to two reasons. First, the runtimes in section 5.4.3 and 5.5.3 have been acquired on a mobile computer which only has 2 CPU cores. A dedicated development system with a server processor was not available to the team during the time when the algorithms have been developed. Second, the algorithm are explicitly designed to work on individual data elements, so that the thread level parallelization is trivial anyway. For example consider algorithm 5.4 for generating a Manipulation patch. The calculation of the TPI data set (line 4) is one of the most time consuming parts of this algorithm. The TPI data is created for each beam individually. A straight forward approach to parallelized this part would be to generate each patch in a separate CPU thread. This would create as many threads as there are beams in the plan which can easily be scheduled on the CPU cores. The same argumentation holds for the generation of the patch amplitudes in line 12. Each amplitude is calculated completely independent from the others which

allows for a trivial parallelization of that task. Typically, commercial planning frameworks run on a dual server processor system. Nowadays, a server processor comprises up to 10 physical cores, so that in total 20 cores are available to the planning algorithm. These systems provide 20 times more computational power compared to the computer system which was used to generate the results of this work. Even if a moderate speedup of 10 is assumed, the runtime of a Recovery process would be reduced to about 10 ms so that approximately 100 Recover steps can be executed to fulfill the real-time constraint of the IDS paradigm. Based on that discussion, it can be concluded that the strategies for implementing the DVR module are fast enough to allow interactive planning on modern computers.

The first application of the IDS framework in section 5.6 demonstrated the feasibility of the new planning paradigm. On the example of a prostate and intracranial case, it was possible to generate a treatment plan with similar features as a conventionally optimized plan. For the prostate case an even better sparing of the rectum could be achieved (see figure 5.29). The high dose region of the rectum is a reduced a little by the IDS framework. Also the number of voxels in the rectum receiving between 3000 cGy and 4000 cGy is lower for the IDS plan. Although, the Recovery did not manage to separate the dose volume histogram curves of the GTV and CTV. Here, the Recovery did not penalized the slight over-dosage of the CTV. For the intracranial plan (see figure 5.30), the minimum dose constraint to the left eye and the right eye of the patient are of special importance. The conventionally optimized plan (dashed lines) provides a good target coverage but does not quite reach the maximal dose constraint for the eyes. Using the IDS framework a significant dose reduction could be achieved in the left eye and in the right eye of the patient. Also the dose to the optical nerves is reduced. However, the Recovery module of the IDS engine did not achieve a comparable homogeneous dose coverage of the target. The clinical quality of the IDS generated treatment plans has to be assessed in a separate clinical study which is not part of this work. However, it could be demonstrated that the IDS planning paradigm is able to interactively generate clinical plans in a reasonable time.

# 6. Summary and Outlook

In this work, a new method called Interactive Dose Shaping (IDS) for generating an intensity modulated radiation therapy (IMRT) plan is presented. Therapy planning using IDS is fundamentally different from the conventional methods. It introduces a paradigm shifting way of how radiation treatments can be planned in the future. The basic ideas of IDS have been described in (Ziegenhein 2008) and published as a patent in (Ziegenhein and Oelfke 2010) and (Oelfke 2012). In this work, the realization of the IDS concept in a three-dimensional therapy planning framework and a first application on clinical cases is described in chapter 5. The conventional method of deriving an IMRT plan is analyzed in chapter 4 and an efficient implementation of conventional IMRT planning is proposed. The discussion section of that chapter motivates the IDS paradigm from a computational perspective.

Conventional therapy planning is based on an iterative optimization method which is driven by a quadratic objective function. The free parameters of the plan are the amplitudes of the discretized radiation fluence. The objective function defines a single valued distance from the prescribed dose distribution in the plan. A dose prescription can be defined on the level of a pre-segmented volume of interest which usually coincides with a clinical organ (volume of interest, VOI) involved in the therapy. Over- or under-dosage of VOI voxels is penalized by a relative weighting factor which must be provided (together with the prescribed dose of that VOI) by the therapist. The optimal therapy plan is assumed to coincide with the set of radiation fluence amplitudes that minimize the value of the objective function.
The physics of dose is provided by a so called dose influence matrix $D_{ij}$ which holds the radiation dose contribution of the discretized fluence amplitude $j$ to a voxel $i$ in the plan. This matrix is pre-calculated and stored in the memory of the planning system before the plan optimization begins. (Nill 2001) identifies this as a modular approach while he points out that the $D_{ij}$ matrix can be calculated by another algorithm even on another computer. For advanced application in medical physics, a high accuracy and a low runtime of the IMRT plan optimization is vital. The accuracy is basically controlled by the generation of the dose influence matrix. The runtime depends on the computational realization of this optimization scenario on modern computers. The optimization problem is analyzed in section 4.1 from a computational point of view. It is discussed that the therapy planning problem using Newton's method based on the pre-calculated dose influence data is a memory-bound problem on modern hardware. That means, the performance of that algorithm is limited by the throughput of the data (the dose influence matrix) which has to be transported from the memory of the planning system to the arithmetic unit. The arithmetic operations which are involved in the optimization are relatively simple and do not use the capacity of modern processors. It is concluded that the planning algorithm which has been proposed more than twenty years ago is not well suited for modern computers. Since it is the most commonly used algorithm for IMRT therapy plan optimization in clinical and research environments still today, an efficient implementation is proposed in section 4.2, 4.2 and 4.4. Each section presents a solution for single-processor system, multi-processor system and a distributed cluster system, respectively. The architectures of these systems as well as an efficient strategy for implementing the plan generation algorithm are discussed. Nearly all CPU-based desktop and server system currently available on the market can be subsumized under one of these three classes. The basic idea of all these implementation is to optimize the data transport between the memory and the CPU of the planning system. Since every class exhibits a different memory-CPU architecture, three slightly different optimal strategies are used.
On the single processor system it is proven that all the available memory bandwidth can be exploited. Based on the analysis in section 4.1 which stated that the planning problem is heavily memory-bound it can be con-

cluded that there is no other implementation that yields a significantly faster execution time of the IMRT plan optimization. It is proven that the proposed implementation is optimal according to the realization on modern CPU-based computers. The total planning time for real clinical cases is in the order of a few seconds (see Tab. 4.1) on a low-cost over the shelf desktop computer. Most commercial therapy planning frameworks need in the order of minutes to generate an optimal treatment plan.
On shared memory multi-processor systems it could be shown that the algorithm scales well with the higher number of memory channels these systems provide. These multi-processor systems are sophisticated architectures and estimating the maximal bandwidth is a complicated endeavor. It is not enough to count the memory lanes and multiply it by there theoretical bandwidth. The actual performance also depends on the mainboard and memory configuration of these systems. A sound estimation would require a deep technical understanding of these systems which is not subject to this work. However, it is shown that the data bandwidth of the proposed algorithm equals the practical bandwidth which is achieved by the well known Stream memory benchmark [1] which is widely accepted in the community. The fact that the Stream benchmark did not yield a significantly higher performance is a strong indicator that the proposed implementation of the planning algorithm also fully exploits the capabilities of the shared memory multi-processor system. In numbers, the implementation reaches a peak bandwidth of about 80 GB/s of clinical data transfer which results in a runtime of less than 3 seconds for most clinical cases. This can be used for extensive planning problems, e.g. for solving the beam orientation selection (BAS) problem (Bangert et al. 2012; 2013). While conventional IMRT planning assumes a pre-defined static incident beam setup, BAS tries to find the optimal set of incident beam angles, too. Thus, an optimization (finding the beam angles) of an optimization (finding the optimal plan given a fixed set of beam angles) is performed for which several thousand of therapy plans have to be optimized. This is only possible if the plan generation is fast enough and (even more important) the optimization runtime does not explode while handling the extreme huge data set of all these possible plans. Both is achieved with the ultra-fast planning algorithm proposed in section 4.2
On cluster systems the performance of the therapy planning algorithm did not scale with the number of computational entities. It turns out that the granularity of parallelization is too high and the synchronization between the processes hampers the massively parallel ansatz from gaining performance. Even on HPC network technologies the synchronization processes prevail and only moderate speedups can be achieved.
In section 4.7 a general conclusion of the IMRT planning method using Newton's optimization on pre-calculated dose data is drawn. It is stated that this method is outdated from both a computational point of view and from a clinical application point of view. From a computational point of view the memory-bound character of this method limits the performance scaling on HPC hardware. Arithmetic performance increases at a much higher rate than bandwidth. In the near future it is not expected that memory transfers are getting significantly faster. In theory it is possible to build specialized shared-memory systems with a lot more bandwidth capacities, although at a inappropriate price. This endeavor inevitably hits a cost wall. From the clinical aspect there are three main drawbacks of the conventional plan optimization method. First, the objective function contains a set of clinically meaningless parameters which makes planning difficult. The prescribed dose which is one set of parameters is intuitive and meaningful to the therapist. Unfortunately, in general cases not all the prescriptions imposed by the therapist can be fulfilled due to the physical nature of radiation. The trade-off is expressed by a penalty factor which states the importance of a dose prescription. The penalty factor is a relative number which has to be found by an trial-and-error approach during planning. The therapist starts with a set of parameters and generates the first plan which is mathematically optimal due to the definition of the objective function. These plans are usually not clinical optimal (not even clinical acceptable) so the therapist refines the parameters and optimizes another plan until the result is satisfying. In general it can be concluded that the capabilities to transpose the clinical knowledge of the therapist into the plan optimization are very limited and rudimentary. Second, the definition of local features of the plan is only limited to the pre-segmented VOIs. It is not possible to control the prescribed dose of only a part of the VOI or on individual voxels. Third, the conventional planning concept is not suited to adapt a therapy plan according to a new patient geometry which is acquired nowadays

[1] http://www.cs.virginia.edu/stream/

prior to the therapy session due to advanced imaging technique. An adaptation can only be considered by a full re-planning from scratch. With the help of the efficient implementation concepts introduced earlier the planning time is not a problem anymore. However, a full re-planning also implies several quality insurance steps which consume valuable treatment time. To avoid that, a local adaptation of an existing plan is desired.

The new treatment planning paradigm introduced in chapter 5 is designed to overcome these drawbacks. It does not rely on an optimization techniques as the conventional method. Consequently, there is no need for a mathematical objective function which is the bottleneck of clinical knowledge transfer from the therapist to the planning algorithm. The IDS concept provides a set of tools to impose local planning features directly into the plan. This can either be used to adapt an already optimized plan or to compile a new plan from scratch by imposing a sequence of local planning features. Each local feature is implemented into the plan interactively. That means on the one hand, that the therapist gets full control over how this local feature has to look like. Due to a number of dose painting or dose shaping tools, the therapist can directly model the dose distribution of the therapy plan. One the other hand, each local feature is implemented in real-time into the plan. This enables the therapist to immediately evaluate the effect of imposing the requested local dose feature on the whole plan. The real-time character of this method requires that the clinical impact of the requested plan feature has to be presented to the therapist within 1 second. This is a strong requirement for the underlying algorithm (i.e. the IDS engine). Thus the concepts of IDS had to be designed by having the computational capabilities of modern computer system in mind.

The realization of the IDS planning paradigm consists of three main parts: The dose Variation and Recovery (DVR) strategy, a new method for implementing a localized modulation of the fluence map into the dose distribution of the plan and the design of a new treatment planning framework which hosts the graphical user interface, the DVR strategies and the new dose update calculation module.
The dose Variation and Recovery module described in section 5.2 implements a local dose feature which is requested by the therapist. To accomplish that, the DVR module follows a two-step strategy: First, a fluence modulation (patch) is calculated which implements the requested local feature into the plan (Variation). This will inevitable also affect the dose distribution in other areas of the therapy plan. To compensate for these unintentional changes the Variation is followed by an automated Recovery process. The Recovery selects a number of voxels from these other areas of the plan and tries to re-install the original dose distribution. Using this two-step approach of imposing a desired Variation and subsequently Recover for inevitable global changes, it is possible to implement local features into an intensity modulated therapy plan. In order to implement a typical Variation, a Recovery of approximately 30 localities in the plan has to be carried out. The effect of each recovered locality has to be evaluated by an update of the dose distribution. Consequently, the Recovery and the Variation perform approximately 30+1 dose calculation updates until the result of the DVR operation is presented to the therapist. To accomplish this task in real-time (not more than 1 second) the DVR strategies and the dose calculation must be extremely fast.
The IDS dose update calculation is the second part of the IDS engine and at the same time one of the most time consuming parts. It is based on the method of singular value decomposed pencil beam kernels. On the one hand, the new dose calculation concept is able to update the dose distribution according to localized fluence Manipulation patches which are generated by the Variation and Recovery process within only a few milliseconds. The performance of this concept comes from a very efficient realization of the pencil beam kernel convolution on modern processors and by using a B-spline model to reduce the number of operations. On a two core Intel i7-2620M mobile processor the fluence Manipulation patches for nine beams of a prostate plan is implemented into the dose distribution in less than 30 ms. This translates into a speed up factor of about 20 compared to a performance oriented implementation of the conventional pencil beam dose calculation algorithm on server hardware. On the other hand, the newly designed dose calculation module is able to deliver the result in a given time constraint. By adjusting the range and the number of the convolution operations it is possible to trade runtime for dose calculation quality. This fits perfectly into the concept of IDS.

The IDS engine together with a powerful graphical user interface was implemented into a new IDS planning framework. This framework also contains the ultra-fast conventional IMRT planning module which was introduced in chapter 4 of this work. Designing a completely new planning framework for IDS was necessary due to two reasons: First, there is no available planning framework which provides a powerful three-dimensional user interface that displays the current plan and allows to modify local features of the plan. This stands to reason, because the conventional IMRT planning concept uses a parameter based interface which only allows the planner to prescribe certain dose values to pre-segmented organs. Second, conventional planning frameworks are not designed to fulfill real-time constrains. Planning is conventionally an algorithm which runs for minutes without a user interaction. Developing the new IDS planning framework was a challenging engineering task. The key features of the IDS engine alone comprise approximately 17000 lines of code. A variety of modern technologies and programming interfaces like OpenMP, SSE, AVX, CUDA, QT and Ogre3D had to be utilized to realize the IDS concept.

This work demonstrates the feasibility of the Interactive Dose Shaping paradigm for three-dimensional clinical planning. On the example of a typical prostate case and an intracranial case it could be shown that conventionally optimized therapy plans can be reproduced by using the IDS planning framework. The IDS strategies have been implemented on modern computer architectures and operate on a real-time scale making the interactive planning character feasible. As a conclusion it can be stated that the Interactive Dose Shaping paradigm is a potential candidate for dynamic IMRT planning scenarios as an alternative to the conventional optimization method.
To evaluate the full potential of the IDS concept a clinical study has to be launched which involves experienced therapist using the IDS framework in a clinical environment. It has to be investigated if the IDS planning concept is accurate enough to implement the request of the therapist on the one hand and if the concept is generally applicable to all clinical planning cases. For that the IDS framework must be ported to a multi-processor server system. All runtimes stated in this work were retrieved on a two core Intel i7-2620M processor. Typical server environments provide 20 times more computational performance. Since all relevant IDS algorithms were designed in the view of modern hardware architectures it is expected that this potential can be exploited. Unfortunately, there was no appropriate workstation available at the time when the IDS concepts have been designed.
Apart from that, the IDS engine must be developed further to guide the therapist through the interactive planning process. With the current implementation the therapist has to request a series of local dose feature on his own to generate a clinical therapy plan. In future work, a guidance module should be designed which proposes certain beneficial Variation steps to the planner and warns the planner if a dose feature request imposes an unacceptable turbulence to the plan.

# A. $\mu$KonRad: Selected pseudo code and source code

**Algorithm A.1** Physical dose calculation for one bixel

1: **function** DoseCalculation(*bixel* $j$, *doseCube* $d$, *matrix* $D_{ij}$, $w$)

**input:** *bixel* $j$, one bixel from the therapy planning setup
*DoseCube* $d$, the cube describing the dose distribution in the patient
*matrix* $D_{ij}$, pre-calculated dose influence matrix
$w$, weighting factor of *bixel* $j$

**local variables:** $l$, index of the dose influencing elements of bixel $j$ in $D_{ij}$
$i$, index to a voxel in the dose cube influenced by element $l$
$dij$, dose contribution value of bixel $j$ to voxel $i$

2: $m \leftarrow$ `numberOfElements`(*bixel* `j`) ▷ number of elements for current bixel
3: **for** $l = 1, ..., m$ **do** ▷ process all bixel elements
4: $i \leftarrow D_{ij}[l]$.`voxelRef`
5: $dij \leftarrow D_{ij}[l]$.`influence`
6: $d[i] \leftarrow d[i] + dij * w$ ▷ increment dose in voxel i by weighted dose contribution from bixel $j$

Listing A.1: IMRT planning data structures

```
1 dijData = array[0..nrOfSlice, 0..nrOfBeams, 0..nrOfBixel] of struct{
2   float data[4];
3   int index[4];
4 }
5
6 dCube = array[0..nrOfSlice, 0..voxelIndex] of double;
```

Listing A.2: Physical dose calculation using dynamic slice scheduling

```
1 index: array[4] of int;
2
3 #pragma omp for schedule(dynamic) ordered
4 for (int runSlice = 0; runSlice < slice_quan; runSlice++)
5  for (int runBeam = 0; runBeam < beam_quan; runBeam++)
6   for (int runBixel = 0; runBixel < bixel_quan[runBeam]; runBixel++)
7    for (int runElement = 0; runElement < reg_quan; runElement++)
8    {
9     for (int runI = 0; runI < 4; runI++)
10    {
11       index[runI] = dijData[runSlice][runBEam][runBixel][runElement].index[runI];
12       data[runI] = dijData[runSlice][runBEam][runBixel][runElement].data[runI];
13    }
14    dCube[runSlice][index[runI]] = dCube[runSlice][index[runI]] + data[runI];
15   }
```

Listing A.3: Physical dose calculation using libNUMA functions

```
1 int thread_id = omp_get_thread_num();
2
3 setUpNumaNodeBinding(thread_id);
4 for (int runSlice = thread_id; runSlice < slice_quan; runSlice = runSlice +
      thread_quan)
```

# B. Interactive Dose Shaping: Selected source code

Listing B.1: Pencil beam convolution in spatial domain

```
1  int startY, endY, startX, endX;
2  float* fluenceSample_p;
3  long sampleMat_offset;
4  float** kernel0_p = beam_param[beam_nr].kernel_plane[0];
5  float** kernel1_p = beam_param[beam_nr].kernel_plane[1];
6  float** kernel2_p = beam_param[beam_nr].kernel_plane[2];
7
8  init(startY,EndY);
9  for (int runY =startY; runY < endY; runY++)
10 {
11   init(sampleMat_offset);
12   init(startX,EndX);
13   for (int runX = startX; runX < endX; runX = runX + 8)
14   {
15     fluenceSample_m256 = _mm256_loadu_ps(fluenceSample_p+sampleMat_offset+runX);
16
17     kernel_m256 = _mm256_loadu_ps(kernel0_p[runY] + runX);
18     conv0_m256 = _mm256_fmadd_ps(fluenceSample_m256,kernel_m256,conv0_m256);
19
20     kernel_m256 = _mm256_loadu_ps(kernel1_p[runY] + runX);
21     conv1_m256 = _mm256_fmadd_ps(fluenceSample_m256,kernel_m256,conv1_m256);
22
23     kernel_m256 = _mm256_loadu_ps(kernel2_p[runY] + runX);
24     conv2_m256 = _mm256_fmadd_ps(fluenceSample_m256,kernel_m256,conv2_m256);
25   }
26 }
```

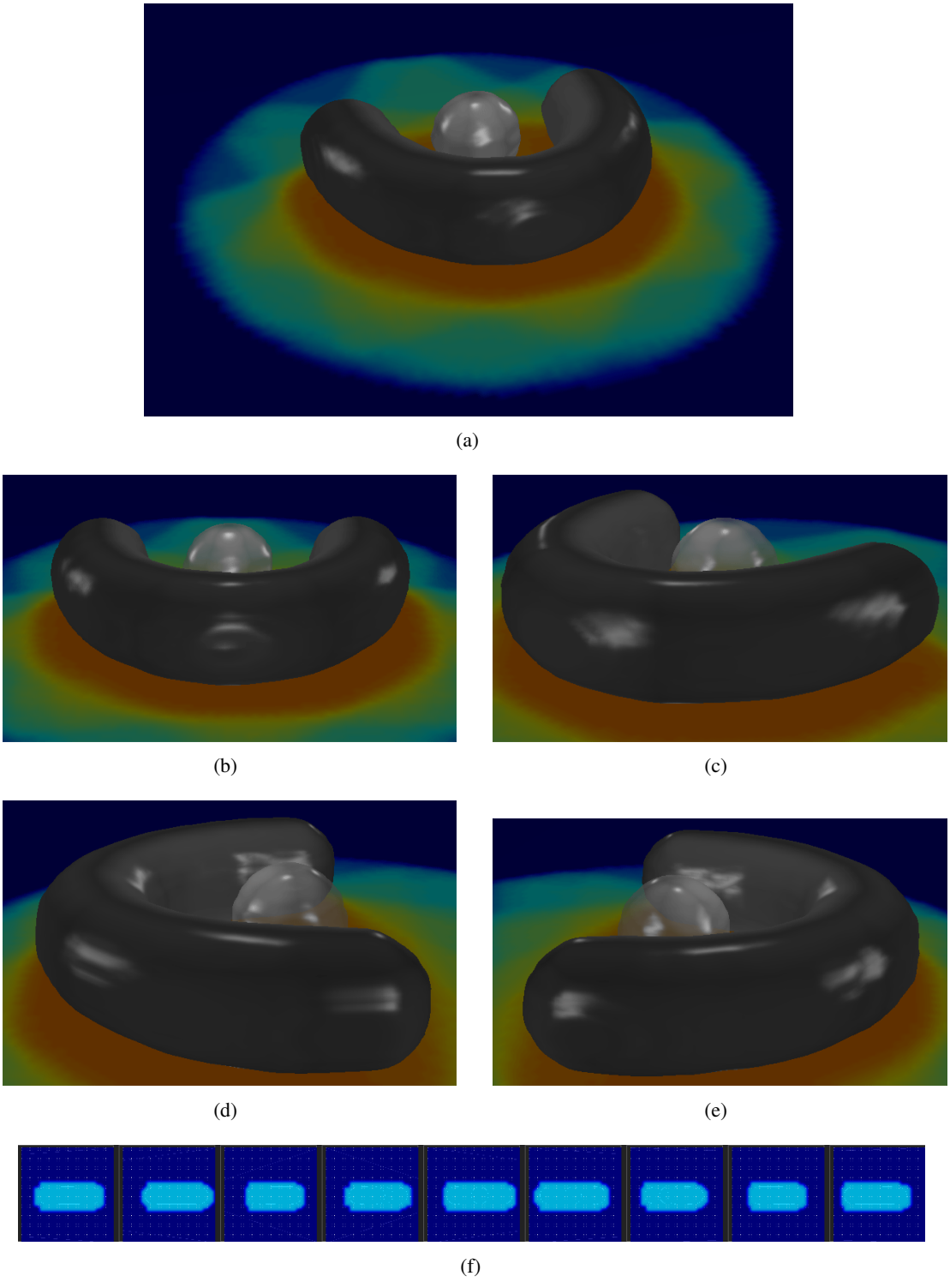
Listing B.2: Efficient matrix multiplication with band value check

```
__m128 aik_m128;
__m128 c_m128;
__m128 b_m128;

for (int k = 0; k < dim; k++)
{
  for (int i = 0; i < dim; i++)
  {
    const float aik = A[dim * i + k];
    if (rangCheck(aik)==true)
    {
      aik_m128 = _mm_set1_ps(aik);
      int startJ = 0;
      int endJ = dim;

      for (int j = startJ; j < endJ; j = j + 4)
      {
        c_m128 = _mm_loadu_ps(&C[dim * i + j]);
        b_m128 = _mm_loadu_ps(&B[dim * k + j]);
        c_m128 = _mm_add_ps(_mm_mul_ps(aik_m128,b_m128),c_m128);
        _mm_storeu_ps(&C[dim * i + j],c_m128);
      }
    }
  }
}
```

Listing B.3: Efficient patient system to collimator system transformation using homogeneous coordinates

```
for (int runZ = 0; runZ < getDimZ(); runZ++)
{
  col23 =  M[beam].Col[2]*z + M[beam].Col[3]*one;

  for (int runY = 0; runY < getDimY(); runY++)
    for (int runX = 0; runX < getDimX(); runX++)
    {
      if (getVOIs()->getDataAtGridCoord(runX, runY, runZ) > 0)
      {
        __m128 x = _mm_set1_ps((x_patsys+0.5f) * voxelSizeX);
        __m128 y = _mm_set1_ps((y_patsys+0.5f) * voxelSizeY);

        __m128 res = col0*x+col1*y+col23;

        __m128 correction = _mm_set_ps(SAD/(SAD-res.m128_f32[1]),SAD/(SAD-res.m128_f32
            [1]),1,1);

        __m128 res_n = res*correction;

        x_pi = round(res_n.m128_f32[3]/(pixelSize))+r;
        y_pi = round(res_n.m128_f32[2]/(pixelSize))+r;


        increaseTpiVoiValue(beam, x_pi, y_pi, d(runX,runY,runZ));
    }
  }
}
```

# C. Therapy plans



(a)

(b)

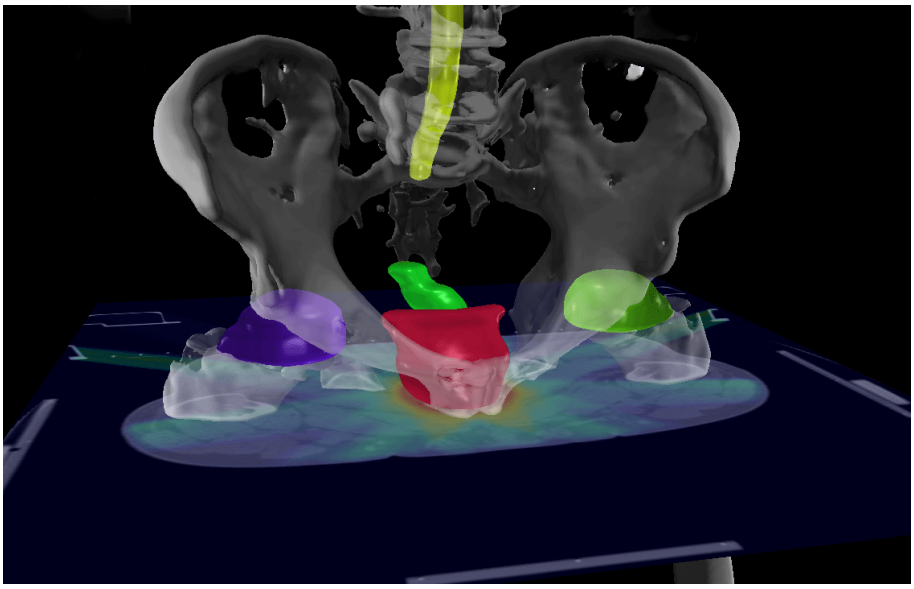
(c)

(d)

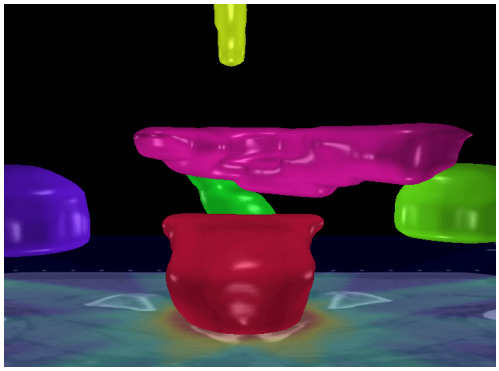
(e)

(f)

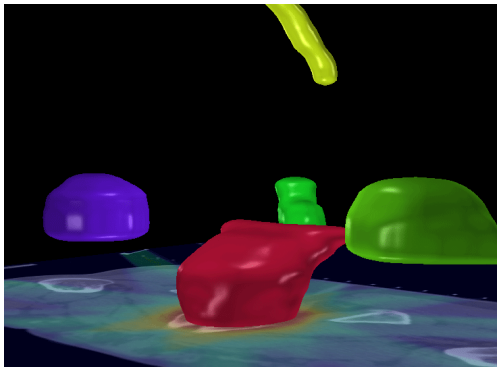
**Figure C.1.:** The geometry of the horse-show shaped phantom case is shown. a) A spherical organ at risk is situated in the emargination of a horse-shoe shaped target volume. The slightly tilted beams eye view is shown for b) beam 0 c) beam 1 d) beam 2 and e) beam 7. f) The fluence amplitudes realizing a homogeneous irradiation of the target volume.



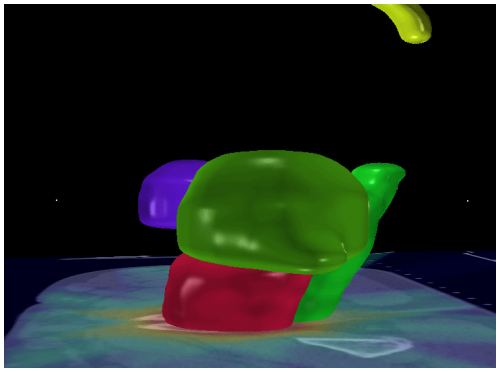
(a)



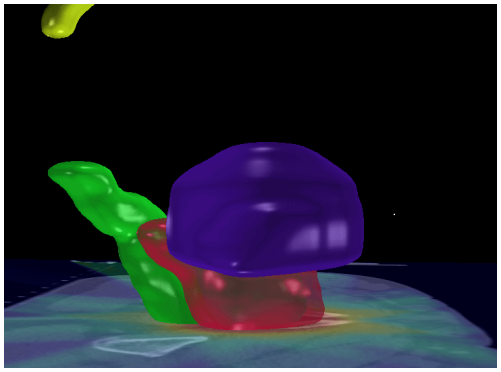
(b)



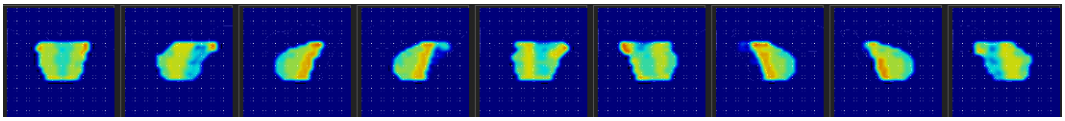
(c)



(d)

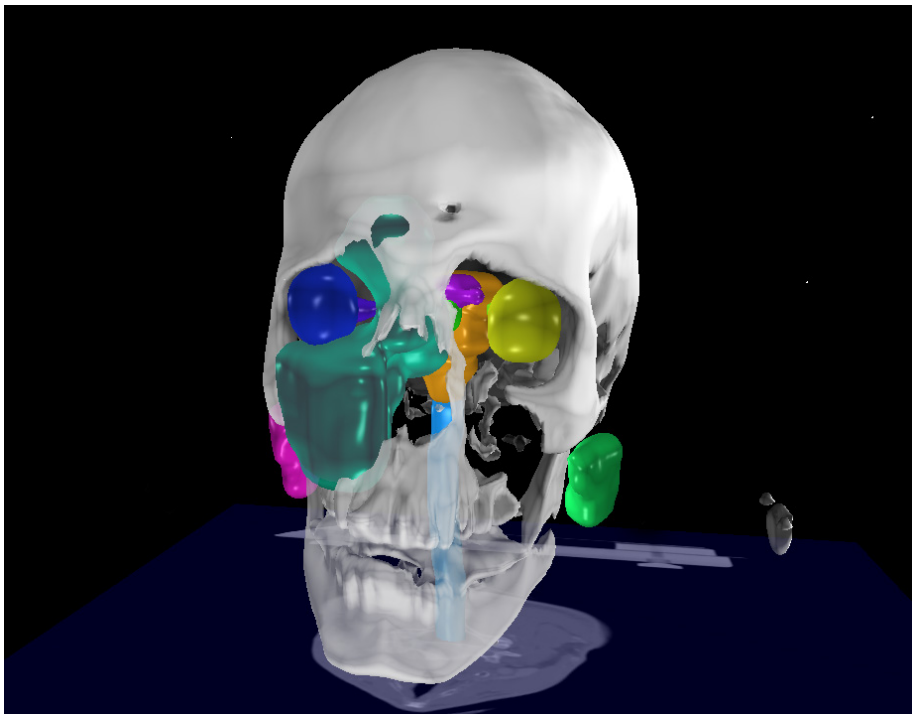


(e)

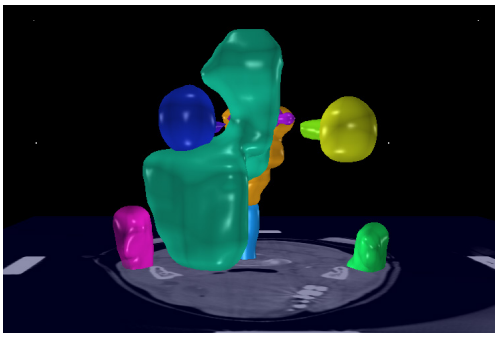


(f)

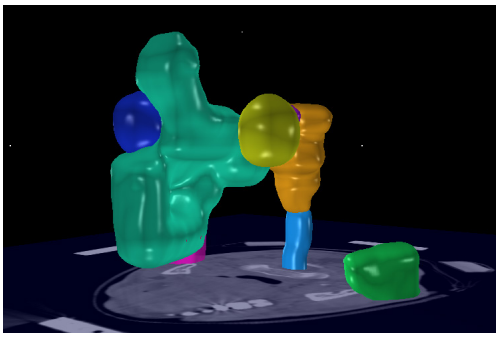
**Figure C.2.:** The geometry of a clinical prostate treatment case is shown. a) The target volume (red) is situated in close proximity to the rectum (green) of the patient. The slightly tilted beams eye view is shown for b) beam 0 c) beam 1 d) beam 2 and e) beam 7. f) The fluence amplitudes realizing a clinical photon treatment plan for the nine beams.



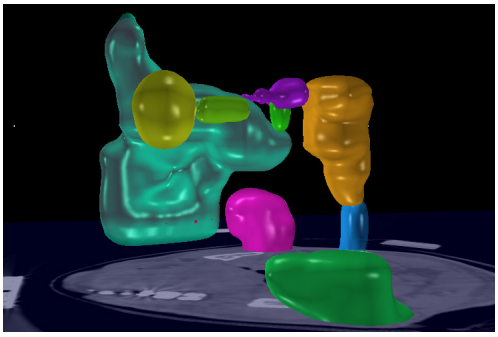
(a)



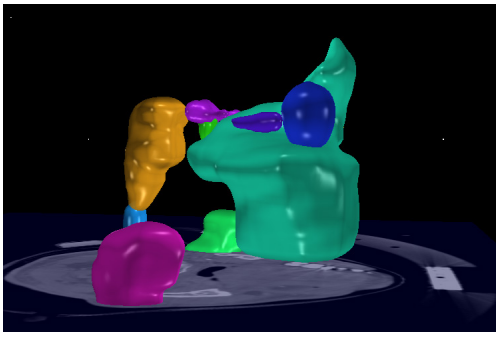
(b)



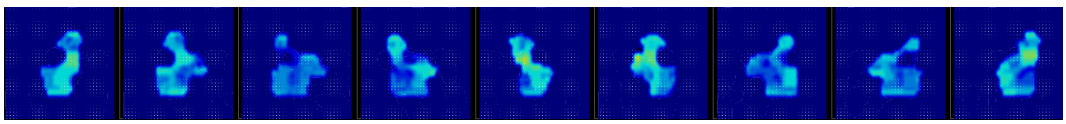
(c)



(d)



(e)



(f)

**Figure C.3.:** The geometry of a clinical intracranial treatment case is shown. a) The skull was partially removed in this image to reveal the target volume (green) which partially range between the right eye (blue) and the left eye (yellow) of the patient. The slightly tilted beams eye view is shown for b) beam 0 c) beam 1 d) beam 2 and e) beam 7. f) The fluence amplitudes realizing a clinical photon treatment plan for the nine beams.

# Bibliography


J. Backus. Can programming be liberated from the von neumann style?: a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978. 21

M. Bangert. *New concepts for beam angle selection in IMRT treatment planning: From heuristics to combinatorial optimization*. PhD thesis, Univeristy of Heidelberg, 2011. 15

M. Bangert, P. Ziegenhein, and U. Oelfke. Characterizing the combinatorial beam angle selection problem. *Physics in medicine and biology*, 57(20):6707–23, Oct 2012. doi: 10.1088/0031-9155/57/20/6707. 33, 82

M. Bangert, P. Ziegenhein, and U. Oelfke. Comparison of beam angle selection strategies for intracranial imrt. *Medical physics*, 40:011716, 2013. 33, 82

S. M. Bentzen. Radiation therapy: intensity modulated, image guided, biologically optimized and evidence based. *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*, 77(3):227–30, Dec 2005. doi: 10.1016/j.radonc.2005.11.001. 11

M. Birkner, D. Yan, M. Alber, J. Liang, and F. Nüsslin. Adapting inverse planning to patient and organ geometrical variation: algorithm and implementation. *Medical physics*, 30:2822, 2003. 21

T. Bortfeld. Imrt: a review and preview. *Physics in medicine and biology*, 51(13):R363–79, Jul 2006. doi: 10.1088/0031-9155/51/13/R21. 11

T. Bortfeld, W. Schlegel, and B. Rhein. Decomposition of pencil beam kernels for fast dose calculations in three-dimensional treatment planning. *Medical physics*, 20:311, 1993. 10, 17, 18, 47, 52, 58

A. Brahme, J.-E. Roos, and I. Lax. Solution of an integral equation encountered in rotation therapy. *Physics in medicine and biology*, 27(10):1221, 1982. 12

C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. 12

Y. Censor, M. D. Altschuler, and W. D. Powlis. A computational solution of the inverse problem in radiation-therapy treatment planning. *Applied Mathematics and Computation*, 25(1):57–87, 1988. 12

C. Cotrutz and L. Xing. Imrt dose shaping with regionally variable penalty scheme. *Medical physics*, 30:544, 2003. 39

D. L. Craft, T. F. Halabi, H. A. Shih, and T. R. Bortfeld. Approximating convex pareto surfaces in multiobjective radiotherapy planning. *Medical physics*, 33:3399, 2006. 38

L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998. 76

P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980. 62

A. de la Zerda, B. Armbruster, and L. Xing. Formulating adaptive radiation therapy (art) treatment planning into a closed-loop control framework. *Physics in medicine and biology*, 52(14):4137, 2007. 21

R. Demming and D. J. Duffy. *Introduction to the Boost C++ Libraries; Volume I-Foundations*. Datasim Education BV, 2010. 76

R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970. 12

M. Frigo and S. G. Johnson. Fftw: An adaptive software architecture for the fft. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1381–1384. IEEE, 1998. 19

W. Fu, Y. Yang, N. J. Yue, D. E. Heron, and M. S. Huq. A cone beam ct-guided online plan modification technique to correct interfractional anatomic changes for prostate cancer imrt treatment. *Physics in medicine and biology*, 54(6):1691, 2009. 21

A. Godley, E. Ahunbay, C. Peng, and X. A. Li. Automated registration of large deformations for adaptive radiation therapy of prostate cancer. *Medical physics*, 36:1433, 2009. 21

D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970. 12

K. Goto and R. A. Geijn. Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software (TOMS)*, 34(3):12, 2008. 54, 55

X. Gu, D. Choi, C. Men, H. Pan, A. Majumdar, and S. B. Jiang. Gpu-based ultra-fast dose calculation using a finite size pencil beam model. *Physics in medicine and biology*, 54(20):6287, 2009. 19

T. Halabi, D. Craft, and T. Bortfeld. Dose–volume objectives in multi-criteria optimization. *Physics in medicine and biology*, 51(15):3809, 2006. 38

S. Hissoiny, B. Ozell, and P. Després. A convolution-superposition dose calculation engine for gpus. *Medical physics*, 37:1029, 2010. 46

D. A. Jaffray, J. H. Siewerdsen, J. W. Wong, A. A. Martinez, et al. Flat-panel cone-beam computed tomography for image-guided radiation therapy. *International journal of radiation oncology, biology, physics*, 53(5): 1337–1349, 2002. 38

X. Jia, X. Gu, Y. J. Graves, M. Folkerts, and S. B. Jiang. Gpu-based fast monte carlo simulation for radiotherapy dose calculation. *Physics in Medicine and Biology*, 56(22):7017, 2011. 46, 60

X. Jia, J. Schümann, H. Paganetti, and S. B. Jiang. Gpu-based fast monte carlo dose calculation for proton therapy. *Physics in medicine and biology*, 57(23):7783, 2012. 46, 60

P. J. Keall and P. W. Hoban. Superposition dose calculation incorporating monte carlo generated electron track kernels. *Medical physics*, 23:479, 1996. 46

W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987. 76

W. Lu, M. Chen, Q. Chen, K. Ruchala, and G. Olivera. Adaptive fractionation therapy: I. basic concept and strategy. *Physics in medicine and biology*, 53(19):5495, 2008. 21

T. Mackie, J. Scrimger, and J. Battista. A convolution method of calculating dose for 15-mv x rays. *Medical physics*, 12:188, 1985. 18

C. Men, X. Gu, D. Choi, A. Majumdar, Z. Zheng, K. Mueller, and S. B. Jiang. Gpu-based ultrafast imrt plan optimization. *Physics in medicine and biology*, 54(21):6565, 2009. 35

R. Mohan, X. Zhang, H. Wang, Y. Kang, X. Wang, H. Liu, K. K. Ang, D. Kuban, and L. Dong. Use of deformed intensity distributions for on-line modification of image-guided imrt to account for interfractional anatomic changes. *International Journal of Radiation Oncology* Biology* Physics*, 61(4):1258–1266, 2005. 21

S. Nill. *Development and application of a multi-modality inverse planning system*. PhD thesis, University of Heidelberg, 2001. 14, 28, 74, 81

J. Nocedal and S. J. Wright. *Numerical optimization*. Springerverlang, USA, 1999. 15

C. Nutting. Intensity-modulated radiotherapy (imrt): the most important advance in radiotherapy since the linear accelerator? *The British journal of radiology*, 76(910):673, Oct 2003. 11

P. Z. U. Oelfke. Radiation treatment planning system and computer program product, apr 2012. EP Patent 2,440,289. 81

W. Ogryczak. Multiple criteria optimization and decisions under risk. *Control and Cybernetics*, 31(4), 2002. 38

D. A. Patterson and J. L. Hennessy. *Computer organization and design: the hardware/software interface*. Morgan Kaufmann, 2008. 76

B. Raaymakers, J. Lagendijk, J. Overweg, J. Kok, A. Raaijmakers, E. Kerkhof, R. van der Put, I. Meijsing, S. Crijns, F. Benedosso, et al. Integrating a 1.5 t mri scanner with a 6 mv accelerator: proof of concept. *Physics in medicine and biology*, 54(12):N229, 2009. 21

A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern recognition*, 1(1):33–61, 1968. 62

H. R. Schwarz and N. Köckler. *Numerische mathematik*. Springer DE, 2006. 50, 51

D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970. 12

R. L. Siddon. Fast calculation of the exact radiological path for a three-dimensional ct array. *Medical physics*, 12:252, 1985. 19

R. L. Siddon et al. Solution to treatment planning problems using coordinate transformations. *Med. Phys*, 8(6): 766–774, 1981. 20

M. Siggel. Entwicklung einer schnellen dosisberechnung auf basis eines pencil-kernel algorithmus fuer die strahlentherapie mit photonen. Master's thesis, University of Heidelberg, 2008. 69

M. Siggel, P. Ziegenhein, S. Nill, and U. Oelfke. Boosting runtime-performance of photon pencil beam algorithms for radiotherapy treatment planning. *Physica Medica*, 28(4):273–280, 2012. 19, 58, 70, 74

A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating system concepts*. J. Wiley & Sons, 2009. 26

M. Sonka, V. Hlavac, R. Boyle, et al. *Image processing, analysis, and machine vision*. PWS Pub. Pacific Grove, second edition edition, 1999. 63

S. Webb. Optimisation of conformal radiotherapy dose distribution by simulated annealing. *Physics in Medicine and Biology*, 34(10):1349, 1989. 12

R. C. Whaley and J. J. Dongarra. Automatically tuned linear algebra software. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–27. IEEE Computer Society, 1998. 54

C. Wu, R. Jeraj, G. H. Olivera, and T. R. Mackie. Re-optimization in adaptive radiotherapy. *Physics in medicine and biology*, 47(17):3181, 2002. 21, 38

Q. J. Wu, D. Thongphiew, Z. Wang, B. Mathayomchan, V. Chankong, S. Yoo, W. R. Lee, and F.-F. Yin. On-line re-optimization of prostate imrt plans for adaptive radiation therapy. *Physics in medicine and biology*, 53(3): 673, 2008. 21

D. Yan, F. Vicini, J. Wong, and A. Martinez. Adaptive radiation therapy. *Physics in medicine and biology*, 42 (1):123, 1997. 21

P. Ziegenhein. Interactive local dose shaping: A new treatment planning strategy. Master's thesis, University of Heideberg, 2008. 22, 39, 81

P. Ziegenhein and U. Oelfke. Radiation treatment planning system and computer program product, dec 2010. WO Patent 2,010,142,421. 81

P. Ziegenhein, C. P. Kamerling, M. Bangert, J. Kunkel, and U. Oelfke. Performance-optimized clinical imrt planning on modern cpus. *Physics in medicine and biology*, 58(11):3705, 2013. 30, 35

# Acknowledgments

First of all, I would like to thank my supervisor Prof. Dr. Uwe Oelfke for his support, his constant flow of ideas in numerous discussions and his enthusiasm for the topic which was very motivating. Many thanks also goes to Prof. Dr. Wolfgang Schlegel for his willingness to act as a second referee.
This work was carried out at the German Cancer Research Center (DKFZ) in Heidelberg and I wish to thank the Helmholtz International Graduate School for Cancer Research for offering me one of their PhD stipends. I wish to thank my colleges from the Department of Medical Physics in Radiation Oncology. In particular I would like to thank Corijn Kamerling who worked with me on the planning framework. My roommates Katrin Welsch, Hendrik Heinrich and Sofia Celi deserve thanks for laughing at my jokes and providing me a friendly environment. Many thanks to Mark Bangert for his fruitful teamwork an the beam angle selection topic and to Simeon Nill.