Cherenkov Imaging in Linac Quality Assurance and Total Skin Electron Therapy Dose Studies

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

in

Engineering Sciences

by Tianshun Miao

Thayer School of Engineering
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

September, 2020

Examining Committee:

Chairman_____
Benjamin B. Williams

Member_____
Brian W. Pogue

Member_____
Michael Jermyn

Member_____
Petr Bruza

Member_____
Timothy C. Zhu

_____
F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

THIS PAGE IS INTENTIONALLY LEFT BLANK, UNCOUNTED AND UNNUMBERED

Abstract

Radiation therapy is widely used for treating cancer, and the linear accelerator (Linac) is the main tool to delivery this. Linac quality assurance (QA), treatment planning, and dosimetry are important tasks that are done to make sure that the radiation doses delivered to patients are appropriate and safe. This thesis covers the development of novel Cherenkov imaging methods for Linac QA, as well as for dosimetry verification in total skin electron therapy.

Basic Linac radiation field QA checks must be completed to confirm that the field shape precisely matches expectations based on configuration. These field checks were efficiently performed using remote Cherenkov imaging on a flat board. The agreement between the dose, the light field projected by the Linac, and the Cherenkov imaging validated that this type of imaging could be a surrogate dosimetry tool. The major benefit of using Cherenkov was that it is a video rate imaging approach, so that arbitrary shapes and entire treatment plans can be imaged without entering the treatment room.

Imaging Cherenkov light as a correlate of dose in total skin electron therapy (TSET) was examined for patients getting treatment following the Stanford technique, where 6 different patient positions are used during irradiation, with supplemental 3D measurement of the patient's body. This dose distribution was projected onto the 3D body model of each position and recorded as a 2D surface texture map. The cumulative dose distribution was calculated by summing up the texture maps of all positions and displayed back on the body model. Statistical analysis was conducted to determine the overall dose uniformity throughout the whole delivery. A comparison of treatments in the

Stanford and rotary techniques was done to compare the dose uniformity, using an angle-dose relationship based upon Monte Carlo simulations of TSET dosimetry. A treatment plan type simulation predicted the relative dose distributions on the body surface. 3D visualization and statistical analysis of these distributions showed that there is better dose uniformity in the rotational technique.

Imaging of a number of patients undergoing TSET at the University of Pennsylvania allowed for visual confirmation of the ability to image dose with Cherenkov imaging. The approach to visualization and display of Cherenkov with 3D modelling animation makes it more efficient to allow both TSET treatment planning and TSET dosimetry verification.

Acknowledgements

Home moving was tied to my early childhood memory, as I always mocked myself that I had never stayed in the same school for more than three years. Now it has been nine years that I have been in Dartmouth, from later undergraduate period to the finishing line of my PhD study. Dartmouth has left a deep imprint in my heart, more than any school or institution. The campus neighborhood is unique across the states, and what is most unique is the people in Dartmouth. The advisors, the classmates, the colleagues, the roommates, and all other faculty and staff in Dartmouth have facilitated the study, the work and the grow-up in my twenties. I really appreciate my life and the people staying all along with me in Dartmouth.

I would like to first say thank you to my PhD advisor, Prof. Brian Pogue. I still remember the day of our first meeting. I was interviewed by him for the research position at DoseOptics, which was a startup company founded by him just then. I was a newly graduated master student with research experience in computer vision. He was looking for a student with computer vision background, even though later I figured out that he was really looking for the candidates with computer graphics background. I felt relieved that I could meet most of the job requirement in DoseOptics after I started my career. I also picking up the knowledge of computer graphic gradually, which helped me a lot in my PhD study.

After I finish two year's work at DoseOptics, Prof. Pogue accepted me as his PhD student in the lab. During my PhD study, Prof. Pogue has provided much more guidance in my academic work than in DoseOptics. His diligence, academic insight, problem-solving skills, and the project management philosophies, has deeply influenced me. I

have come across numerous challenges in my academic research and project management, and he is always available to help and encourage me to solve the research problems. I am weak in my project management, especially when many tasks are in my to-do list. He has also helped me a lot to figure out the work priority and parallelize the tasks. I am really grateful for Prof. Brian Pogue's mentorship in my PhD study and I think my career will be influenced by him more and more in the future.

Prof. Benjamin Williams is my PhD co-advisor and my committee chair. I would also appreciate him for his continuous support in my study. He has enlightened me with many research ideas, most of which has been incorporated into my thesis work. His academic diligence and carefulness give me huge impact. I would like to say thank you to other committee members of my PhD study. Prof. Micheal Jermyn has helped me a lot in computer programming. The programming knowledge and philosophy that I learned from him will benefit much in my future study and work. Prof. Petr Bruza guided me a lot in my PhD work, especially that his philosophy of "Occam's Razor" and his pursuit of perfectionism has deeply influenced my thinking. Prof. Timothy Zhu is extremely helpful for teaching me how to solve complex problem. The clinical research environment is very complicated and unexpected problems may arise anytime. Prof. Zhu has guided me a lot in dealing with the problems in clinical scenarios. I am sure his mentorship will continue to help me to solve unexpected problems in my future work.

I want to appreciate the help from department of radiation oncology in Norris Cotton Cancer Center of Dartmouth College Medical Center. Prof. David Gladstone, Prof. Lesley Jarvis and Prof. Rongxiao Zhang has helped me a lot to provide me a lot of

guidance and facility my research in clinics. The radiation oncology team also provided a lot of convenience for my research.

I want to say thank you to my fellow lab-mates in the Optics of Medicine Lab, including but not limited to Jacqueline Andreozzi, Ethan LaRochelle, Phuong Vincent, Irwin Tendler, Rachael Hachadorian, Benjamin Maloney, Muhammad Ramish Ashaf, Samuel Streeter, Daniel Alexander, Ronny Rahman, Alberto Ruiz, Xu Cao, Jing Xin, Jason Gunn and Sally Hull. They have helped me a lot through my PhD study, and I am very happy working with them in the same lab, as they always bring joys to the office.

Last but not the least, I would like to thank my parents, Huixuan Liang and Yongchun Miao, for their continuous love and support throughout my life. Our home is always moving, but we, as a family, are always together. You are always in my heard and on my side.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Linear Accelerator	Linac
Quality Assurance	QA
Total Skin Electron Therapy	TSET
Monitor Unit	MU
Multi-leaf Collimator	MLC
Electronic Portal Image Device	EPID
Dose Area Histogram	DAH

Chapter 1: Introduction

1.1 Overview of Medical Linac Quality Assurance

Radiation therapy is widely used in treating tumors and the medical linear accelerator (Linac) is the most commonly used radiation source, providing electron and photon beams for treatment.[1] To make sure that each Linac delivers the expected radiation beam, medical physicists conduct many types of quality assurance (QA) procedures to check the beam configurations.[1,2] These procedures can be classified in a few categories, of which the dosimetry and mechanical QA are the main two. In dosimetry QA, the main checks are the output factor, output constancy, as well as depth and in plane beam profiles.[2] The output factor QA verifies if the radiation beam dose delivered matches the Linac system control input, which is defined in terms of monitor units (MU).[3] This is conducted by measurement of the radiation dose at the depth location where the dose is maximum (D_{max}) for certain beam sizes and comparing the dose measurement with the value of input the MUs.[3,4] In output constancy check, a dose measurement is conducted for the same Linac setup with the input MU values every time to check the if the dose measurement deviates from the normal dose reading given the same MU values.[2] The measurement of depth dose curve and planar radiation dose lateral profile are also standard QA procedures to check the output. In these procedures, dose at different depths is measured along the vertical central axis or at different locations in the same vertical plane.[1,2,5] The measurement is compared with 'golden beam data', which are the expected dose values as verified during the initial commissioning of the device.[6] The dose measurement in different Linac configurations can be conducted in terms of dosimetry

QA procedures. For instance, the output constancy check can be conducted in different gantry angles, following the guidelines of Linac QA.[2]

Dosimetry QA procedures are conducted using the phantoms and dosimeters. The output factor measurement is expected to be conducted annually.[2] Measurements are done in a water tank with an ionization chamber, to achieve high accuracy and constancy.[3] The output constancy QA for electron and photon beams are conducted monthly or quarterly using ionization chamber and solid water phantom due to the easy setup.[2] Beam profile measurements are usually made with the output factor QA. Both the depth dose measurement and the planar dose profile can be extracted using the water tank and ionization chamber. The percent depth dose profile and the planer dose profile can also be measured by using radiation film in different depths of solid water phantoms.[7]

The mechanical QA checks involves validation of mechanical movement modules of the Linac, containing the beam shaping module such as jaw and Multi-leaf collimator (MLC), and positioning modules, such as rotary collimator, rotation of the gantry, and movement of the couch.[8] QA on the beam shaping module checks if the beam pattern matches the input from the Linac console, while QA on the positioning modules checks if positioning readings on these modules matches the reading of other measurement tools. For example, the laser positioning markers are used to locate the Linac isocenter in three dimensions and can be used to verify the couch height and longitudinal readings on the Linac console. The light field from inside the Linac projects the beam shape pattern configured by the Linac jaws and MLCs to the isocenter plane using a white light source inside the Linac head region. This is used to verify where the radiation beam shape and the positioning of the jaws and MLCs matches with the treatment field. Angle

measurement tools, such as protractors and electrical levels, are used to measurement the collimator and gantry angles to verify the rotational position readings shown on the Linac front panel or at the Linac console.

The dosimetry tools can also be used to conduct the QA on the mechanical components of the Linac. Radiation film has been widely used to check the rotation center of the gantry and rotation of the collimator accuracy through star-shot analysis.[9] In this analysis, physicists put one piece of radiation film in the horizontal plane containing the isocenter and rotate the collimator to deliver the thin strip of beam onto the film in different angles. The thin strip of beam is configured by vertical and horizontal jaws. The result shows a film exposure that exhibits a star pattern in the center, where the intersection point represents the rotational center of the collimator.[9] Image analysis can be done to extract the size of the center and decide whether the size measurement is within the tolerance in the guideline.[10] Similarly, the radiation beam size can also be extracted using the film and physicists can check the beam size with the light field size to decide whether they are in consistency.[11] Not only the film, the electronic portal imaging device (EPID) and other electronic dosimeter arrays can serve as the similar tool to verify the radiation beam patterns.[12] The resolution of these devices could be limited by the size of the dosimeters.[13] Thus, medical physicists need to care about the resolution, other specifications of these dosimeters when doing QA procedures.

The Linac QA guideline lists the QA procedures for some other modules, such as functionality of the safety interlocks and the respiratory gating system.[2] The safety interlocks are usually checked daily or monthly by intentionally causing the safety risks, such as pressing the safety button, to check if the procedure enables the safety interlocks.

The respiratory gating systems are checked with a breathing phantom to test if the Linac is triggered by breathing patterns.[14] The QA procedures are also conducted for MLCs. These procedures are similar with the dosimetry and mechanical parts. For example, the radiation film or the EPID can be used to check the MLC shapes and the dynamic movement of the MLCs can be checked using the picket-fence test using one of these dosimeters.[13]

QA test	Description	TG 142 List	Cherenkov feasibility
TG 51 test	Check Correction factor at d = 10cm for 6MV and 16 MV beams	Not listed	No, need water tank and electrometer measurement
Photon Output Consistency	Interpret number of MU from dosimeter reading at d = 10cm	Monthly Dosimetry	No, need solid water phantom and dose measurement at depth
Photon Output Linearity	The linear response between the dose measurement and input MU	Annual Dosimetry	No, need solid water phantom and dose measurement at depth
Photon Dose Rate	Dose reading constancy with same input MU, but different dose rate	Monthly Dosimetry	No, need solid water phantom and dose measurement at depth
Photon Backup MU	Verify backup chamber reading matches with the primary reading	Monthly Dosimetry	No, need solid water phantom and dose measurement at depth
Photon Output factor	Measure relative ratio of dose of different field size in terms of 10x10 field	Monthly Dosimetry	No, need solid water phantom and dose measurement at depth
Enhanced Dynamic Wedge Factors	Measure relative ratio of dose of different wedge in terms of 10x10 field without wedge	Annual Dosimetry	No, need solid water phantom and dose measurement at depth
Photon beam profile flatness and symmetry	Measure relative ratio of dose change at different locations in the plane in terms of center	Annual Dosimetry	Possible, if superficial dose is approved
Electron TG 51	Check Correction factor at d_{max} from 6MeV to 22Mev	Not listed	No, need water tank and electrometer measurement
Electron output Consistency	Measure relative ratio of dose of different depth in terms of dose at d_{ref}	Not listed	No, need water tank and electrometer measurement
Electron output Linearity	The linear response between the dose measurement and input MU	Annual Dosimetry	No, need solid water phantom and dose measurement at depth
Radiation Light Field QA	Distance between the light field center and the crosshair center	Monthly Mechanical	Yes.
Optical Distance Indicator reading	Check if the ODI reading matches with mechanical pointer reading	Daily Mechanical	Possible, can read the ODI directly from cam, but hard to adjust height
Laser Alignment	Check if the laser intersects at iso-center	Daily Mechanical	Yes.
Mechanical/Digital reading Check	Verify if the readings of gantry, couch and collimator angle match.	Monthly Mechanical	Possible for collimator and couch check, hard for gantry check
Radiation isocenter for collimator rotation	Iso center deviation for different collimator angle	Annual Mechanical	Yes, done by star-shot
Radiation isocenter for couch rotation	Iso center deviation for different couch angle	Annual Mechanical	Yes, done by star-shot
Radiation isocenter for gantry rotation	Iso center deviation for different gantry angle	Annual Mechanical	Yes, done by star-shot, but need to move calibration board
Emergency Button and Safety light	Check interlock buttons and safety signs	Daily Safety	Partially Possible, through surveillance
MLC leaf speed	Check the speed of MLC from dynalog file	Monthly MLC	Yes
MLC Quality Check	Check the shape of MLC	Weekly and Monthly MLC	Yes, by picket fence
Imager Quality Check	Check the KV, MV imager	Monthly and Annual imager	No, need to read HU inside phantom

Table 1.1: Procedures performed in the monthly Linac QA as specified by Task Group 142 (TG-142) with the blue boxes marking the potential applications of using optical imaging methods

While some of the dosimetry procedures of Linac QA are performed with radiation film, it usually takes large amount of time and workload to calibrate, read and analyze the film data to get the information directly. The measurement of doses and beam shapes by film can be potentially replaced by remote optical imaging techniques, from scintillation and Cherenkov, which are quick and easy to analyze through automatic image processing software. For example, the imaging system with a scintillation sheet has been used to acquire beam shape information and perform star-shot analysis in an MRI-Linac QA.[15] Table 1.1 summarizes the procedures of monthly QA performed in the Southwestern Vermont Medical Center (with advice from Frank Rafie, Medical Physicist), where some of them could be implemented using Cherenkov imaging. This exercise was a first attempt at evaluating the potential role for this type of remote imaging in a routine QA process.

1.2 Overview of total skin electron therapy

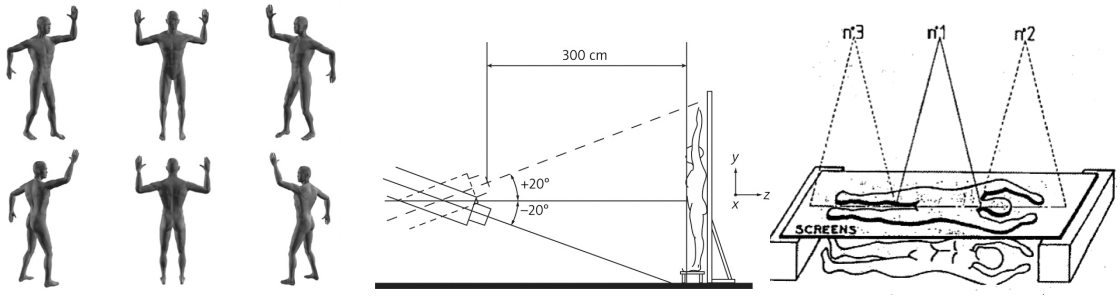
Mycosis fungoides is the most common type of cutaneous T-cell lymphoma, which has symptoms of flat patches, rashes, lesions, thin plagues or even the tumors in the patient's skin.[16] The studies show that the abnormal interaction of the lymphocytes may lead to cancer of the T-cells and cause their malfunction to inhibit tumor development and eliminate foreign microbial agents.[16] Therefore, mycosis fungoides is diagnosed with accumulation of mature T-cells detected in the body surfaces.[17] Mycosis fungoides is rare with incidence of about 0.36 per 100,000 person-years.[17] Clinically, mycosis fungoides and Sezary syndrome, which is a more aggressive type of T-cell cutaneous lymphoma, can be classified in 4 stages (T1—T4), based on percentage skin area of plaque. Patients

with T1 disease have a healthy life expectance, while patients with T2 disease have a median survival of about 10-12 years and 25% risk of progressing into more advanced disease.[17] Therefore, earlier diagnosis and treatment of is important to expand the patient's life expectancy.

Radiation therapy, especially electron beam treatment, is the classical treatment of mycosis fungoides as an aggressive approach to preventing bacterial infection and sepsis over the body surface,[18] because the electron beam dose only accumulates in the superficial skin surface and decays rapidly with deeper tissue regions.[19] Because mycosis fungoides usually occurs on large area or even the whole-body surface, the total skin electron therapy (TSET) was designed to cover the whole body with radiation dose to treat mycosis fungoides. The mycosis fungoides patient stands in front of the radiation source, a few meters away, with different positions to try and ensure that all different parts of the skin are exposed in the radiation beam with nearly equal distribution.[18] There are three main positioning techniques designed to the whole-body skin surface: (i) the Stanford technique, (ii) the rotational technique and (iii) a technique involving the patient being shifted through the beam.[18]

In the Stanford technique, the whole treatment is divided into 6 fields, and is usually conducted in two days with 3 fields in each day. Each field corresponds to one standing position and have two subfields, in which the Linac gantry rotates and the Linac gantry head points up- or downwards to make sure the does distributes uniformly in the plane where the patient stands.[20] A scattering filter, which is commonly a large plastic sheet, can be placed between the patient and radiation source to increase the electron scattering effect and improve the dose uniformity.[18] In some of the standing positions of

this Stanford technique, the patient's body is tilted in the lateral direction, so that the lateral sides of the patient are also covered in the electron beam.[21,22] Some parts of the body area can hardly be exposed in the radiation beam under these positions, such as the perineum, top of head and shoulders, and the soles of the feet. These areas are usually compensated by boost fields, in which the patient is placed on the treatment couch and these areas faces Linac gantry head in shorter distances, or by positioning of scattering blocks above the patient.[21] Some other areas, such as eyes and nails can be shielded with lead covers due to their high sensitivity to radiation dose.[23]



(a) (b) (c)

Figure 1.1: Different positions in the Stanford technique (a),[18] the rotational technique (b)[18] and the lateral patient-shifting technique (c).[24]

In the rotational technique, the patient stands with the fixed position in a rotatory stand, which is moving during the radiation treatment.[25] The stand rotates in a constant speed to make the radiation beam cover most parts of the patient's skin.[26] As in the Stanford technique, there are dual fields used, in which the Linac gantry head points up- and downwards, and the scattering filter can be applied to make sure the dose distributed uniformly in the patient's standing plane.[18] Unlike in the Stanford and rotational techniques, the patient lies in supine and prone positions on the ground in patient-shift

technique. For each position, three adjacent radiation fields are applied coronally to cover all parts of the body and a composite Lucite screen is used to boost the radiation dose in the lateral side of the patient's body.[24] This thesis focused on imaging patients in the Stanford technique, which is widely used in TSET at Dartmouth and University of Pennsylvania. It also explored simulations of the rotational technique as a performance comparison in later chapters.

Dosimetry is important in all kinds of radiation therapies, including TSET. Before the treatment, physicists can use solid water or boards with classical dosimeters, such as diodes and films, to measure the dose distribution in the patient's standing plane.[20] During the treatment, the dosimeters are usually taped to different locations of the patient's body and are read after the treatment to check if the dose readings deviate from the prescribed dose.[21] However, this dose measurement does not cover all parts of the body and some regions, such as the top of the head, are hard to measure because the radiation beam is tangential to these regions.[25] Therefore, dosimetry of TSET is still an open problem that is worth exploring.

1.3 Cherenkov imaging

Cherenkov emission is generated from high speed particles travelling through the dielectric material with the speed higher than light in the medium.[27-29] It is usually observed in the nuclear reactor[30] or cosmological images[31], where high speed particles are abundant due to radiation. Based on the characteristics of the Cherenkov emission, detectors designed to observe Cherenkov light are used in radiation-related research and experiments, such as particle physics experiments and astrophysical observations.[32] Imaging of Cherenkov emission has been used in biomedical research involving Linac

treatments, especially in radiation therapy in a number of centers, and especially well developed at Dartmouth.[33] High energy particles, such as X-ray photons and electrons, travel from the Linac gantry head and can generate the Cherenkov photons through the dielectric media, such as water, and human tissues.[34] The Cherenkov light intensity is low but still detectable through the intensified imaging system.[35] The Cherenkov signal can also be converted to fluorescent or scintillation signals with higher intensity through the phosphor or scintillators in the medium.[29] The study has shown that for the same material, the light intensity of Cherenkov signal, or the fluorescent or scintillation signals derived from the Cherenkov emission are linearly dependent on dose magnitude in a mono-energetic radiation beam and in the absence of tissue attenuation.[28] Therefore, it is feasible to interpret the dose distribution from Cherenkov imaging, with appropriate calibration between the dose value and Cherenkov light intensity.[28]

Based on the Cherenkov-dose relationship, Cherenkov imaging can be used as a tool to perform QA tasks for the medical Linac. Cherenkov imaging in a water tank can help to confirm percentage depth dose curves, that can be used to compare with the golden beam data.[29] Physicists can also derive the 3D dose distribution of the water tank through imaging the Cherenkov signals from the water tank at a range of angular views and with a full filtered back-projection recontruction[36] or by pairing Cherenkov imaging on water tank with EPID measurement.[37] Thus, there are a number of ways that Cherenkov imaging can facilitate some of the QA tasks required.

In addition to imaging in phantoms, Cherenkov signals can also be detected from the human body in radiation therapy,[38] where it is possible to conduct surface dosimetry directly on the patient during the treatment. Due to the scattering opacity of most human

tissues, Cherenkov signals can only be imaged from superficial tissues translucent to the light emission, which is from approximately 4-7mm average depth below the skin surface.[39] This makes it potentially possible to interpret the dose distribution over the patient's skin surface throughout the treatment. For instance, the Cherenkov images can be used to analyze the dose in areas of entrance and exit beam of the breast, head and neck treatments.[40] It has also been used in TSET as a tool to observe the overall dose distribution on the patient's body skin.[41] However, because of the light scatter and absorption of the tissues, the Cherenkov-dose relationship is affected by some factors, such as the tissue properties and curvature of the surfaces. Additional calibration procedures are needed to correct the effects of these factors in deriving the dose from the images of Cherenkov emission.[42,43]

1.4 Cherenkov Light transport in tissue

Biological tissues are the heterogenous materials that have complex interactions with light. The light-tissue interaction involves several physical mechanisms.[44] When travelling through the biological tissues, some of the photons can be absorbed by the electrons in the molecules, elevating them to excited states. When the photons transfer back to the ground state, some of the energy absorbed is dissipated as heat energy,[45],[44] through the process of nonradiative relaxation. The reemission of photons also from fluorescence or phosphorescence. These two processes can be differentiated based on the lifetime of the process. The fluorescence process is in the order of nanoseconds, while the scale of the phosphorescence could be millisecond.[46] The photochemical process happens when the energy transfer to other molecules.[47] The electron in the ground states can also be elevated into a virtual state between the ground and the excited states, which is further

transferred back to the ground state through Raman scattering and reemits the photons, with frequency corresponding to the energy gap between the virtual and the ground states.[48] The absorption in tissue can be attributed mainly to the non-radiative process from hemoglobin, melanin, and water.[49] The hemoglobin and melanin absorb strongly in the spectrum of less than 600nm due to the iron content,[50] while water absorbs strongly in the infrared spectrum.[51]

The light scattering in the tissue mainly originates from the light refraction and reflection from the different structures in the tissue materials,[52] attributed to the nonradiative relaxation with photon reemission when the photon energy transferred from the excited states to the ground states. The photon scattering generated from the nonradiative relaxation incurs no frequency change and is classified as elastic scattering.[44] The photons can also scatter with frequency shift in the process of fluorescence, phosphorescence and Raman scattering, defined as inelastic scattering.[46,48]

Many biomedical instruments have been developed to measure light transport in tissue, and the interpretation is based upon the theory of light-tissue interaction. Oxygen saturation can be measured based upon the different absorption spectra of oxy- and deoxyhemoglobin.[53] The optical scattering mechanism can provide information of the size and number density of the cellular components and structural matrix elements.

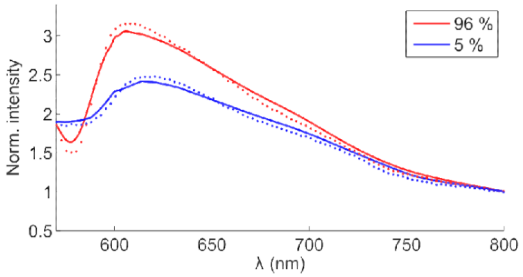


Figure 1.2: Cherenkov spectrum from the human tissue with different oxygen levels[54]

The model of light transport in biological tissue is characterized as the absorption(μ_a) and effective scattering coefficient (μ_s'), representing the physical process of absorption and scattering. Many studies have been conducted to measure the properties of tissue and found that light transport in tissue is dominated in the infrared wavelengths by scattering, as the absorption coefficient is much smaller than the effective scattering coefficient.[55] The scattering of light in the medium can be approximated by Mie theory in the near infrared,[56] and Rayleigh theory when the particle size is much smaller than the light wavelength, such as in the UV and blue wavelengths.[57] Using these models in different ways, the overall picture of light transport can be estimated by the radiative transport model, assuming that scattering is largely elastic and that absorption is independent from the scattering processes.[58] This model based upon the Boltzman transport equation can be simplified further to a diffusion equation with the assumption of the scattering being isotropic in angular phase when observed over long distances. This simplification of diffusive transport provides a convenient method to avoid modeling the angular phase function of the scattering and provides for an equation that yields the fluence rate as the measurable observable.[59] This light transport modeling is accurate over macroscopic distances of several millimeters, and is widely accepted as a reasonable way to estimate light interaction phenomena in biological tissue. The diffusive behavior also causes the spread of light in tissue, lowering the observed spatial resolution seen when light signals are measured from tissue.[44,52]

Mathematical derivation and simulations have been used to establish that within highly scattered media with a high albedo, it can be modelled as an isotopically scattering medium, if the light measurements are all after multiple scattering events or if modeled at

the microscopic level, the anisotropic scattering must be included. This macroscopic observation of the scattering appearing isotopic over long distances is defined as the principle of similarity in transport scattering.[60] In the scattering medium, the phase function is the probability distribution of the scattering angle when the event of the scattering happens. The phase function is experimentally derived but can be approximated and parameterized by many analytical functions, in which Henyey-Greenstein phase function is most widely used and the equation without normalization is given by[61]

$$P_{HG}(\theta, g) = \frac{1 - g^2}{[1 + g^2 - 2gcos(\theta)]^{3/2}}$$

The parameter of *g* characterizes the isotropization of the phase function, in which the value of g is 0 or 1, when the phase function is isotropic or fully forward scattered. After multiple scattering events, the phase function is modeled as[60]

$$P_{HG}{}^{n}(\theta, g) = \frac{1 - g^{2n}}{(1 + g^{2n} - 2g|g^{n-1}|\theta^{3/2})}$$

The phase function converges to the isotropic function as the value becomes uniform with *n* growing large and the phase function getting independent of *g*. Monte Carlo simulations[60] also confirm that the aggregated phase function is almost isotropic even for a highly anisotropic medium after multiple events of scattering, as shown in Fig 1.

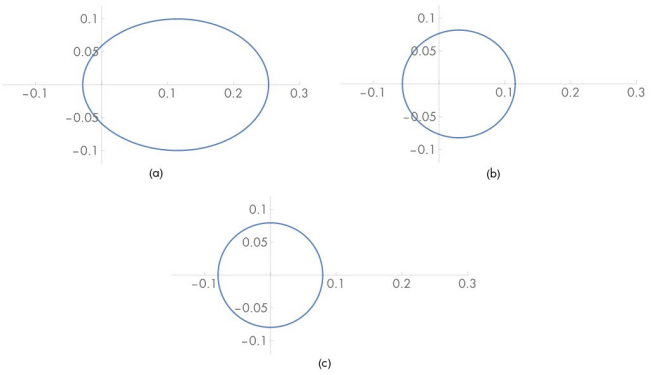


Figure 1.3: The phase function of particles in a high anisotropic medium with g = 0.9 after 100 (a), 1000 (b) and 10000 (c) scattering events[60]

The diffusion theory provides a simplified solution for the light travelling through the homogenous, and highly scattering medium, approximate the medium to be isotropic from the principle of similarity. Light traveling through the medium can be completely modeled by the radiation transport equation[59], given by

$$\frac{1}{c_m}\frac{\partial I(r,\vec{s},t)}{\partial t}$$

$$= -\vec{s}\nabla I(r,\vec{s},t) - (\mu_a+\mu_s)I(r,\vec{s},t) + \frac{(\mu_a+\mu_s)}{4\pi}\int_{4\pi} p\left(\vec{s},\overrightarrow{s'}\right)d\Omega' + Q(r,\vec{s},t)$$

The location in the transport equation is represented by position *r* and direction vector $\vec{s}$, with c_m represents the light of speed in the medium. The left side of the equation characterizes the change of the particle fluence, represented by its first order derivative by the time. In the right side of the radiation transport equation, the first term represents the particle loss in its travelling direction and the second term represents the loss due to scattering and absorption processes, which is linearly dependent on the fluence. The third terms characterize the particle gain due to the absorption and scattering from all

directions, represented by the integral over the solid angle. The last term in the right side represents the gain from the particle source in the location

In most studies, the steady state light fluence is of interest, and in many cases the model can be reduced to 1D. Therefore, in the scattering dominated medium, the transport equation can be reduced to the diffusion equation[39], which is

$$-D\frac{\partial^2\varphi(z)}{\partial z^2}+\mu_a\varphi(z)=S(z)$$

where z is the 1D depth location below the tissue surface and $D=\frac{c_m}{3(\mu_a+(1-g)\mu_s)}$, based on the Henyey-Greenstein phase function. The diffusion equation can be easily solved by either analytical solutions with the boundary definition of the surface light fluence or through numerical solutions.[39].

Like all light sources, Cherenkov photons undergo scattering within human tissue. When radiation particles travels through the biological tissue, the Cherenkov photons are created from the fast ionization-charged particles with a broad spectrum.[33,54] This can be modelled in radiation transport theory, either by solutions to the radiation transport equation [62] or by statistical Monte Carlo simulation.[63] Applications using the Monte Carlo method, such as GAMOS and GEANT4 are based on the sampling of the trajectory of the radiation particles and the interaction coefficients for absorption and scattering.[63] These optical photons generated from the Cherenkov emission undergo absorption and scattering process due to the complex structure of the tissue. The human tissue absorbs the photons in the UV and blue part of the spectrum due to the components of hemoglobin and melanin.[54] Thus, the Cherenkov emission from the human tissue is largely observed in the red and infrared range.[64] The Cherenkov photons scatters within the complex structures and layers of the human tissue, which can be also modelled by the

radiation transport model, which can be approximated by the diffusion equation, as explained earlier, if the tissue parameters are well known.[39] This diffusion of Cherenkov light causes a blurring of the boundary of the radiation beams observed incident upon tissue, as will be examined here.[65] The transmission of the Cherenkov photons modeled by the Monte Carlo method in the GAMOS software, is based on the Geant4 package, and can simulate the light transport of the Cherenkov particles with the plug-in module of the radiation induced light transport in biological media.[66] This tool is a very efficient way to simulate and model Cherenkov generation and transport together from the incident radiation dose.

1.5 3D modeling and computer animation

Three-dimensional (3D) modelling has been widely used in data visualization and image processing in medical imaging world. The 3D or 4D CT images have been used to analyze the location, size and dynamics of tumors.[67] Additionally, MRI images are commonly used for better soft tissue discrimination, or to assess the changes such as the deformations during surgery or assessing injury.[68] In radiation therapy, even though the patient is restrained in most cases, patient movements can still affect the beam delivery either from simple breathing or body motions.[69] The use of 3D modelling has been demonstrated to help in tracking patient movements during treatment.[70] For instance, in the recently developed MRI-Linac systems, 3D imaging is utilized to monitor the patient's breathing to replace respiratory gating systems and deliver the beam precisely to the tumor even with motion present.[71] In this way, the radiation dose can be delivered to the expected organ and not leak into the surrounding tissue regions.

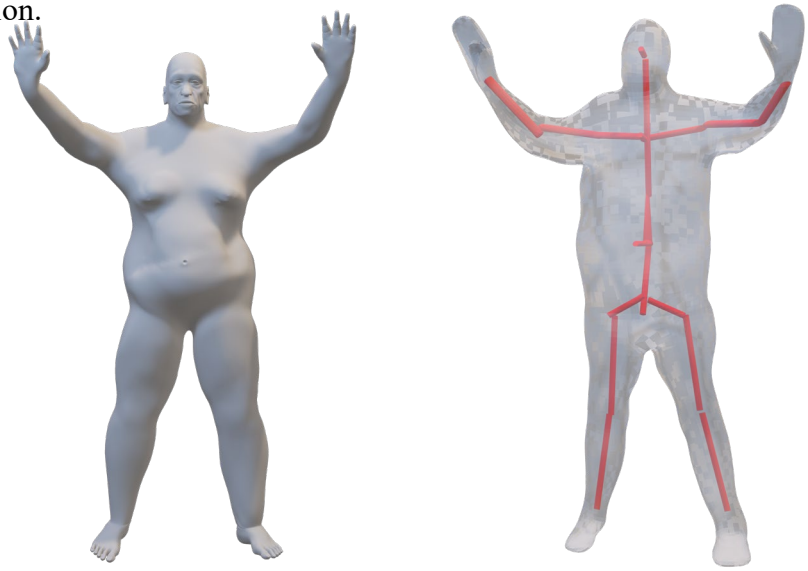
In TSET, the patient's movement is also important and needs to be taken into consideration as the patient changes position in some techniques, such as the Stanford and patient-shift techniques. In classical TSET plans, physicists use stands, handles, and some other markers to minimize motion and mark the patient's position. However, the patient's position is not fixed and minor temporal changes in the patient's position commonly occur, and even may not be observed by the therapists. Thus, the effect of the patient's movement or changes in the patient's position have seldomly been studied on the dose distribution over the patient's surface. A 3D imaging system has also been incorporated in TSET to track the patient's position to check if there are deviations from the designed TSET positions.[21]

The Stanford technique involves the patient's movement during the treatment and computer animation is a tool to analyze the patient's movement. Computer animation techniques are used in movie and gaming industries to model the movement of characters as realistic as possible. The computer animation workflow starts with creation of the mesh-based model of the character, to mimic the shape of the human or the animal, sometimes with artistic effects.[72] Then an animation skeleton is added inside the model of the character and serves as the "bones" of the character, which defines the pivot locations of the joints.[73] In this way, the character can be animated by moving or rotating the skeleton attached to the model. The surface of the model is bound to the animation skeleton through the weight map, representing the influence of the skeleton movement to the surface movement.[74] The weighed map can be represented by the weighted equation[75] as

$$q_i = \sum_j w_{i,j} M_j p_i$$

where *i* and *j* corresponding to the index of the vertices of skin and skeleton, *w* is the weight, *M* is the transformation matrix representing the movement of the animation skeleton. *p* and *q* are the locations of *i* before and after movement.

The weight map can be edited interactively to eliminate the abnormal surface deformation due to skeleton movement. A common practice is to, post process the surface model by 'polishing' using sculpture tools to reduce abnormal deformation after the animation.



(a) (b)

Figure 1.4: The example of the 3D model of a TSET patient in AP position (a), and the animation skeleton (in red) was added to a 3D human model (b).

The 'texture' is an important part of the character and is usually recorded in the 'UV' map, which is a 2D data structure that displays the entire skin surface on a single 2D map in UV space.[76] Each vertex in the 3D model corresponds to one or multiple points/locations in the UV map and one triangular/quadratic shape in the 3D model maps to one similar shape in the UV map. An analogy of the UV map to the 3D model is the 2D world map commonly used to visualize the earth model which represents the map of the world. The strength of the UV map is that it is fixed to the 3D character during the

animation.[76] Therefore, the texture of the 3D character should be unchanged when the character is moving.

1.6 Camera calibration

Many applications of 3D modelling incorporate the 2D image taken by a camera. These applications need to read the camera's information, such as the location and other camera characteristics to correlate the images into a 3D model. The camera's information can be extracted through the procedures of the camera calibration and the information is represented by the calibration matrix.[77] The camera calibration is usually conducted with the images of calibration patterns[78], which contains high precision points of interest that can be easily recognized by computer vision algorithms, such as the Harris corner detector,[79] and measured manually by the researchers. The calibration matrix is then derived from the pixel locations of these points of interest in the image and their real physical locations.[77] The calibration documentation and software are packaged in many open source[80] and commercial software tools.[81] Therefore, it is easy to apply this calibration approach calibrate and create 3D applications, as was done here. This is especially important for complex shapes such as the full human body of patients, where there are the complexities of arms, legs, variations in body habitus, and complications such as tattoos and hair. The 3D modeling done here was all achieved with 2D camera scans of the image field.

1.7 Thesis overview

This thesis is divided into three tasks made in the author's graduate study: a validation study of Linac beam QA based on Cherenkov imaging, TSET dosimetry using Cherenkov imaging and computer animation techniques, and a TSET geometry/technique

simulation study to determine the optimal delivery plan. All three topics are related to radiation therapy and were conducted with Cherenkov imaging and 3D modelling. The work introduced in this thesis built a generic workflow to incorporate 3D surface modelling and Cherenkov imaging in dosimetry analysis of the radiation treatment.

The Linac QA was conducted through the 2D Cherenkov imaging system with a thin slab of plastic phantom located in the horizontal plane containing the isocenter. For 3D localization, a piece of paper printed with the checkerboard pattern was taped to the surface of the slab phantom to extract the camera matrix of the Cherenkov imaging system location. Then the imaging system took the images of the light field and the Cherenkov emission with different jaw sizes. The light field images were taken by turning on the light field switch without a radiation beam, while the Cherenkov images were taken with radiation beam on and the light field switch off. The two types of images were converted to planar distribution using the camera matrix. The sizes of the jaws were derived from the planar distribution and compared with the input sizes in the Linac console. This study validated the feasibility to do routine QA based upon this type of remote dose imaging by Cherenkov. This work was published as "Cherenkov imaging for Linac beam shape analysis as a remote electronic quality assessment verification tool" in Medical Physics by Miao T, Bruza P, Pogue BW, et al.

The TSET dosimetry task was conducted with the Cherenkov imaging system and the 3D body scanner. The body scanner extracted the 3D mesh of the patient's positions of the Stanford technique and the Cherenkov imaging system acquired the images of emission during the TSET delivery. The Cherenkov images were converted to dose distributions with a Cherenkov-dose relationship from calibration, and the Cherenkov

camera matrix was extracted using the camera calibration procedure similar as in the light field study. The patient's 3D model was constructed to fit the 3D mesh of one position and animated to other positions using standard 3D computer modeling techniques. The dose distribution was projected onto the 3D model of each position with the information of the camera matrix and was recorded in the UV map as texture. The cumulative dose distribution was derived by summing up the dose distributions on the UV maps of all positions and visualized in the 3D model of the patient. Statistical analysis was conducted to evaluate the dose distribution and observe the high and low dose region in the patient's body. This work was published as "Computer animation body surface analysis of total skin electron radiation therapy dose homogeneity via Cherenkov imaging" in Journal of Biomedical Imaging in SPIE by Miao T, Petroccia H, Xie Y, et al.

The TSET dose simulation task used 3D human models acquired in the TSET Cherenkov dosimetry study. Before the dose simulation, the Monte Carlo simulation was performed on cylindrical water phantoms to extract the relationship between the relative dose magnitude and the incident angle between the beam and the surface normal, for cylinders with different radii. The 3D models of the patient were manually labelled with curvature radius in different parts of the body. In the process of dose simulation, the normal vector of each vertex of the patient's 3D model was extracted to compare with the incident beam vector and the dose value was calculated based on the angle-dose relationship derived in the Monte Carlo simulation and the label of the curvature radius on that vertex. The dose simulation process was performed on each position of the Stanford technique and the position of each angle in rotational technique. The dose distribution was recorded in the UV map of each position or each angle, and the

cumulative dose distribution were derived by summing up all the UV maps. Statistical analysis and 3D visualization of the dose distribution was conducted to evaluate which of the Stanford and rotation techniques contributed to more uniform dose distribution in the TSET. This work is currently submitted for review to International Journal of Radiation Oncology, Biology, Physics as "Computational Dose Visualization & Distribution Comparison in Total Skin Electron Treatment Illustrates Superior Coverage by the Rotatory Technique" by Miao T, Zhang R, Jermyn M, Bruza P, Zhu TC, Pogue BW, Gladsone DJ, and Williams BB.

Chapter 2: Linac beam shape analysis using Cherenkov imaging

The contents in this chapter is adapted from the following publication.

Miao T, Bruza P, Pogue BW, et al. Cherenkov imaging for Linac beam shape analysis as a remote electronic quality assessment verification tool. *Med Phys.* 2019;46(2):811-821.

2.1 Introduction

Verification of the accuracy of the Linac radiation beam, as shaped by jaws and MLCs, is a critical quality assurance component in clinical radiotherapy. The radiation beam is not visually seen, but the projected light field from the Linac head is used as a surrogate to visualize the projection of the beam on the table, phantom, or patient. The guidelines to verify that the size of beam agrees with prescription are very tight, and so the radiation field shape measurement is generally performed using radiographic film or external portal imaging devices (EPID) with appropriate markers such as BBs to mark the field line edges. The medical physicist compares the film or EPID readout with that of light field.[2,82-84] Both the film and EPID measurement can achieve submillimeter precision. However, due to penumbra effects, the boundaries of the radiation are not always as clear as might be desired,[11] and thus there remains some subjectivity in the determination of the beam boundaries from either film, EPID or light field. This is compounded when beams are very small or large, or when complex shapes are used. Physicists' time and equipment required are the limiting factors associated with workload in Linac QA.[85] The calibration procedures are required to interpret dose reading from the film and EPID measurement, which also increase the work load of the physicists to do the 2D dose measurements using film and EPID.[86,87] There is an opportunity to do beam shape

verification completely electronically if a secondary measurement of shape could be acquired at the SSD of the patient, and in this study Cherenkov imaging was evaluated for this application.

Ionization chambers and diodes are widely used in Linac QA to measure point dose because of their accuracy and fast readout.[86,88,89] Many recent studies and products have been developed using 2D or 3D arrays of ionization chambers or diodes to extract dose distributions in phantoms to verify the treatment plans in external beam radiation therapy (EBRT). Such applications have also been used to extract lateral dose profiles to do light field congruence test in Linac QA, or even in more complex QA such as multi-leaf collimator (MLC) shape verification, or even IMRT plan verifications.[8,90-92] However, due to the large size of individual ionization chambers or diodes, these tools do not permit high resolution dose measurement.[88,93] The common resolution of the ionization chamber is 5 mm, [91] which is much larger than the resolution of radiographic film, which can approaches 0.1 mm. [85] In addition to the problem of lower resolution, the observed error can reach up to 10% for ionization chambers measurements due to their large size.[90] In addition, ionization chamber and diodes cannot measure the light field at the same time of dose measurement, requiring the physicists to do light field measurements manually, which should agree with the beam size to a tolerance of 2mm or 1%.[2,82,83] These latter measures are often taken by physicists using graph paper or other methods of physical alignment.

Cherenkov emission was recently adapted as a tool to analyze dose distribution on the surface of phantoms and patients' skin[33,94]. Experiments have shown that the Cherenkov signal intensity is linearly correlated with dose delivered to the surface of a

plastic phantom.[95] Based on this relationship of linearity, human studies have been done to evaluate the surface dose distribution on skin in breast treatment[40] and total body electron treatment,[41] as a verification of the treatment plan. This clinical work is still ongoing to determine the potential value, but imaging of the 3D dose distributions in a water tank, appears to agree very well with dose distributions simulated by the treatment planning system (TPS) and verified by film.[29,36,96] Recent work has also extended this to verify beam shapes of complex radiation treatment plans, such as IMRT, and confirm the match lines between adjacent beams.[97] These Cherenkov signals can be directly read using camera systems with a time-gated intensified camera.[35,97] These cameras can have high pixel density, and so it is feasible to acquire high resolution dose delivery shapes of the beams incident upon solid phantoms, or even patients, from Cherenkov imaging.[28,94] The Cherenkov camera can also acquire Linac light field images/videos, which makes it possible to compare light field shapes to dose delivery shapes. Since both Cherenkov and light field information are saved as digital images/videos, image processing techniques could make the analysis of light field and radiation field coincidence seamlessly automated, and more objective than the physicist visually verifying light field projection images on grid paper as compared to the film images.

2.2 Method and Materials

2.2.1 Measurement and Experiment Setup

Measurements were conducted in two treatment rooms with two different medical linear accelerators (Varian 2100CD, Varian Medical Systems, Palo Alto, CA). Beams of 6MV photons were used to irradiate the phantom and film. The dose rate of the beam was 600 Monitor Units/minute (MU/min) delivered at a SSD of 100cm. For each measurement

imaging Cherenkov emission, a total of 200 MUs were delivered and to maximize the signal on each film, a total of 1000 MUs were used. During the measurement, the gantry and the collimator of each Linac were fixed at 180° angle in Varian coordinate. Square beam shapes were using the XY jaws, setting the beam sizes of 5×5, 10×10, 15×15 and 20×20 cm^2.

Two different camera systems were used to assess performance, having been produced by DoseOptics LLC as early prototypes of time-gated intensified CMOS cameras (C-Dose™, DoseOptics, LLC, Lebanon, NH). Both image acquisition systems were coupled with fixed focal length lens (50mm f/1.8, Nikon Inc., Belmont, CA), and calibration of each included flat field response correction using a custom LED panel for continuous bright non-saturating signal (Thorlabs, Newton, NJ) and dark field correction with the lens cap on the camera to simulate the camera sensor closed to light.

A square beige Acrylonitrile Butadiene Styrene (ABS) plastic slab (40cm×40cm×1cm), served as the phantom for imaging both light field and Cherenkov emissions, as well as placement of the film. This was placed in the horizontal plane with SSD=100cm, and centered on the beam axis. The edges of the slab were oriented in parallel with the axes created from the alignment laser, in both X and Y directions. The focal plane of each image acquisition system pointed at the isocenter of the inac, and each were positioned 3 meters above the ground, on the right side of the treatment couch from the patient's perspective, to make sure the ABS slab was in the field of view of the image acquisition system. During the measurement, the room lights in each treatment room were turned off to minimize the background noise.

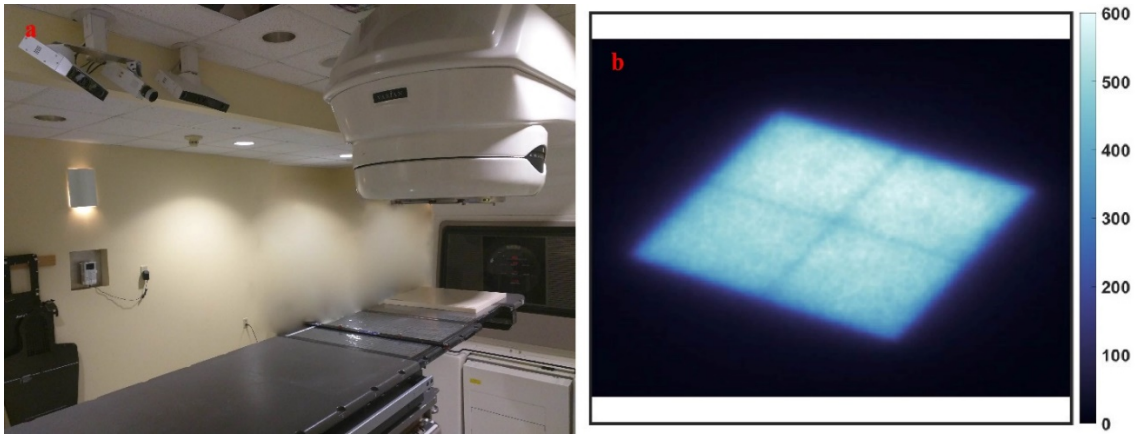


Figure 2.1: The treatment room is shown with the ABS test plate placed at isocenter on the couch and camera mounted to the ceiling, shown in (a). A typical image of the Linac light field on the board is shown from a 10x10 cm beam in (b), as captured by the Cherenkov camera in image sensor reading.

2.2.2 Position calibration for Cherenkov imaging

Before the measurements of Cherenkov emission were done, a transformation matrix was calculated to correct for the skewed perspective angle of the camera, relative to the plane of the imaged phantom. A checkerboard pattern was placed on the surface of ABS slab and used to construct this projective transformation matrix. The pattern was composed of 6x8 squares with consecutive black and white grids and the side of each grid had a length of 2.3 cm square. Images of the pattern were post-processed using OpenCV checkerboard function, based on the Harris corner detector algorithm,[98] extracting pixel locations of the corners in the image as discrete points to map to.

The point locations *i* are defined as $p_i = [x_i, y_i, 1]^T$, in which x_i and y_i are the X and Y components and the 1 as the third augmented term. The corresponding physical location of *i* was defined as $q_i = [X_i, Y_i, 1]^T$. The pixel and physical locations of all

corners can be summarized as matrices $p = [p_1, p_2, \dots, p_n]$ and $Q = [q_1, q_2, \dots, q_n]$, where n is total number of corners in the pattern. Then the relationship between the physical locations and image pixel locations can be expressed as $Q = AP$, in which A is 3×3 perspective projection matrix. It is feasible to calculate matrix A from the matrices P and Q with enough measurements of the corner locations. This was implemented in the image processing toolbox of MATLAB (v 9.2.0 R2017a, The Mathworks, Natick, MA), and allowed the optical intensity of Cherenkov signal and light field shown in the skewed camera perspective to be mapped into the physical locations on the plane of the slab, as illustrated in Figure 2.2.

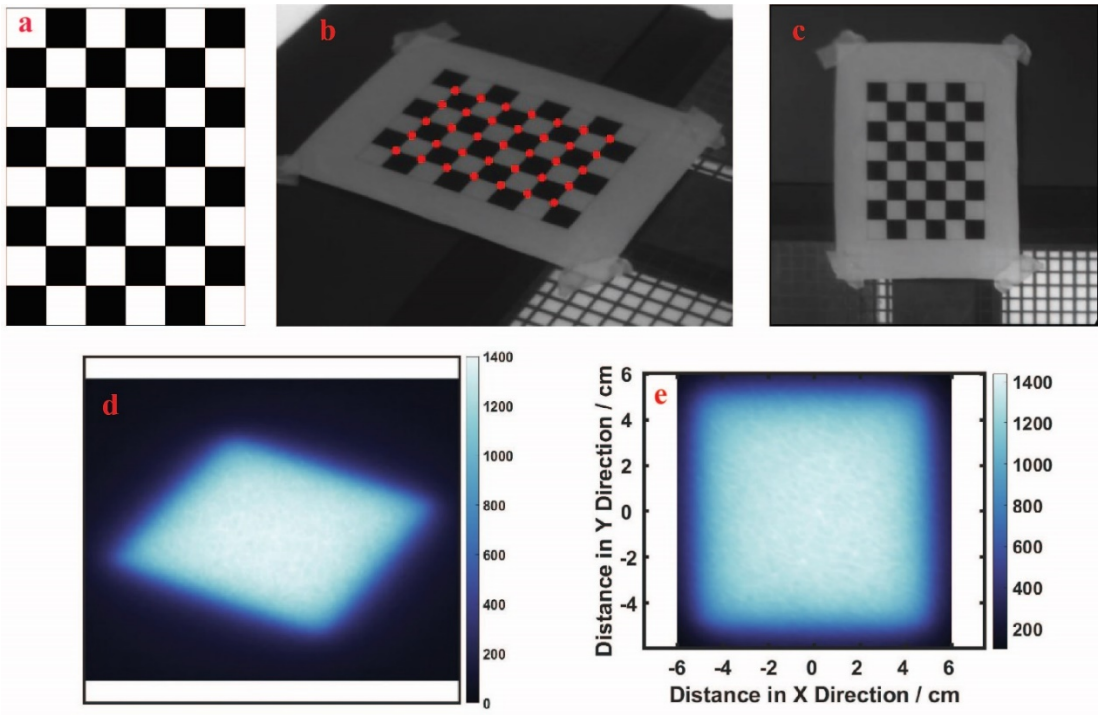


Figure 2.2: In (a) the square patterned board used for calibration is illustrated, with (b) a calibration image taken from camera showing the corners detected (red dots) using the OpenCV library, and remapped (c) to be projected as the undistorted board image. In (d) a Cherenkov image of a 10x10 cm^2 beam is shown, and (e) the remapped Cherenkov image is shown as a square.

2.2.3 Cherenkov and light field imaging

During beam delivery, images were acquired at a frame rate of 23 frame per second for approximately 460 video frames, in 20 seconds. At the end of each measurement, background frames were extracted for an additional 20 seconds, to allow for background subtraction. The raw video frames were first post-processed with three image processing procedures: frame averaging, flat-field/dark-field correction[99] and background subtraction. The flat dark field video frames are taken with a lens cap attached to the camera system, while the flat field frames are taken with a flat field correction panel attached to the lens. The light intensity is distributed uniformly in the flat field correction panel. In the frame averaging stage, a single image labeled as I_l, I_c, I_b, I_d and I_f, is created by taking a moving window temporal average of 100 frames in the middle of each piece of video, for each of light field, Cherenkov, background, dark-field and flat-field. Then the images of light field, Cherenkov and background were processed with pixel-wise flat-field/dark-field correction, using equations listed in I, II, and III, below, where the front factor is simply to normalize the intensity to the maximum of the flat field measurement, and the latter factor removes the effect of dark field pixel variation.

$$I_c'(i,j) = \frac{max(I_f)}{\left(I_f(i,j) - I_d(i,j)\right)}\left(I_c(i,j) - I_d(i,j)\right) \quad \text{(I)}$$

$$I_l'(i,j) = \frac{max(I_f)}{\left(I_f(i,j) - I_d(i,j)\right)}\left(I_l(i,j) - I_d(i,j)\right) \quad \text{(II)}$$

$$I_b'(i,j) = \frac{max(I_f)}{\left(I_f(i,j) - I_d(i,j)\right)}\left(I_b(i,j) - I_d(i,j)\right) \quad \text{(III)}$$

where *i, j* are the pixel indices in the X and Y directions of one image, and $I(i,j)$ is the corresponding pixel reading and $max(I)$ is the maximum pixel reading of one whole image. $I_c^{'}$, $I_l^{'}$, and $I_b^{'}$ are the Cherenkov, light field and background images processed by correction. In the last step of image processing, the background image is subtracted from the Cherenkov and light field images using equation IV and V, to remove the background noise. The background image is taken when there is no radiation field, to include the background light signals.

$$I_c^{''}(i,j) = I_c^{'}(i,j) - I_b^{'}(i,j) \quad \text{(IV)}$$

$$I_l^{''}(i,j) = I_l^{'}(i,j) - I_b^{'}(i,j) \quad \text{(V)}$$

The resulting 2D distributions of Cherenkov and light field were reconstructed through equation VI to VIII, with the perspective projection matrix *A*, derived from the spatial transformation calibration. The line profiles could then be extracted in both X and Y directions for Cherenkov and light field.

$$D_c(x,y) = I_c^{''}(i,j) \quad \text{(VI)}$$

$$D_l(x,y) = I_l^{''}(i,j) \quad \text{(VII)}$$

$$[x,y,1]^T = A \cdot [i,j,1]^T \quad \text{(VIII)}$$

The value of *x* and *y* define the physical location, as shown in Figure 2.2(e), and $D(x,y)$ is the corresponding Cherenkov or light field distribution at that location, as shown in Figure 2.2(d). The profile of the squared beam width is measured in the X and Y directions of jaws.

The boundaries from each image are not perfectly clear in light field, the Cherenkov emission image, or the film, due to the noise of the camera's image sensor and beam penumbra. Thus, two standards were examined to determine the boundaries of

Cherenkov signal or light field images: 1) the full width at half maximum (FWHM) and 2) maximum slope profile estimate. The standard of FWHM is estimated with average values of the maximum and minimum signal intensity, along the profile line, but the definition of the width necessarily depends upon the definition of the maximum and minimum values, as illustrated in Figure 2.3(a). Thus, when peak and background intensities are noisy, the pixel readings need to be averaged to estimate the peak value of the curve. Also, when the light imaged is diffused by the test object, it may be that the lateral diffusion alters the apparent penumbra of the imaged beam. The second standard definition for boundary edges is the location of slope maxima, as defined by the extrema points of the first derivative of the profile line. These locations are illustrated in Figure 2.3(b), and for a typical profile plot are noticeably a few millimeters narrower than the FWHM definition. Both were tested here.

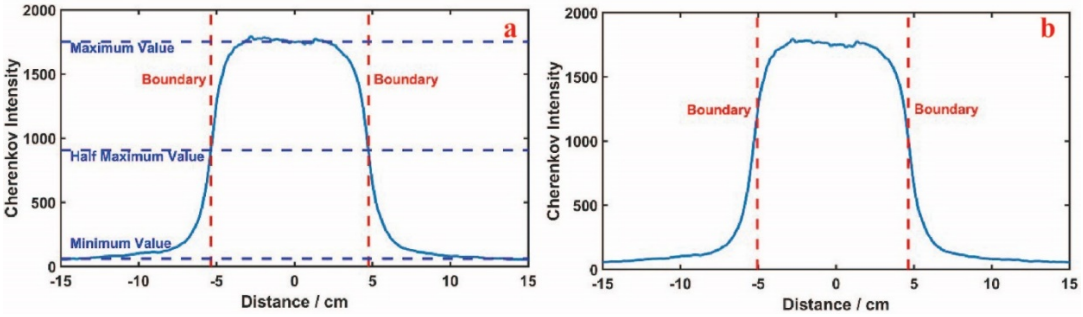


Figure 2.3: The sequence of image analysis is shown in (a) with a line profile from the Cherenkov image showing the standard of Full Width Half Maximum Illustration (c): Boundaries extracted using the standard of maximum derivatives

2.2.4 Beam measurement by film analysis

The film measurement served as the gold standard for measurement to evaluate the performance of light field and Cherenkov measurements. In this paper, the GaF films (Ashland Advanced Materials, Bridgewater, NJ) were used to measure the 2D dose

distribution of the surface at the phantom surface. The films were then digitized by a film scanner (Epson, Long Beach, CA) , and the intensity of the red channel was used to calculate dose, due to its higher sensitivity to dose change than the green or blue channels.[85]

In order to calibrate the film intensity to dose units, multiple 2"×2" squared patches were used and placed in solid water at d_{max} = 1.6cm for a 6MV photon beam, with full backscatter, and absorbed doses between 0 and 500cGy, were delivered. A third-order polynomial equation was used to fit the correlation curve of dose to film reading.[85]

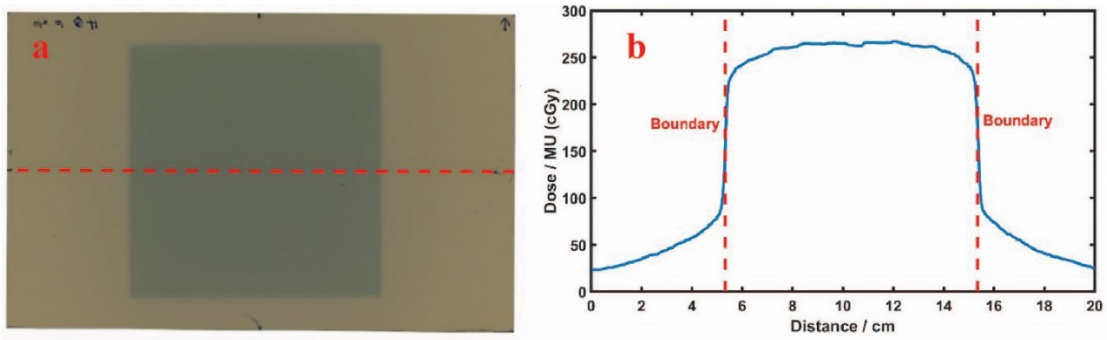


Figure 2.4: Beam width measurement using GaF films (a): Film scan for 10cmx10cm beam and 5cmx5cm beam, with line profile region in 10cmx10cm beam (b): Dose profile along the line, with red lines representing the boundaries extracted through maximum derivative method

In the 2D dose measurement, solid water slabs of 60cm thickness were placed on the treatment couch to provide backscattering, and one piece of film was taped on the top surface at SSD = 100cm, with the central axes aligned to the Linac isocenter. About 300 to 400 cGy was delivered in the central region. Each piece of film was read directly after the measurement and converted into a dose distribution according to the calibration equation. The boundary standards used in Cherenkov and light field analysis were also

applied to dose distributions, to extract the widths of beams in X and Y directions. The width of dose profile was defined as beam width, as shown in the bar graphs.

2.2.5 Inter-observer analysis of the field size estimation

An inter-observer analysis was conducted to measure how accurate any single visual estimate of the field edge could be achieved by spatial coincidence testing between the film and the observed light field, as is commonly done in quality assessment procedures. In this analysis, each of two medical physicists participated to measure the size of the light field for ten repeated measurements of a beam, with side lengths varying randomly from 19.5cm to 20.5cm at 0.1 increments. The sequence of beam sizes was generated randomly by computer, and the display panel of the Linac was obscured to ensure the physicists did not know the beam size planned. During the measurement, they used grid paper and a ruler to measure the beam size, and recorded the results of each beam side distance. These distances were then compared to the known input sizes.

A standard light field to beam edge coincidence test was performed with a piece of grid paper with radio-opaque markers on the edges for a 20cm×20cm square, and a piece of phosphor film. Before the light field congruence test, the phosphor film was placed underneath the grid paper and the grid paper was aligned with the light field of a 20cm×20cm square beam, so that the edges of light field bisected the ball bearing markers, as illustrated in Figure 2.5(a). As in the film measurement, the couch was set to SSD = 100cm, and the grid paper was placed at the isocenter. In the congruence test, the phosphor film was irradiated with 2MU using a 6MV photon beam and a dose rate at 600MU/Min. After the film was exposed and scanned into the computer, the physicists checked the location of the markers projected on the phosphor film, as shown in Figure

2.5(b). The locations of the beads are used to determine success in the beam edge position localization. In this sensitivity analysis, there were eleven squared beams used, with the size of the beam ranging from 19.5cm to 20.5cm, with 0.1cm increment. Images for analysis were then chosen at random for measurement.

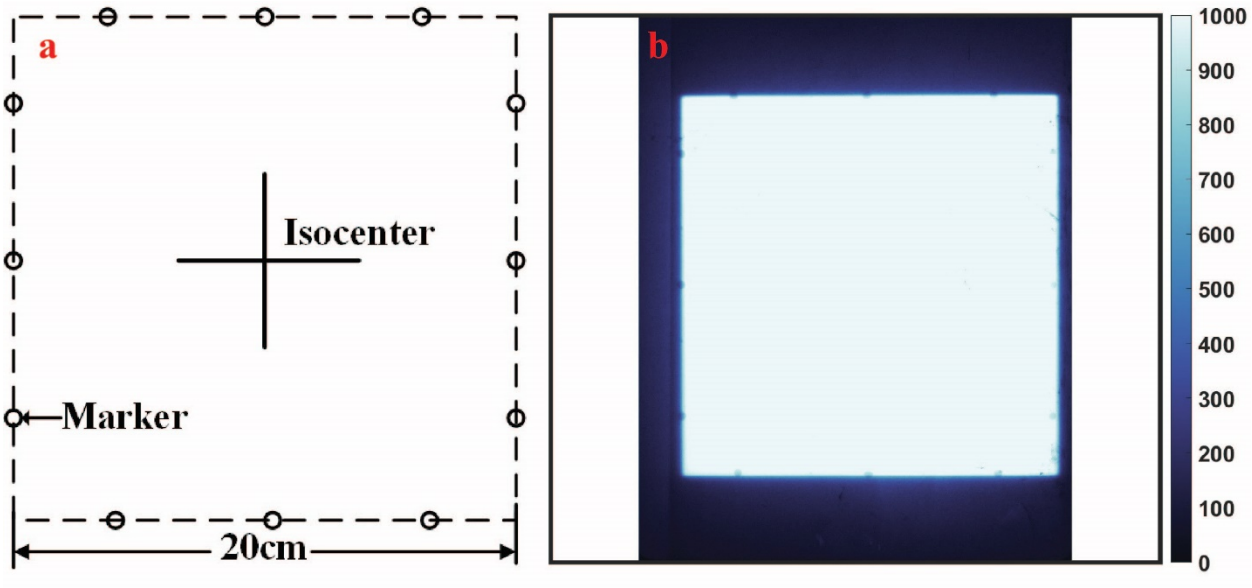


Figure 2.5: A common method used in light field quality assessment is illustrated with (a) a pattern used for a 20x20cm^2 square beam. In (b) one example processed phosphor film is shown after exposure, with the location of the circular bead markers visible on each of the 4 edges.

2.1. Results

2.3.1 Image Transformation Verification

The matrix used for image transformation are verified using checkerboard patterns and black squared patterns. The checkerboard patterns consist of 22×16 black or white cells. The size of each cell is 1×1cm. In the verification test, the images of the checkerboard patterns are acquired by the Cherenkov image acquisition system. The pixel locations of the grid corners (21×15) are extracted and their physical locations are translated using the

transformation matrix derived before. The physical locations derived for the cell corners are compared with their real locations measured by hand. The errors of the derived physically locations are within 1mm on average, with less than 1mm standard deviations.

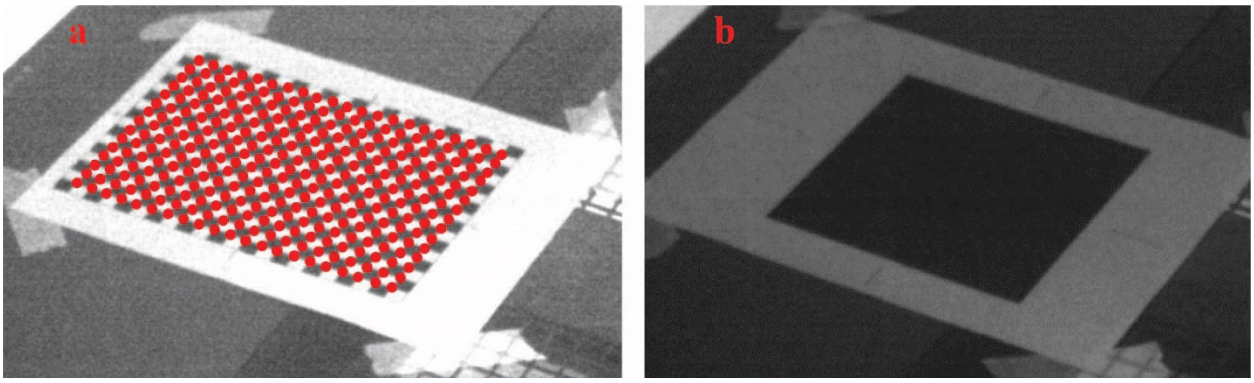


Figure 2.6: Paper image patterns to verify the errors of image transformation: (a): Checkerboard patterns with 1cm×1cm cells, (b): 15cm×15cm black square pattern to simulate radiation and light field with the same size.

The sizes of the black squared patterns range from 5cm×5cm to 20cm×20cm, to simulate the beam shape of radiation and light fields. The widths of the black squared patterns are measured by the same methods as that for the radiation and light fields. The measurement made by the Cherenkov image acquisition system has the error within 1mm for all sizes of the squared pattern. In both measurement of checkerboard and black squared patterns, the lens correction has also been tested and its impact is minor for the measurement error using the image transformation matrix.

2.3.2 Square beam width estimation

The results of measurements are summarized numerically in Table 1, listing the width from light field, Cherenkov emission analysis, and film measurement. These values show the better width estimation of maximum slope on each edge, which matched the true beam sizes better than FWHM. Mean distance error were about 1mm to 2mm for the

max slope method for light field and Cherenkov emission analysis, respectively. This was averaged over all beam size ranges from 5cm×5cm to 20cm×20cm, and the width was classified into X and Y jaw directions. When comparing to the GaF film measurements, the light field measurement is generally within QA guidance tolerance (the greater value of 2mm or 1%) using either definition. However, some measures of Cherenkov width with FWHM estimation were not within tolerance using the standard FWHM definition, suggesting that the optical scattering of the edge was likely widening the estimated edge more than would be desired. As such, the width measurement performed better using the definition of the edges by their extrema difference, as defined as the maximum slope of each of the lines. There was not a significant difference observed between directions of measurement nor between the two different rooms.

Location	Direction	Nominal Beam Size/cm	Film Width /cm	Light Field Width /cm	Cherenkov Width/ cm	Light Field Error / mm	Cherenkov Error/mm
Linac 1	X	5	5.0	4.8	5.3	2	3
		10	10.0	9.7	9.9	3	1
		15	15.0	14.9	15.1	1	1
		20	20.0	19.9	19.9	1	1
	Y	5	5.1	4.9	5.0	2	1
		10	10.0	9.8	10.4	2	4
		15	15.0	15.0	15.0	0	0
		20	20.0	19.8	20.0	2	0
Mean Error						**1.6mm**	**1.4mm**
Linac 2	X	5	4.9	4.9	5.1	0	2
		10	9.9	9.9	9.7	0	2
		15	14.9	14.9	15.1	0	2
		20	19.9	19.9	20.0	0	1
	Y	5	4.9	4.9	5.1	0	2
		10	9.9	9.9	10.2	0	3
		15	14.9	15.0	15.1	1	2
		20	19.9	20.0	20.0	1	1
Mean of Error						**0.3mm**	**1.9mm**

Table 2.1: Summary of results of beam width measurements as measured from two independent Linac tests

2.3.3 Sensitivity Test of Light Field measurement

In the light field width matching, the inter-observer study used light field alignment using grid paper, and then x-ray exposure to bead markers of a fixed 20x20 beam, and exposure to phosphor film, to check if the radiation field on the film matched with the light field as seen at the SSD location. As shown in Figure. 2.5b, the markers on the edge of the squared pattern leave shaded areas on the phosphor film underneath the paper. However, due to the penumbra effect of radiation fields, the dose distribution changes gradually near the boundaries. In the observer test with 11 repeated test patterns, with human estimation of the light field, the inter-observed error of two physicists had a total range of 0 to 2.0mm, with a 1.0mm average inter-observer error. These error estimates from light field are within the tolerance given by the guidelines of Linac QA. [2]

2.3.4 Measurement of Cherenkov field of complex treatment plan

Measurement was done to analyze the beam shapes of complex treatment plan. In this experiment, the Cherenkov field of an IMRT QA plan with both moving and complex shapes were acquired using the Cherenkov imaging system. The shapes of the fields were transformed into the isoplane, and compared with the contours of MLC configurations, extracted from the dynalog files of the treatment. The shapes of Cherenkov fields after transformation matches the patterns of MLC projected on the isoplane.

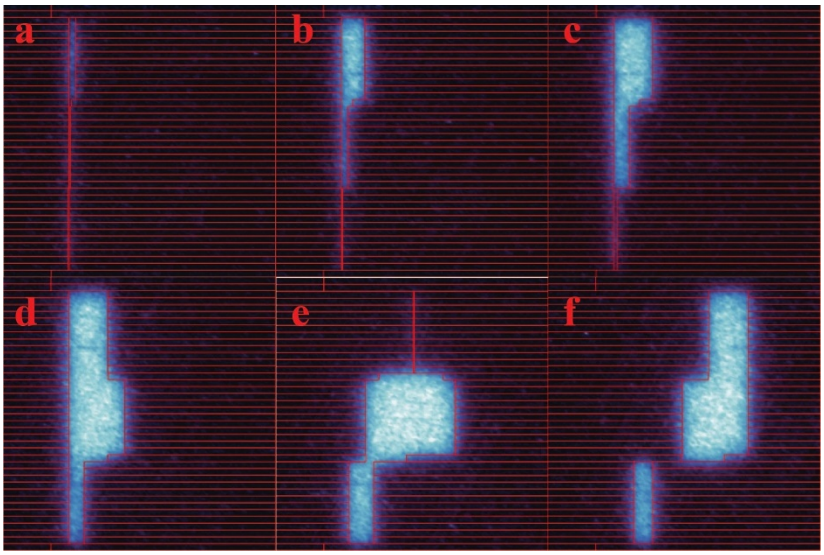


Figure 2.7: Cherenkov images of IMRT QA plan on the isoplane, compared with MLC shapes projection, with leaves marked by red rectangles

2.3.5 Isocenter measurement by rotating collimators

The isocenter measurement, commonly fulfilled by the "star-shot" analysis, can also be fulfilled by the Cherenkov image analysis. In this experiment, the 30cm×0.5cm squared beams radiated the plastic phantom on the isoplane, with collimator angles of 30°, 90°, 150°, 180°, 240° and 300°, to form a star pattern in the isocenter. The 2D dose distribution is also measured by film for the beams with these collimator angles. For each collimator angle, 400 MUs are delivered from the Linac machine to get high contrast of film reading.

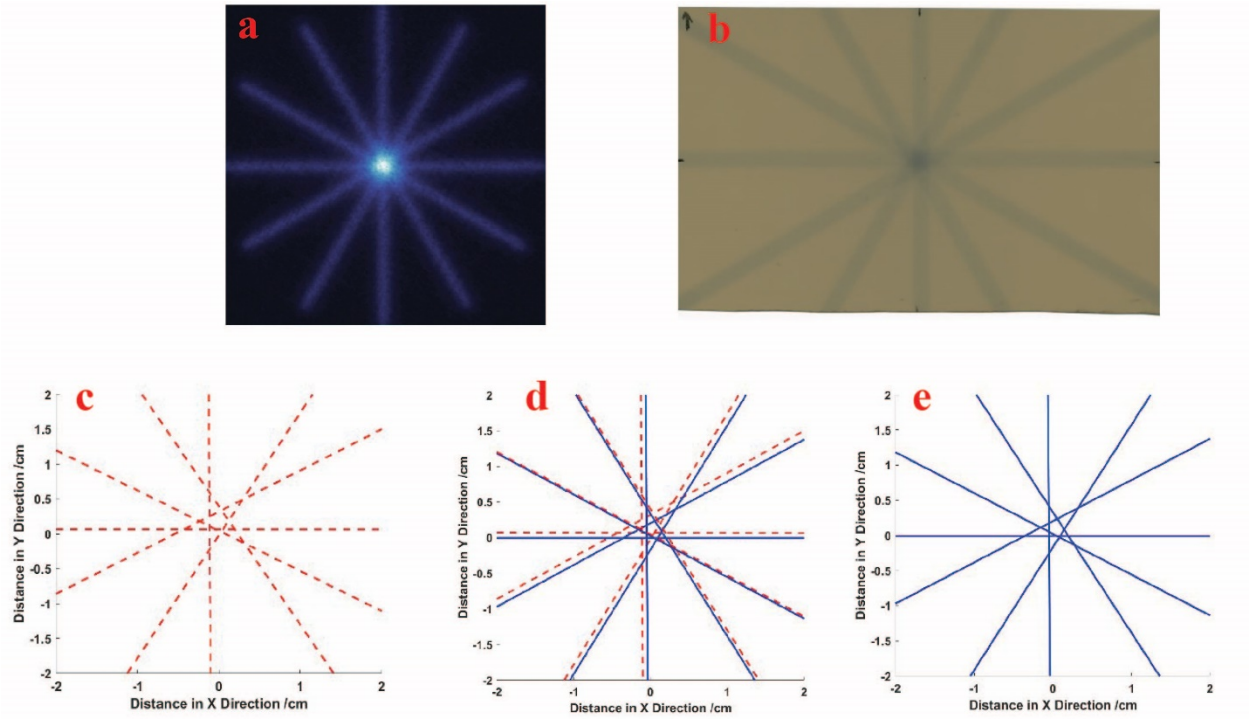


Figure 2.8: Star shot analysis using measurement of Cherenkov and film, (a): the Cherenkov field measurement; and (b): the film measurement of 30cm×0.5cm radiation beams for different collimator angles, the center axis of beams extracted in the Cherenkov profile (c) and film profile (e),with the comparison shown in (d).

The central axes are extracted using the methods in "star-shot" QA analysis for both Cherenkov and film measurements[100,101]. The locations of the central axes and star shapes formed by the axes are similar for Cherenkov and film profiles. For Cherenkov profile, the radius of the star is about 2.1mm, while it is 1.9mm for the film profile. Cherenkov measurement achieves similar performance as film measurement in the "star-shot" QA of isocenter for varied collimator angle. Similar procedures of Cherenkov image analysis can be done for varied bench angles in the "star-shot" analysis. Thus, Cherenkov image analysis is a potential tool for Linac QA of isocenter.

2.4. Discussion

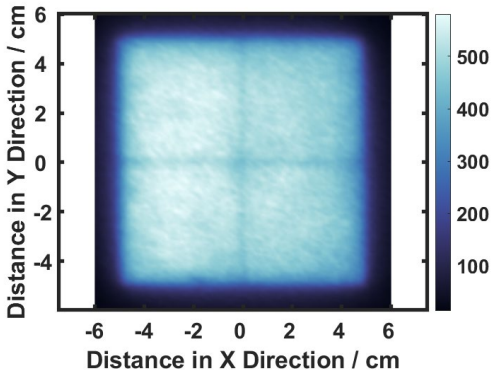
The results here demonstrate that there is good spatial agreement between the measurements of light field, Cherenkov emission and film using the two boundary standards of half maximum and extrema difference. Based upon physical principles, ideally, we expect there to be a perfect match between the film measurement of the beam and the Cherenkov images, because both signals are produced by the beam dose and it is known that both are monotonically related to dose.[95] As such, the idea of using Cherenkov imaging as a replacement for film makes intuitive sense, and the primary value is in the ease of use of electronic imaging (Cherenkov) versus physical imaging using film. Other than the submillimeter error in the process of the image transformation, the likely areas for spatial disagreement between them would be in areas of penumbra or scattering, attributed to the scattering of Cherenkov light signal inside the plastic phantom. From this aspect though, the non-linearity of film is well known, and the inherent value of the linearity of Cherenkov emission with dose should be superior.[95] There are known variations in sensitivity to beam energy and beam type of electrons or photons in both, but each of these factors can be calibrated for, if necessary. However, the value of this study has largely been in the simple confirmation of the ability to spatially map the Linac beam as observed through imaging with film, Cherenkov imaging and the light field.

The observer study of using the light field to estimate the beam size and comparison of testing for a fixed 20cmx20cm x-ray beam helps illustrate that fact that most alignment is done with a +/-1mm tolerance on average with a 0-2mm range of error. It is well known that there is this level of subjectivity when physicists are involved in

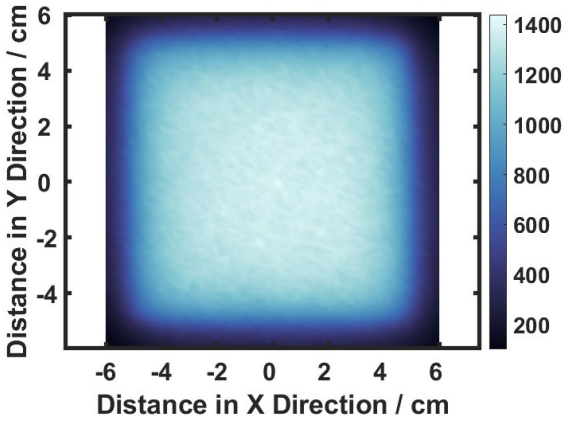
deciding the boundary of the radiation field manually. So as a result, there is a preference to complete light field to beam congruence validation electronically wherever possible. The value of Cherenkov imaging is that the images are acquired at the location of the surface, at SSD = 100cm, similar to film. Also, the value of all electronic capture and electronic image processing evaluation removes human errors from the post-processing evaluation of this data, which can be especially accurate and potentially time saving for routine quality assessments. In principle, imaging can be fully automated, other than placement of the board upon which the irradiation is completed.

One issue to consider is the observed value of the Linac light field and Cherenkov emission relative to the film. They are very similar, as shown in FIG. 9, and the size of light field and Cherenkov emission are accurate representations of the beam shape derived from the GaF film. However, the boundaries of optical images (a) and (b) on an ABS phantom, and (c) on a solid water phantom are more blurred than those of the radiation field as reported by the film (d). Part of the blurring effect likely comes from the scattering of the beige plastic board, which has also been observed in many other plastic materials. Some plastic material, such as solid water phantom (d), can have sharper penumbra than the ABS phantom, due to the optical properties of different plastic materials. A physicist can determine the edge of the light field by visual inspection, while the light field derived from the camera system has more blurred boundaries and care must be taken to estimate the edges accurately. Therefore, modifications of the board could potentially reduce the penumbra size of both light field and Cherenkov emission. In the future, a less translucent plastic sheet could be used in the measurement to improve the light signal while minimizing the optical scatter diffusion at the edges. The ABS sheet

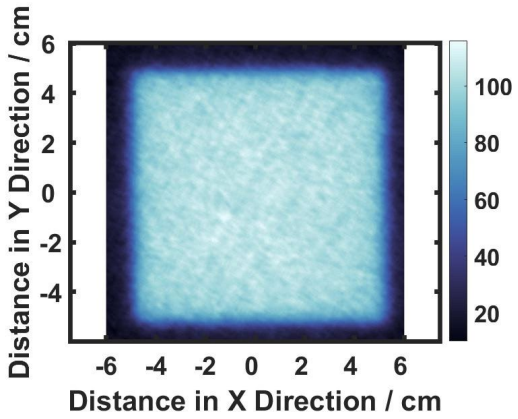
was chosen as a general-purpose board for testing, due to its higher Cherenkov response, and it is still likely that superior materials, such as a solid water phantom with brighter colors, may be available to maximize light field edge imaging. GaF film was used as a gold standard comparator here because it is maximally sensitive to higher energy photons, as compared to phosphor film which is also sensitive to lower energy photons. Thus, the penumbra effect of radiation field in the phosphor film in Figure 2.5b is more obvious than that seen in the radiation field measured by GaF film, as in Figure 2.9d. To reduce this effect, the phosphor film is always used with film cassette to filter out lower energy photons.



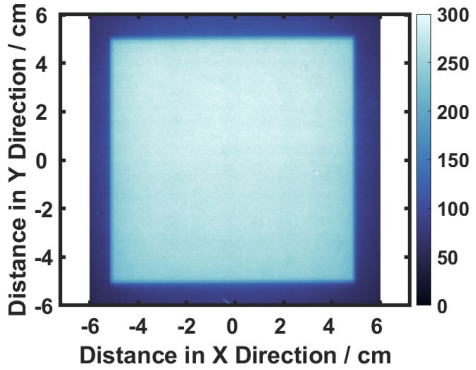
(a)



(b)



(c)



(d)

Figure 2.9: Comparison between the 2D profiles of light field, Cherenkov, and dose of 10x10cm^2, 6MV, photon beam. (a): 2D profile of light field, (b): 2D profile of Cherenkov emission in an ABS board, (c): 2D profile of Cherenkov emission on a solid water phantom, (d): 2D profile of dose from GaF film measurement.

Using appropriate definition of the boundaries for the Cherenkov emission and light field, such as the extrema of derivative, shown here, the sizes of the light field and Cherenkov emission spatially matched that of the light field, albeit with edges which were more diffuse than observed with the film, but they were substantially similar to the

observed edges of the light field. Sizes estimates derived from FWHM showed less accuracy with some variation up to 3 to 5mm, for the measurement of light field and Cherenkov emission image, relative to the film. However, measurement of both light field and Cherenkov emission image achieved much less error when using the maximum slope estimates of edges.

The beam shape analysis protocol using Cherenkov image acquisition system can be extended to other QA procedures using beam shapes, such as MLC verification and isocenter QA, as shown by the star shot analysis. The results section shows the match between Cherenkov and radiation fields for complex shapes, although in this preliminary work the match to the MLCs was just shown for qualitative reference. Future work on using Cherenkov imaging for IGRT is possible, although this needs to be examined in future studies. The QA process using a Cherenkov image acquisition system has potential advantage in terms of time and workload as compared with film and/or manual measurement of the light field. Even though the Cherenkov camera needs to be installed in the treatment room, once it is installed, the system, the acquisition and processing can be fully automated.

2.5. Conclusion

We have demonstrated the use of electronic imaging of Cherenkov emission to test the ability of using Cherenkov imaging to verify radiation field beam sizes from medical linear accelerators. This system can take images/video frames of Cherenkov emission and light field of square beams but has potential for arbitrary shaped beams and more complex tests such as star shots. The measurements of Cherenkov emission show good congruence with film measurement and light field, with some caution needed in the

appropriate standard of a boundary definition. Without calibration of the image acquisition system, the discrepancy between light field or film and the Cherenkov image is about 1-2mm or 1% on average, which is within the tolerance given by the QA routinely used for medical linear accelerators.[2] Therefore, it is feasible to use this image acquisition tool to run QA procedures with appropriate refinement of the software tool used.

Chapter 3: Body Surface Dose Analysis of TSET

The contents in this chapter is adapted from the following publication.

Miao T, Petroccia H, Xie Y, et al. Computer animation body surface analysis of total skin electron radiation therapy dose homogeneity via Cherenkov imaging. *J Med.Imaging.* 2020;7(3):034002.

3.1 Introduction

Mycosis fungoides is the most common form of cutaneous T-cell lymphoma,[16] and some patients have superficial lesions throughout the skin of the body, which can require large radiation areas or even the whole skin surface.[102] Therefore, total body dose uniformity can be the goal for radiation therapy of this disease to achieve partial remission and low toxicity.[18,103,104] In most TSET setups, the Linac gantry points towards the standing patient and a broad electron beam treatment is delivered to cover the patient's entire skin area.[105,106] There are largely three techniques to achieve total skin coverage, with one having the patient rotate slowly on a stand while standing with arms and legs spread,[107] another having the beam scan across the patient in prone and supine positions near the floor, and the third being the Stanford technique where the patient stands in 6 different limb and body positions that are designed to expose many body surfaces to the beam sequentially.[103] The vast majority of centers appear to use the latter Stanford technique for historical reasons, although little published data exist documenting this. Validation of the coverage homogeneity is challenging, and while phantom tests can be done,[43] this is labor intensive, and more commonly, dosimeters, such as TLDs, OSLDs, and diodes, are attached at multiple positions on the patient, to directly monitor the dose delivered to specified verification points.[108]

In particular, the curvature of the body and the complexity of trying to get relatively homogeneous dose over the entire skin surface is extremely challenging. There is a goal of +/- 10% agreement in dose over all the skin relative to the dose at midline[19,21], but this is rarely checked with dose measurement over the large area of the patient's body surface because of the logistics and the length of time required for the set up. As such, there is relatively little published and expected in any individual TSET treatment about confirmation of dose homogeneity. Yet, imaging tools and surface capture tools are actually inexpensive and readily available, and to test the homogeneity of coverage, with the use of animation tools to map the dose onto the anatomic locations. The goal of this study was to test the ability to visualize non-contact Cherenkov imaging on an anatomic surface map the 3D patient skin as a surrogate of dose delivery, to determine if computer animation methods could be better utilized to map out the homogeneity of the coverage to the skin.

Cherenkov photons are generated from relativistic electrons traveling through dielectric materials, such as water and human tissue.[28,33] This Cherenkov emission can be imaged from the surface of tissue,[22,38] during X-ray or electron beam in radiotherapy, with the intensified time-gated camera.[95] Experiments and theory have shown that the intensity generated in tissue is generally linear with the dose deposited on the corresponding surfaces, but that there are variations in intensity, largely due to blood vessels, capillary blood volume, and surface reflectivity that can be corrected for by either reflectance or CT number.[43] The Cherenkov imaging was used to derive the 2D dose distribution in TSET treatment.[21] Thus, to a first approximation, the superficial Cherenkov emission distribution is generally representative of the relative dose distribution. Surface images of

Cherenkov emission present a view of dose delivery which provides intuitive understanding of the beam-tissue interactions. This approach to imaging dose delivery has a major potential advantage for large area irradiation such as TSET where total skin coverage needs to be verified to check the anomaly regions, such as overlaps or gaps in field uniformity.

The Stanford technique is one of the three main treatment techniques used in TSET and is the only approved TSET technique in the study sites of Dartmouth Hitchcock Medical Center and the Hospital of University of Pennsylvania. In the Stanford technique for TSET, the patient is placed in 6 distinct anatomic positions throughout treatment. There is no anatomical imaging technique that can capture all 6 limb and body positions for planning in a single or fused 3D image set; therefore, classical 3D CT information is unavailable. It is however, relatively easy to create 3D optical surface scans of the body, since these scanners are now commonly used for daily position verification during the radiation treatment.[109] In this study, 2D Cherenkov image data was projected onto the 3D optical scan surfaces through texture mapping, which is widely used in computer graphics and photogrammetry research. Medical use of these animation methods have occurred in endoscopic imaging, where optical images are textured onto the 3D meshes of the CT images to visualize the internal structures and locate fiducial markers as a surgical aid.[110] In our study, texture mapping of radiation dose as estimated from Cherenkov emission imaging was mapped onto a single 3D body mesh. The study presents the first use of a 3D animated model of the patient during the TSET treatment, which is digitally altered to match the 6 positions of the Stanford technique, such that cumulative dose image can be mapped onto the surface.

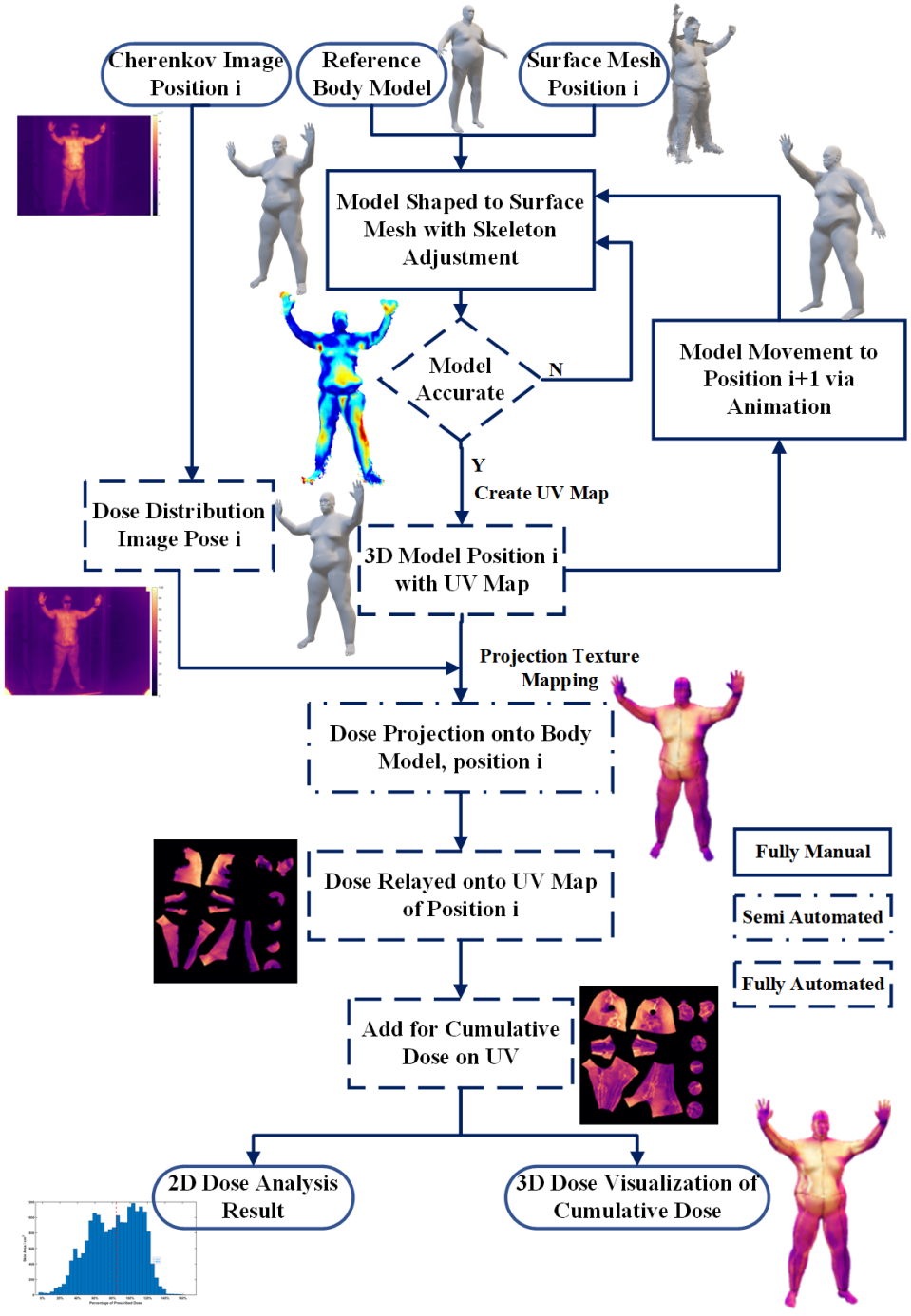


Figure 3.1: The analysis workflow of mesh creation and Cherenkov mapping onto the different distributions, from TSET images, using computer animation techniques.

3.2 Material and Method

The overall concept of the study is illustrated graphically in the flow chart of Figure 3.1. The 2D Cherenkov images and partial 3D body contour meshes of the patient were acquired during a clinical TSET treatment using the Stanford techniques.[21] The Cherenkov images were converted into the 2D dose distribution images, with the point dose measurement in the patient body and flat field correction.[21] The partial 3D body contour meshes for the anterior-posterior (AP) and posterior-anterior (PA) positions were used to build a 3D model of the complete patient body contour, based on warping a reference human body model to match the measured partial 3D meshes. The patient specific 3D model was evaluated and refined through iterative comparisons to the measured partial 3D body meshes and manual editing of the 3D model to minimize discrepancies. After the 3D body model matched the measured data, a mapping was defined between the 3D surface and a fixed 2D representation of the whole-body surface, called a UV map, where U and V refer to the axis of this 2D texture space. The use of a UV map enables the summation of the 3D Cherenkov data across different patient positions, independent of patient rotation or alternations of limb and body positions. Variants of the 3D body model were then created with adjustments to match each of the remaining positions, such as right anterior oblique (RAO) and left posterior oblique (LPO), etc. With 3D models now available for each patient position, the 2D dose distribution image data for each position were projected onto the 3D models and then cast onto UV maps. The UV maps for each position were then summed up to generate a cumulative dose distribution for the entire body for the complete TSET treatment. This cumulative dose distribution could then be visualized on the 3D body model of the

patient as a delivery verification tool and used for 3D Cherenkov emission distribution analysis.

3.2.1 Human Imaging & Treatments

The total skin electron therapy treatments (TSET) were conducted in the linear accelerator (Linac) room of Perelman Center of Advanced Medicine in University of Pennsylvania. All imaging was approved by the Institutional Review Board (IRB) and all procedures followed this approved protocol. The experimental setup included the patient in a wooden stand, with handles for patient position support. A clear plastic spoiler was positioned in front of the patient to provide electron scattering and beam energy degradation of the electron beam from the Linac (Varian Truebeam, Palo Alto, CA). The patient was 5 meters away from the Linac nominal electron source, which pointed towards the patient with two specific angles. Laser line positioning was used on the stand to make sure it was in the same position for different patients and each daily treatment fraction. Before TSET, certain parts of patient's skin, such as the nails or eyes, were shielded with lead, to reduce the radiation dose in these regions where prescribed by the physician.

The Stanford technique was used for treatment. The whole TSET treatment was divided into 2 cycles, with 3 of the 6 patient positions treated each cycle. Each patient position was irradiated using an upper field and lower field, in which the gantry angles were set at 106° and 74°. On cycle A, the patient stood in AP, LPO and RPO position, while on cycle B, the patient stood in PA, LAO and RAO positions. Before the treatment, multiple OSLDs or diodes, were attached to the patient to monitor the dose delivery at specific point locations across the patient's skin. Dose measurements were acquired for the first

fraction of cycle A and B. The laser alignment is turned off during treatment and the room light was dimmed above the patient to minimize background light reflected from the front of the spoiler during Cherenkov imaging.

3.2.2 Camera Systems

There were two camera system setups in the Linac, one to acquire the 2D Cherenkov frames during the TSET and the second to acquire the 3D skin surface contour of the patient before TSET. The Cherenkov camera (CDose, DoseOptics, Lebanon, NH) was located beside the couch, which served as the reference location to make sure the Cherenkov camera was in the position for different TSET treatment days, and used a 28mm lens (Sigma, Ronkonkoma, NY) for similar field of view as the Linac beam coverage. The 3D structured light surface camera (generation 1, Occipital, San Francisco, CA) was positioned in front of the patient and the distance between the patient and the 3D camera was approximately 2 meters, the optimum distance for a useful 3D scan. Before and in between the irradiation, the therapists helped the patient stand in each of the different positions and the patient held these positions with the support of handles during the treatment. Immediately after each position was set, the 3D camera recorded the skin surface contours. Through classical 2D camera calibration methods, the position of the Cherenkov camera was derived relative to the TSET isocenter, which is located at the patient's standing position and at the same height of the Linac isocenter. The position information of the Cherenkov camera was used in the projection texture mapping stage of 3D image processing. The 3D contour mesh of the patient includes partial contouring of the wooden stand, which served as the position reference for the patient's skin surface contour.

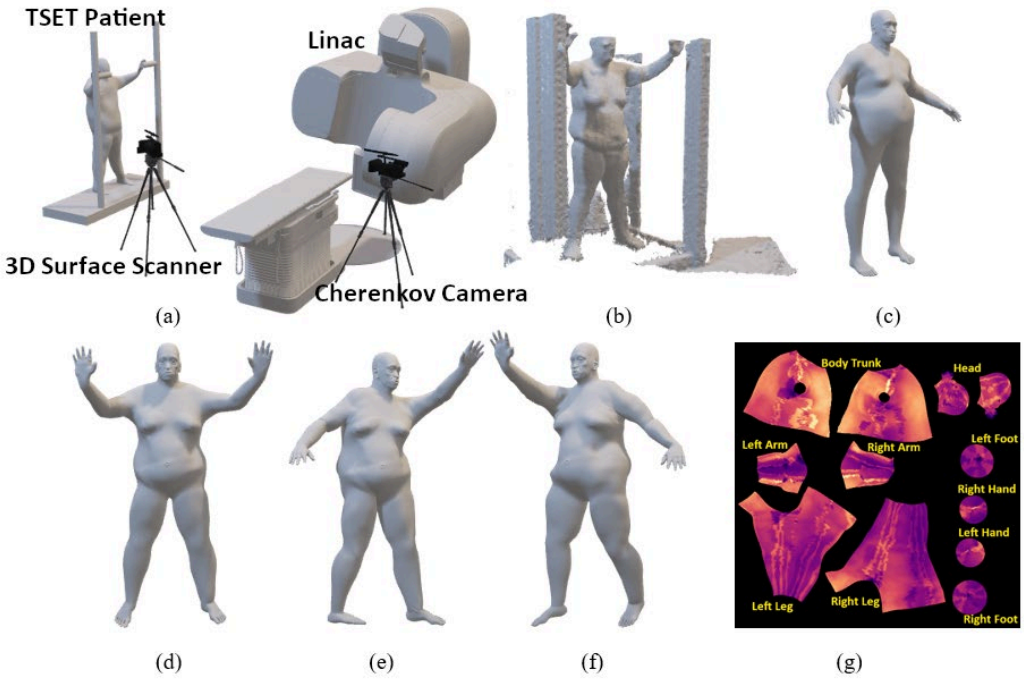


Figure 3.2: (a): The TSET clinical setup with Cherenkov camera behind the treatment table and 3D sensor beside the TSET stand. (b): The 3D raw mesh of the patient in AP position with TSET stand, extracted from 3D sensor. (c): The 3D human body model used as reference to make the patient's 3D model. (d): The 3D body model of the patient in AP position manually created based on the reference body mesh to match the raw mesh, with the 3D model in (e) RAO and (f) LAO positions altered from the AP position through animation. All the 3D body models share the same UV map layout (g).

3.2.3 3D Human Model Construction

Due to its limited perspective view, a single 3D structured light sensor only extracts partial meshes of the patient's position. While this limitation can be mitigated in future studies through the installation of multiple 3D sensors, definitions of the unmeasured regions were resolved by using a full body reference mesh purchased for this purpose (Turbosquid, New Orleans, LA). The reference contour mesh of human body was

manually edited using the sculpture tools in 3D animation software (Maya, Autodesk, Mill Valley, CA) to fit the partial meshes of the patient's AP and PA positions by expansion. The body regions needing most additional reference mesh information were the feet and hand areas.

The manually edited mesh accuracy was evaluated by iterative calculation of the Hausdorff distance[111], that estimates a metric of distance from vertices of the edited patient body mesh to the nearest point on the contour surface mesh extracted from the 3D sensor. This Hausdorff distance evaluation was used to provide feedback for editing of the patient's 3D model. Many reference body model shapes are available for use, such that a reference model can be chosen which closely matches the body mass and relative shape of each patient. The reference body models include animation skeletons used for altering the body model into different limb and body positions. During the editing process, the skeleton of the body model was also adjusted to fit the body shape and size, providing the ability to move the arms and legs to match the 6 positions as well as possible, as illustrated in Figure 3.2.

Computer animation software uses a surface space mapping plane called the UV map, which is a 2D layout presented or displayed on a square, to record the surface texture of the animated 3D mesh[112], as illustrated in Figure 3.2(c). The UV map is usually represented by a 2D square image and is invariant in animiation.[112] In Stanford techniques, the patient deformed to different positions during the treatment, which made it hard to sum up dose values directly in the patient's 3D model. However, the UV map is invariant in the deformation process of moving the limbs or posture, which makes it the ideal tool to examine total Cherenkov/dose summed from all fractions. The UV map is

created in Maya by manually defining the cutting seams, which are the boundaries of textured region in UV map. Most vertices in the 3D body model map to a single point in the UV map. In some regions, such as the edges and corners of the cutting seams, a single vertex corresponded to multiple points in the 2D UV map. The correspondence relationship between the UV map and 3D model was recorded with the 3D body model and used for texture mapping and texture recording in later steps.

The reference body model has the built-in animation skeleton. Through the animation skeletons, the 3D body model based on the AP and PA positions was altered to those of other positions, such as the LPO and RAO positions. The animation process was manually done by rotating the joint of the animation skeleton, to simulate the patient's movement and to change their position. After altering the animation skeleton, there were some mismatched regions between the body model and the body contour mesh. Additional detailing steps using the 3D editing tools are needed to make the body models to fit the contour meshes that are extracted from the 3D body scanner.

3.2.4 3D Dose Distribution Analysis

After flat field correction, the cumulative Cherenkov image of each position were converted to dose distribution with the Cherenkov to dose conversion ratio.[21] The conversion ratio were derived by the ratio of the diode measurement in the chest region to the mean Cherenkov intensity in the corresponding region. Projective texture mapping[76] used a ray-tracing algorithm, which is one of the classical rendering algorithms in computer graphics. As shown in Algorithm 1, the method tracks the light rays from each pixel of the Cherenkov camera sensor back to the 3D body model of the patient. The ray tracing algorithm located the vertex corresponding to the image pixel through the first

intersection of the light ray and the 3D model. If there were no light rays intersecting with one vertex on the 3D body surface, the texture value of that vertex was assigned zero. The texture values of the vertices in the 3D model were mapped onto the points in the UV map and the 2D texture distribution was interpolated from the texture values of the individual points in the UV map.

Algorithm 3.1: Projection Texture Mapping

Input: Cherenkov/Dose Image I_{ch}, 3D body model G, UV map correspondence **F**
Loop: Pixel (m, n) in I_{ch}
    Generate camera ray **R**(x, y)
    If R(m, n) intersect with G at location (x, y, z)
        Search the UV location (i, j) of (x, y, z) through (i, j) = **F**(m, n)
        UV(i, j) = I_{ch}(x, y)
    end if
end Loop
Output: UV as the Cherenkov/Dose UV map of the 3D body model

The projective texture mapping method included the modules of ray tracing and the interpolation of the texture values. The ray tracing module was an interactive graphical user interface implemented in VTK toolkit (Kitware, Clinton Park, NY) with support from DoseOptics, LLC. It read the dose distribution images and the patient's 3D body model, to create the textures on the 3D body model. The projection approach used the Cherenkov camera information from the positional calibration file, such as its location and orientation relative to the patient to simulate the Cherenkov camera view of the patient. However, the camera calibration was not done at the same time of the patient's Cherenkov imaging and the camera position was not fixed between different fractions. Thus, some manual adjustment, which was included in the module, was needed to tune the Cherenkov camera parameters for accurate texture mapping. The interpolation module was implemented with the scatter points interpolated in MATLAB (Mathwork,

Natick, MA), converting the dose distribution from the 3D body contour to the UV map that represented the body skin in 2D.

With the dose distribution recorded over the 3D body model and the 2D UV map, it is then possible to complete further analysis. For example, in this study, the mean and the standard deviation could be extracted from the area dose distribution, and body components that had significantly higher or lower overall dose could be identified form the cumulative dose values of the whole treatment.

Algorithm 3.2: Statistical Analysis of Dose Distribution on Body Area

Input: Dose Distribution UV, Triangulated 3D body model G
Loop: Face f_i in G
    A(i) = Area($f_i(1), f_i(2), f_i(3)$)
    D(i) = mean($UV(f_i(1))$, $UV(f_i(2))$, $UV(f_i(3))$)
end Loop
Output: A as the arrays of area for all faces, **D** as the dose value for all faces.

Even though the UV map is a two-dimensional array representing the dose distribution over the patient's surface, they cannot be used directly to analyze the area dose distribution due to the area distortion of converting 3D model to 2D area. In order to analyze the area dose distribution, we assumed that the planner geometry with small area had uniform dose distribution. The 3D model of the patient's body is consisted of multiple polygonal faces with small areas. The body model was first triangulated to make sure all faces on the surface of the body model were triangles. The area of each triangle was calculated from the locations of its vertices. The dose values in the vertices were extracted from their UV locations, and the dose value of the triangle were derived as the mean of the dose values in its three vertices. Finally, the area distributions of the dose value were derived by calculating the areas and dose values of all triangular faces in the 3D body model.

3.2.5 Verification Analysis on a Cylindrical Phantom

A basic phantom experiment to validate the positional accuracy involved imaging a cylindrical water bucket as a tissue phantom, which was placed on a rotary table on the TSET stand, with its center aligned with the height of the Linac iso-center. The Cherenkov and 3D scanning cameras were placed similarly to the human TSET work, to extract the 3D contour of the bucket and image Cherenkov emissions. A turntable was used to rotate the water bucket, emulating variations in rotational position of a TSET patient. Pieces of black marker tape were affixed to the outer wall as marker fiducials that could be viewed by both cameras.

The cylinder was irradiated by 6MeV electron beam to capture Cherenkov imaging. The gantry angle was set at 270 degrees so that the gantry head faced the water bucket along the isocenter line. In the experiment, the bucket was irradiated twice with turntable rotations. For each rotation angle, 200 MU of electron beam was delivered to the bucket. Before this experiment, the locations of the tape markers in the film edge were recorded as texture in the 3D bucket model, using a marker plate on the top. The black tape markers were visible in the Cherenkov intensity, such that these locations could be compared with prerecorded texture in the 3D bucket model to measure the accuracy of projective texture mapping.

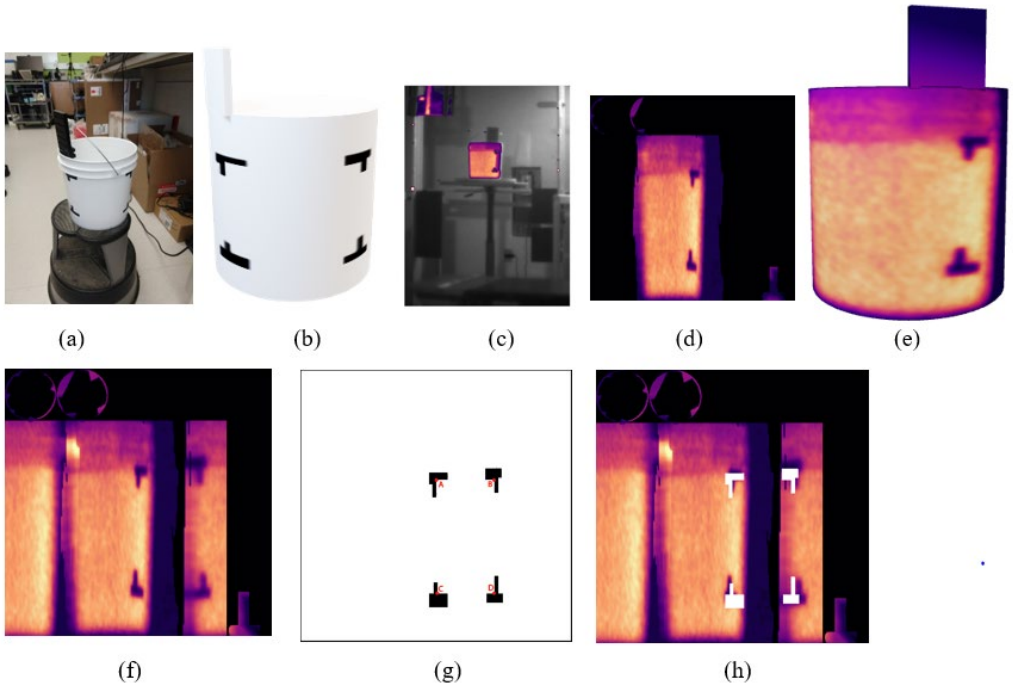


Figure 3.3: Phantom study using a water filled bucket: (a): photograph of the bucket with black fiducial corner markers and reference positioning plate above. (b): a simplified 3D model of the water bucket from the surface scan. (c): the cumulative Cherenkov image of the water bucket in one rotational position, with corresponding Cherenkov UV map layout (d) and Cherenkov texture on the 3D model (e). (f): The cumulative Cherenkov distribution from the water bucket in rotations 1 and 2 projected onto a UV map for the bucket. (g): The locations of the black fiducial marker corners in the UV map with a red dot marking the corners that used to evaluate spatial accuracy, and (h) compared with the cumulative Cherenkov distribution.

3 Results

3.3.1 Cylindrical Phantom Study

The experiment using the cylindrical phantom was to test the accuracy of the texture projection. During the process of projection texture mapping, a square plate on the top of the cylinder was used as fiducial maker to provide a reference for the camera location. The black marker taped corners were used to test the spatial accuracy of texture mapping, as shown in Figure 3.3(h), for Cherenkov distribution in the UV map, to match with their real locations on the UV map. Four corner locations, marked by the red dots, were used to evaluate the spatial accuracy of the projection algorithm. The deviation of these four locations are 10.1mm (A), 8.7mm (B), 7.5mm (C), and 12.8mm (D), with the mean value of 9.8mm. These distance errors would result in a minimal lateral dose error for regions within the bucket, likely less than 1% because the dose intensity map varies slowly with lateral position. However, if these distance errors were near the edge of a sharp fall-off in dose, then there could be considerable dose mapping error.

3.3.2 Human Study

The mean Hausdorff distances are listed below representing the difference between the raw meshes extracted from the 3D sensor and the model mesh created manually. The mean distance, which is the average distance from the vertices in the raw mesh to the model mesh, was between 1.84 and 3.23 cm with a mean of 2.54 cm (as listed in Table 1). The 3D sensor extracted high resolution raw meshes of human body, with many noisy regions, especially on the boundaries. The mesh resolution of the reference model is much lower, which provides a lower number of control vertices for manual mesh editing. The lower resolution of the reference human model can be regarded as the 3D fit of the

raw mesh, resulting in the deviations of the reference model from the raw mesh, represented by the Hausdorff distance metrics.

These distance errors would translate into Cherenkov intensity projection errors in estimating the dose on the skin surface at these positions. There is no exact translation in how a distance translates into dose because the dose values vary across the body, as some distance errors would provide very little dose error, but some distance errors such as those near the edge of the patient could produce 100% dose error.

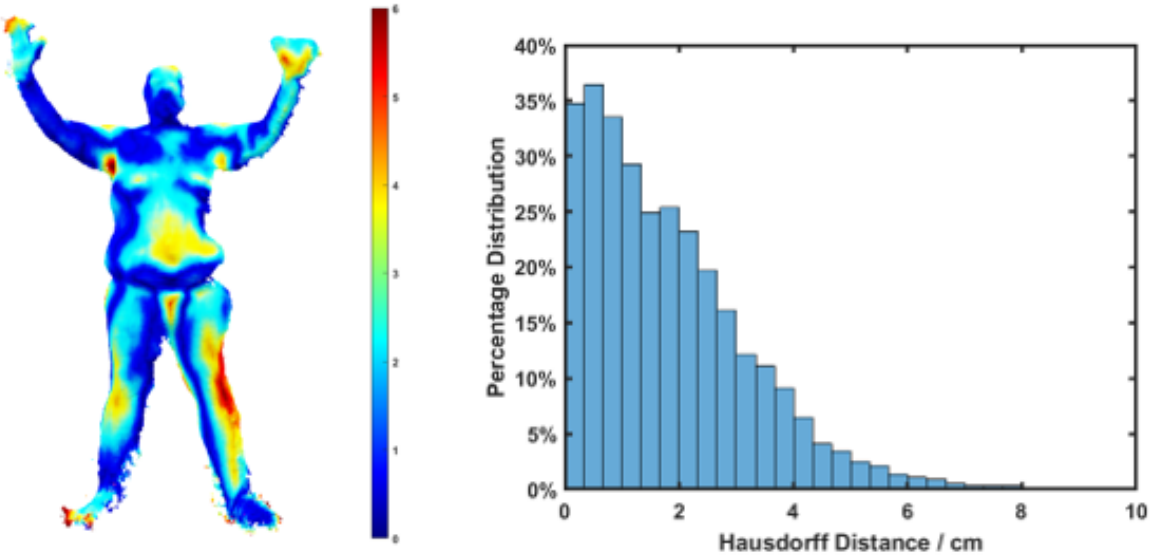


Figure 3.4: (a): Hausdorff distance distribution from the raw mesh to the model mesh in AP position, (b): Histogram of the Huadoroff Distance distribution in AP position, with the mean value of 2.54 cm for all positions.

Table 3.1: The mean Hausdorff distance from the extracted raw mesh to the body model for the 6 positions in TSET treatment

Positions	AP	PA	LAO	RAO	LPO	RPO
Mean Distance/cm	1.84	3.23	2.81	2.18	2.23	2.95

The body surface analysis workflow using computer animation techniques were verified on two patients. Patient 1 had a more obese body shape, while patient 2 was thinner. For each patient, the dose distributions were derived from the cumulative Cherenkov images in all 6 positions. The dose distributions were then projected into the patient's 3D models and recorded on the single UV map. The final cumulative dose distribution was derived by summing up the dose distributions of all positions on the UV map.

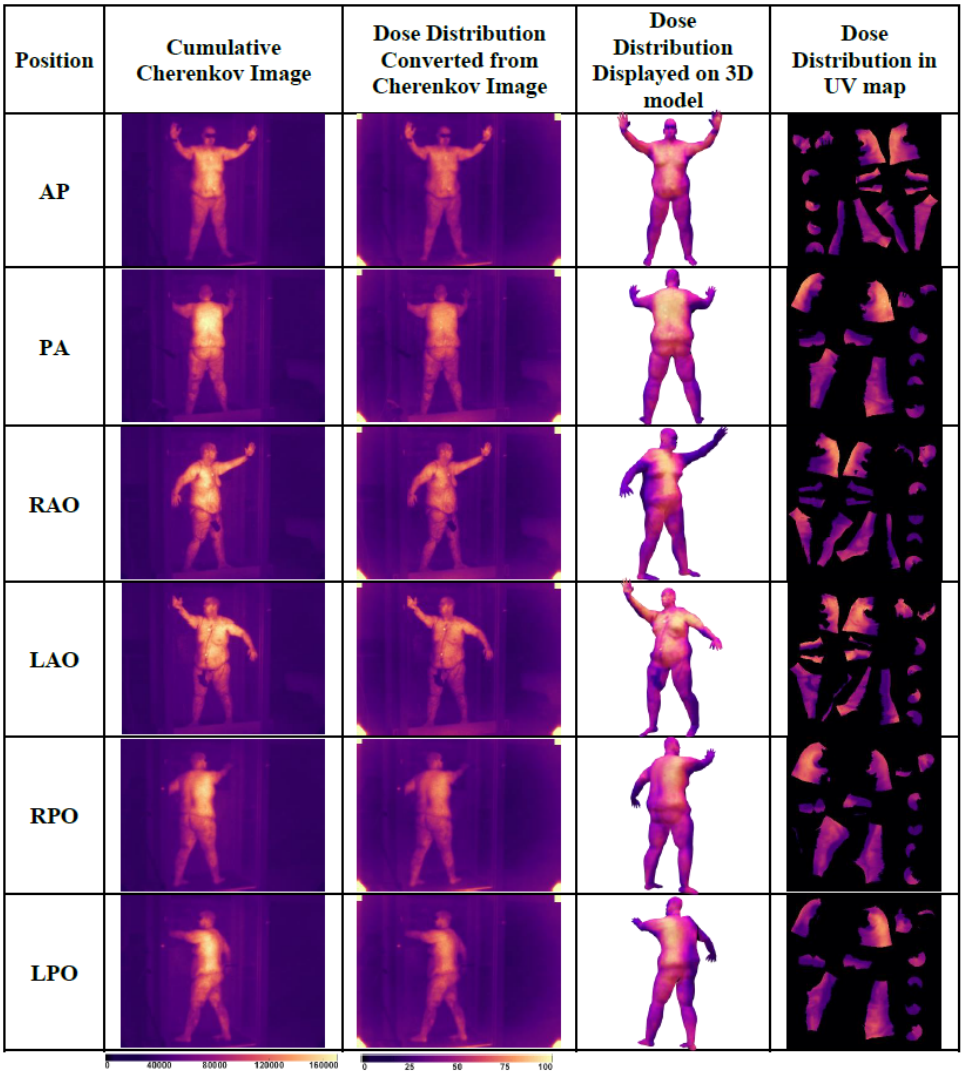


Figure 3.5: The summary of the results of 3D Dose (cGy) /Cherenkov (Intensity count) study in TSET Patient 1.

This 3D dose visualization can be used to find regions of low or high exposure from the electron beam treatments. Cumulative dose summaries across all positions can be created by summing across the sets of 2D UV data and reprojecting onto the 3D model, as shown in Figure 3.6. The locations of the fiducial feature points in the texture, such as the scintillators in AP, RAO and LAO, match the physical attachment locations, which are near the region of the chest and umbilicus.

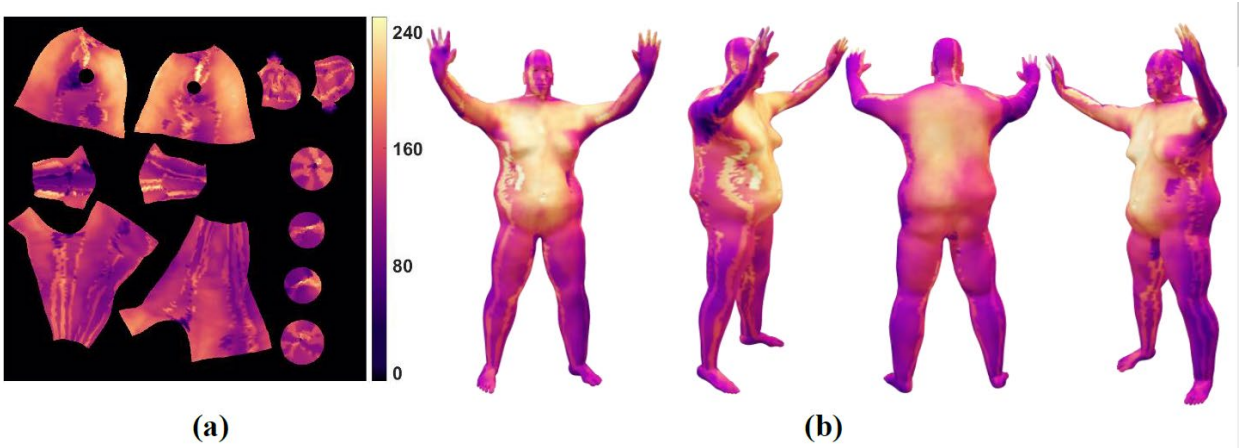


Figure 3.6: The cumulative dose distribution of Patient 1 by adding up dose distributions (cGy) in UV map for 6 positions, visualized in UV map (a), and the patient's 3D model in AP position with different perspective views (b).

The similar workflow was applied to Patient 2 to study the dose distribution over the body. The area dose distributions were derived for both patients from their dose distributions on the 3D models and UV maps. The histograms of their dose distributions in reference of the dose value measured on the umbilicus are plotted in Figure 3.7, with the dashed red line representing the mean values of the dose distributions. The refence dose of Patient 1 measured at umbilicus by OSLD is 198.4cGy, compared with the mean dose derived from Cherenkov of 147.0 cGy and standard deviation of 35.3cGy. The

reference dose of Patient 2 is 186.9 cGy, with the mean dose derived from Cherenkov of 155.8 cGy and standard deviation of 52.7cGy.

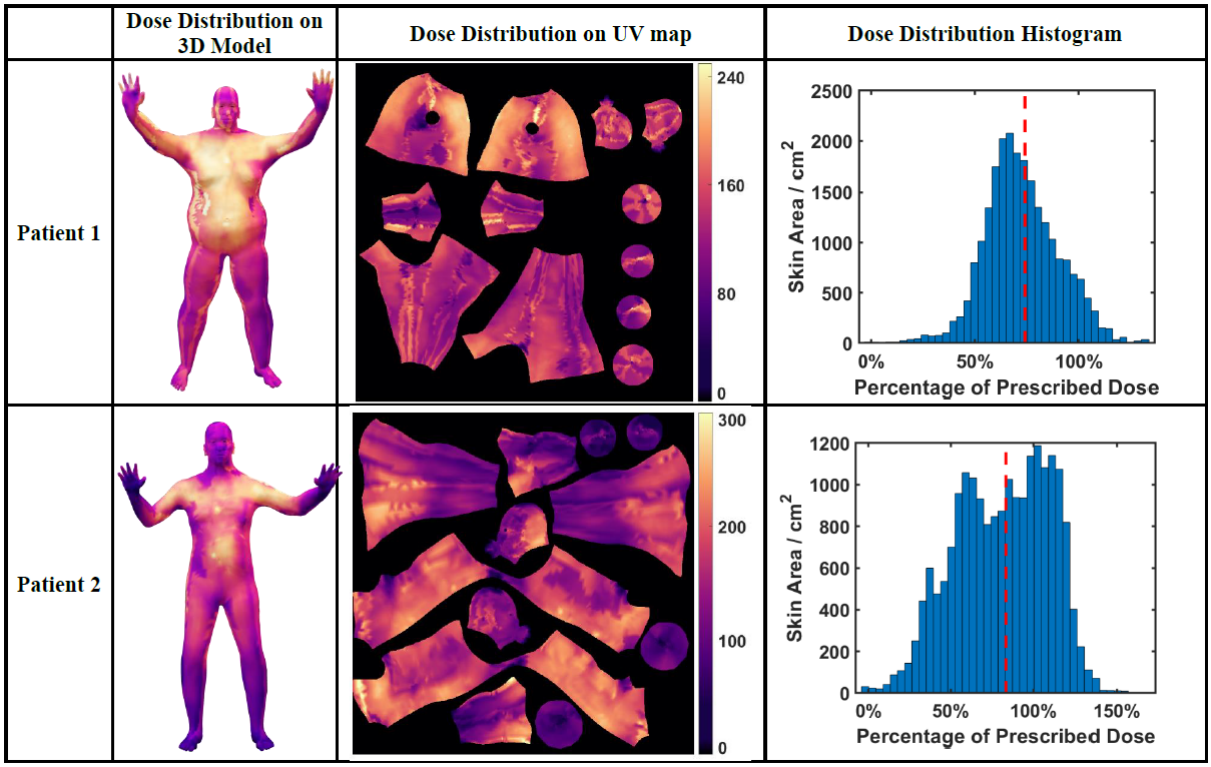


Figure 3.7: The comparison between the dose distribution of patient 1 and 2 throughout the TSET, represented in the 3D model, UV map and dose distribution histogram.

The statistical analysis generally matches the *in vivo* diode measurement in different locations of the patient during the TSET, as the chest and the umbilicus regions had the higher dose while the regions of legs, feet and hands had lower dose. The scintillator also affected the dose reading from Cherenkov imaging as the scintillation signal was much higher than the Cherenkov signal.

3.4. Discussion

The cumulative 3D dose distribution provides a possible comprehensive surface view to display the total dose distribution from different body positions. Visualization of this total

dose distribution shows that the body trunk coverage was relatively uniform. However, there are clearly regions of overlap between the fields that lead to the bright lines present in the lateral areas of the legs and body. Simulation studies indicate that the vertical lines represent true overlap regions between the different fields, which makes these regions over or under exposed as compared to the midline. Perhaps most interesting observation are the areas of very low Cherenkov intensity which were not expected, such as the underside and back of the arms, the head, and the lateral aspect of the legs. It is worth reiterating that Cherenkov intensity is known to be slightly different from dose, due to attenuation by blood, body hair, etc. So, the lower intensity at the legs is not entirely indicative of lower dose at the legs, however lateral or rotational variations in the dose distribution are likely indicative of dose heterogeneity. Additionally, features such as the back versus the front of the arms are likely indicative of lower dose levels.

The error of the final cumulative dose distribution could be attributed to two steps in the workflow. The first type of error derived from the interpretation of the dose value in the 3D Cherenkov imaging. After camera and dose calibration, the dose interpretation could achieve good agreement with OSLD measurement (within 6.1% error) and diode measurement (within 10.9% error).[15] The second type of error derived from the geometries of the cameras and projection texture mapping. The projection error was between 2 to 3cm in phantom measurement and this was the focus in this study. The 3D dose/Cherenkov analysis on both a patient and the bucket phantom tested the limits of spatial accuracy in this setup, for the current geometry of a single Cherenkov camera and a single 3D body sensor camera. The slight differences in positioning between the two cameras provides some non-overlapping spaces from the differences in perspective view,

although this was largely solved by using a common reference mesh that was warped to fit the body surfaces. In addition to testing the positioning accuracy, the projection of Cherenkov light onto the human animation mesh can be ensured for accuracy by testing for coverage to match at geometrical features on the patient body, such as the joints and the edges of the body in view. These implicit fiducials, together with other fiducial markers, such as the scintillation dots at the chest and umbilicus regions, serve to ensure that Cherenkov mapping can fit the body mesh within the accuracy specified by the bucket testing. There are peripheral limits on the perspective view of the Cherenkov camera and 3D sensor, as well as choices in lenses and positioning used on the cameras. In the positions such as PA, RPO and LPO, the TSET stand occludes the image view of the left hand and arm, which make the corresponding 3D body model miss texture regions on the left hand and arm. The 3D sensor did not have high enough spatial resolution to extract the shape of hands and feet, which makes the 3D textures in these regions different from the Cherenkov images. The handles blocked the line of sight for the 3D sensor in the hand regions. The shielding regions in the patient's lower body limited the perspective views of the Cherenkov camera, as shown in the RAO and LAO positions. Each of these limitations can be overcome by a stand that incorporates multiple 3D scanning cameras within the stand itself and this design is in progress.

Many modules in the workflow of projecting dose/Cherenkov images onto the patient's 3D body model can be automated, even though some manual labor is needed for some modules. The workload of editing the patient's 3D model is similar with contouring work in treatment planning system. But the procedures of editing the body model is clear and the number of editing tools is limited. The health care practitioner can be familiar with

the editing tool after the training similar with CT contouring in treatment planning system. The premade 3D human body models with animation skeletons save significant work of editing and animating the body models. With the premade 3D model, users only need to detail the body surface with common sculpture tools to match the patient's raw mesh scanned from the 3D sensor. The 3D software with calculation module of Hausdorff distance can automatically create the Hausdorff distances between the body model and raw meshes, which provides quick feedback of the performance of mesh editing and animation. After the body model is created for one position, the UV map can be constructed automatically from the 3D software. In the projective texture mapping module, manual input is needed to tune the camera view angle and position. The rest of the texture mapping can be done automatically by software, with the dose/Cherenkov distribution in the UV map as the output of this module. The dose/Cherenkov UV map is finally used to texture the 3D body model by the 3D software. Therefore, in the whole workflow, manual inputs are only needed to create human body models for the 6 positions and adjust the camera parameters in the texture mapping process. In future improvement, the camera alignment can be done once, with the fixed installation of the Cherenkov camera in the room. The workload of the editing and animating body mesh can be further reduced with more types of reference body model, allowing the user to choose the reference body model that is closest to the new TSET patient.

The distance errors from mesh creation are critical to minimize, because they can directly influence dose mapping errors, which would be largest near the periphery of the body. The key factor in this process is to keep the lateral errors as small as possible so that the projection of dose onto the 3D mesh does not have large error, because a distance that

move the body out of the project would have a 100% dose error, for example. Otherwise, lateral shifts in the center of the body would not be that large, likely being just a few percent dose error. In the shifts reported here, the alignment of the fingers and toes was not highly accurate due to limited resolution of the mesh, and so the dose values at these locations would be unreliable. However, the mesh accuracy throughout most of the body had a median Hausdorff distance of 2-3cm error, which would translation to a mean dose mapping error of 3-5%, but the range could be anywhere from 0% up to about 20%. While the mean dose error is reasonable, the absolute error range would be unacceptably high, and so the key factor in this type of display is to place accuracy error bounds on the resultant dose map, with calculated confidence intervals estimated based upon the known Hausdorff distances.

Even though the Stanford technique is the only approved techniques in our sites, the workflow introduced in this study can be easily applied to other main TSET techniques. The other TSET techniques do not require the patient's anatomical movement during the treatment. Thus, the animation procedures in the workflow can be skipped. In the technique using rotary table, the researcher extracts the 3D body model of the patient's initial position. During the treatment, the researcher can take the Cherenkov images and record the rotation angles without the 3D scanners. The Cherenkov/dose images are projected to the body model from the corresponding rotation angles using the projection texture mapping methods and the cumulative Cherenkov/dose distribution is derived through summing up the distribution on the UV map.

3.5 Conclusion

Three-dimensional modeling and animation methods can be used to visualize and analyze the accumulated dose in TSET via display of the summed dose/Cherenkov images on a single body surface. With the help of the 2D UV surface mapping between the 3D surface maps in different body positions, the total dose/Cherenkov distribution can be summed to allow for inspection of the regions of field overlap and possible lack of field coverage. The fiducial markers used to test human body TSET and cylinder phantom experiments indicate that spatial accuracy in the range of 2cm is currently feasible, and is most critical near the edges of the body mesh to ensure that projection of high dose gradients are not mis-projected. Areas of poor spatial agreement were largely due to limitations of the body surface camera used, which is readily solved with a more complete camera system for surface scanning. In this preliminary study the areas of least coverage appeared to be the back of the arms and legs, while lateral areas on the torso and legs appeared to have overlap regions of higher coverage. Further quantitative investigation with full body scanning and calibrated Cherenkov to dose conversion are likely possible in future systems.

Chapter 4: The treatment simulation system of TSET

The contents in this chapter is adapter from the following literature, which will be submitted after the defense of this thesis.

Miao T, Zhang R, Jermyn M, Bruza P, Zhu TC, Pogue BW, Gladsone DJ, and Williams BB. Computational dose visualization & distribution comparison in total skin electron treatment illustrates superior coverage by the rotational technique

4.1 Introduction

Patients with T-cell lymphoma, such as Mycosis Fungoides, have lesions in the superficial layers of their skin[16,18] for which electron radiation treatment can be prescribed over the whole body surface.[113] This total skin electron therapy (TSET) should have a uniformity of dose distribution, in which the electron fields are designed to deliver approximately the same amount of dose to the entirety of the skin.[102] To fulfill this uniformity goal, a couple different TSET techniques have been designed, with the two techniques most commonly used being, (i): a 6 position treatment strategy called the Stanford technique[103] and, (ii): a rotational technique where the patient is rotated with arms up in a uniform position for the complete beam on time.[26] In this study, a comparison of dose coverage was simulated by computer animation methods, augmented by electron dose estimations for different sized curved body parts, to examine if improved individualized treatment planning and interpretation might be possible..

In the Stanford technique, the 6 different standing positions are designed to allow exposure of most of the patient skin surface, with alternating arm and leg positions and discrete whole body rotations relative to the incident electron beam.[103] In the rotational technique[26], the stand for the patient usually rotates continuously through 360° at least

once or a few times. While these methods provide reasonable coverage, inevitably there are some areas that cannot be appropriately covered, such as the soles of the feet, shoulders, perineum, and top of the head, and so they are given a separated boost electron field.[20] Some other regions sensitive to electron fields, such as eyes or nails, may be shielded with lead to lower the amount of the dose as needed in these regions.[20] So, while TSET can be customized to fulfill the dose delivery requirements to different parts of body, there is little in the way of computational dose simulation as is conventional with most other radiotherapy treatments. This was the motivation for this work, to examine a methodology for whole body TSET surface dose simulation.

During treatment, dose verification of TSET is usually done at point measurement sites, such as with diodes or OSLDs placed on the skin during treatment.[108] These devices provide cumulative dose readings after the treatment and these are used as a validation of delivery. Novel optical dose measurement, such as Cherenkov[40,41] and scintillation imaging,[114] have also been developed to measure the entire area skin dose distribution in TSET[22] and other radiation treatments.[115] However, despite their values as verification measures, these approaches provide dosimetric information only after the treatment,[22] so that if anomalies of over- or under- dose are observed the treatment plan and/or patient positioning would need to be modified ad hoc to compensate for them. It would be beneficial to have a TSET simulation system[116] to facilitate accurate and consistent treatment, especially in cases of non-standard body shape or special needs for dosimetry.

Monte Carlo dose simulation software has been widely used in dosimetry studies of radiation treatment.[117] Research has been conducted to study dose distributions in phantoms with a simple geometry of the electron beam.[118] However, most of these

studies do not include the model of the whole human body, which makes it hard to incorporate these tools clinically. The studies described here are based on use of the GAMOS Monte Carlo software to estimate dose in curved tissues, and development of a lookup table to provide an estimate of the treatment dose to the whole human body in these TSET techniques as a function of angle between the body and the incident beam.

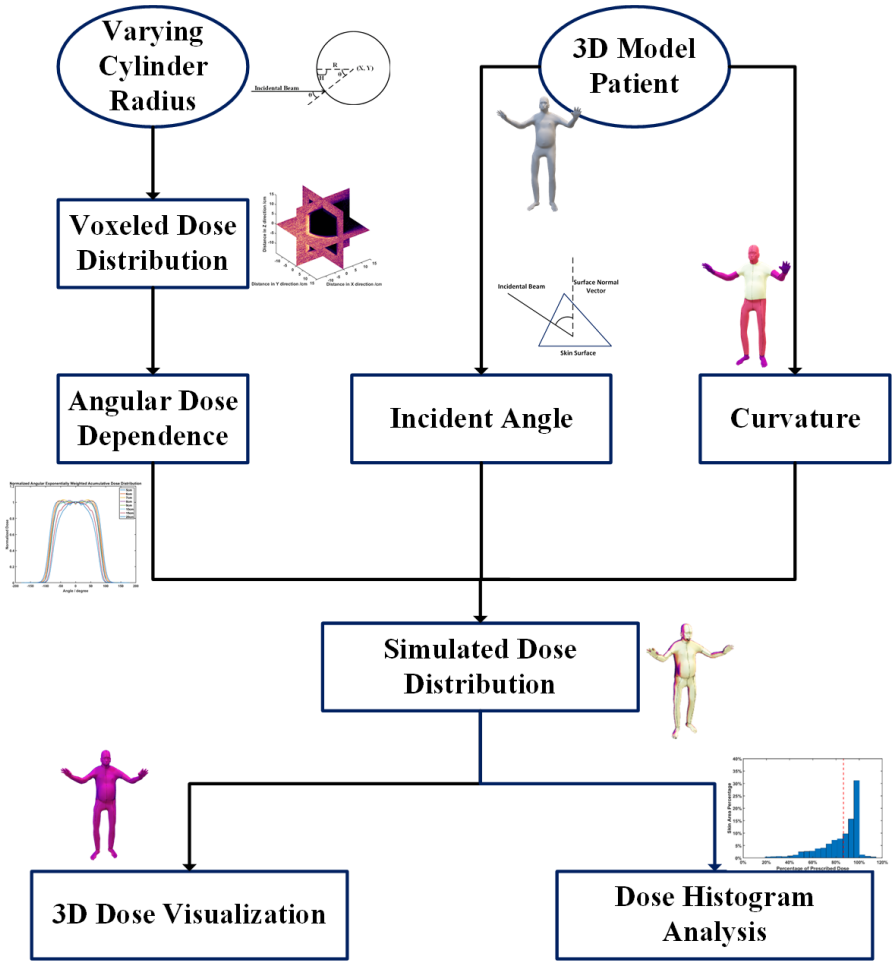


Figure 4.1: The workflow diagram of the TSET treatment simulation system used to extract relative dose distributions for TSET patients with voxelized dose from Monte Carlo simulation of different curvatures, integrated with a patient-specific surface models.

4.2 Method and Materials

4.2.1 General

The dose comparisons were conducted through a proposed TSET simulation and visualization system, based on Monte Carlo dose simulation of cylindrical water phantoms, to estimate dose on each body part. The GAMOS Monte Carlo simulation code generated dose distributions over these cylinders, and the angular dose dependence was then extracted from the dose distributions on the phantoms of different radii. For patients, the approximate tissue radius of curvature was estimated based on the surface mesh and anatomic location, for example arm, leg, finger, torso, or head. A lookup table of values from Monte Carlo simulations together with the surface normal vectors, provided the information needed to estimate relative dose at each point over the patient's body surface.

The Stanford TSET technique delivers treatment while the patient assumes multiple poses to fully cover the skin surface. Accordingly, the 3D model of each position was processed through computer animation methods in the treatment simulation system and the skin surface dose distributions were summed up for different poses, with the cumulative dose distribution displayed on the patient's 3D model[62]. Even though the patient's position was fixed during the rotational technique, the whole body of the patient was rotated with the rotary table underneath. The treatment simulation system ran for each rotation angle and the cumulative dose distribution was derived by summing up the dose distributions of all rotation angles.

4.2.2 Cylinder Phantom Dose Simulation

Water cylinder phantoms were used to approximate different body parts of patients, such as the trunk, legs, and arms, during radiation therapy[119] to estimate the relationship of dose with angle around the cylinder. In this study, GAMOS was used to simulate these electron beam studies, modeled as a 30cm × 30cm parallel beam with energy of 6MeV.[20] The initial direction of the electron beam was horizontal and directed at the centers of the phantoms for all simulations to provide dose versus angle data along the surface of each cylinder. The medium surrounding the phantoms in the simulation was air. Therefore, significant scattering occurs, and the beam diverges to cover phantoms larger than the nominal size of the un-scattered beam.

The radii of the water phantoms in the simulations ranged from 5cm to 20cm to approximate the sizes of different body parts, such as the body trunk, arms, and legs. The water phantoms were located approximately 3.25 meters from the electron source, where a patient would stand during the treatment. The medium of the phantom was defined as water. In each round of dose simulation, a phantom of one size was placed in the simulation environment and 10^8 particles were generated from the electron source to the phantom. Each phantom was divided into cubic voxels of 1mm×1mm×1mm, similar to the CT image resolution. After simulations, the dose value was recorded for each voxel inside and outside the phantom, and the dose distributions on and under the surface of the phantom were extracted sequentially.

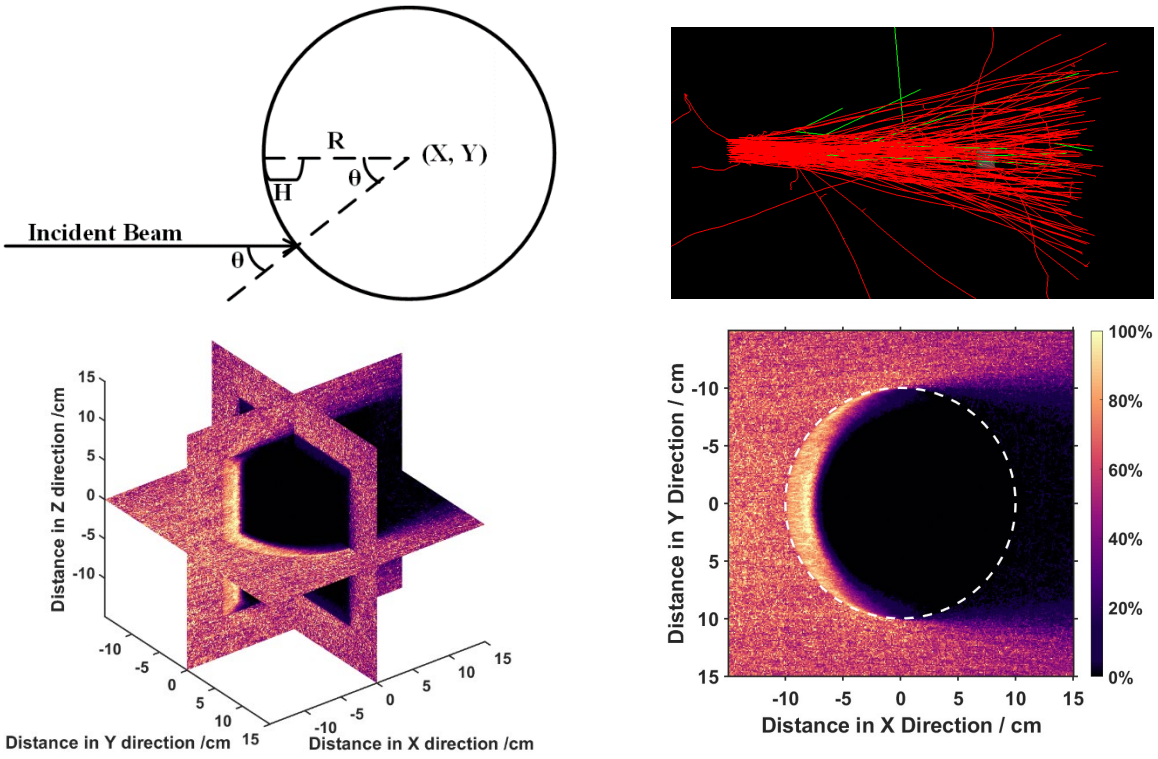


Figure 4.2: GAMOS simulation of the electron beam on a cylindrical water phantom: (a): Top view of the incident electron beam and the phantom, (b): illustration of the electron beam trajectories. (c): The 3D dose distribution viewed in slices of X, Y and Z plane, (d): The planar dose profile in the central region of the cylinder, with the white dashed circle representing the cylinder contour.

4.2.3 Dose estimation versus angle

The angular dose profiles at different depths below the surfaces were extracted from the voxelized planar dose distributions of the cylindrical phantoms. The angular dose profiles ($D_r(\theta)$) were calculated via interpolation of the planar dose profiles ($I(X,Y)$), where (X_0, Y_0) is the location of the central axis of the cylinder phantom center, R is the radius of the cylinder, and H is the depth from the cylinder surface. Notation of the variables was denoted in Fig. 4.2(a).

$$D_r(\theta) = I\big(X_0 - (R-H)cos(\theta), Y_0 - (R-H)sin(\theta)\big)$$

The angular dose profiles were calculated for depths between 0mm and 20mm, with 1mm increments. In this study, surface dose was estimated by accumulating the dose over several ranges of depths, including 0-3mm, 0-5mm, and 5-10mm. A uniform weighting function was used to accumulate the dose in depth.

The angular dose profiles were normalized to the value at 0°, the surface normal angle facing the electron beam. For computational efficiency, these normalized angular dose profiles were then re-parametrized as a function of the cosine of the angle between the surface normal and the incident radiation angle. The cosine-dose relationship was modeled as a 9^{th} order polynomial, which made it easy to calculate the relative dose value given the incidental angle of the electron beam. Cylindrical water phantoms with different radii had different angular dose responses, so one function converting cosine values to dose was derived for each radius and used in the patient dose prediction based upon the body part radius.

4.2.4 Dose distribution verification using cylindrical water phantom and radiation film

An experiment was conducted with a cylindrical water phantom to verify the angular dose distribution from the GAMOS simulation. The phantom was a plastic cylinder filled with water with radius of 14.8cm, comparable to the cylinder of 15cm radius in the GAMOS simulation. The cylinder was positioned in the TSET isocenter, which was same as the center of the TSET patient positioning. The radiation beam was delivered with upper (289.5°) and lower (250.5°) divergent fields, as used for clinical treatments, and 600 monitor units (MU) of beam delivered in each field. Dosimetric film (EBT3, Ashland, inc) was calibrated from 0 to 1000 MU and the number of delivered monitor

units was chosen to make sure the film dose was within the sensitive range. After the experiment, the film was read using a digital scanner, calibrated for linearity of response, and the angular dose profile was extracted.

4.2.5 3D Patient model extraction

In the TSET treatment geometry, a 3D camera was set up to extract the body mesh of a patient in different positions for the Stanford technique. These 3D body models included all 6 positions, which allowed for creation of surfaces for each position. Each body position had the same skin, just warped to a different stance, and so they preserved the same skin map. This was called a UV map, which was a 2D data structure to record and sum the relative dose values corresponding to the points over the patient's 3D body even when the body form was in each of the 6 different positions. Thus, the cumulative dose distribution was derived by directly summing up the dose distributions for all positions[62].

Different body parts were characterized by their radii of curvature. Because of these variations in size, the body parts had different dose responses in relation to the electron beam incident angle. In this study, the radii of curvature of different body parts were labeled using the UV map by hand with the help of flood fill algorithm.

Body Parts	**Patient with Low Body Mass**	**Patient with High Body Mass**
Body Trunk	15	20
Head	12	14
Legs	12	14
Arms	7	9
Hands	4	5
Feet	6	6

Table 4.1: The approximated cylindrical radii of curvature (cm) of different body parts for the patients used in the TSET treatment simulation

The 3D body model created for the positions for the Stanford technique was modified to generate the patient's position in rotary technique. During the rotary technique, the patient is fixed in one body position with arms up and the whole body is rotated, as would be accomplished using a rotating platform. This study assumed the TSET patient did one 360° rotation and the rotation was discretized into 6° segments. Therefore, there were 60 body models with the same position, but different rotation angles. Like the models using in the analysis of the Stanford technique, the body models in the rotational technique shared the same UV map. The treatment simulation system simulated the dose for all 60 body models and recorded the dose values on UV maps. The total dose distribution was derived by summing up the dose distribution on the UV maps of the body models with the same weight.

4.2.6 Dose projection onto the body surface

The treatment simulation system simulated the relative dose on each point of the patient's body model, based on the angle between the incidental electron beam and the normal vector of the corresponding point. This system assumed the direction of the electron beam was horizontally incident on the standing patient. It traversed all the discrete points, which were created by the 3D surface editing software, and extracted the surface normal vector for each point. The cosine value of the angle between the surface normal vector of each point and the incident beam vector was calculated by the dot product of these two vectors. The treatment simulation system checked the labeled radius of curvature of that point and used the parameterized dose to cosine value relationship of that radius to derive the relative dose value of that point. Any point other than the point assigned by the 3D

surface software could be interpolated by the dose values in the neighbor. The interpolation was processed through the scattered interpolant in MATLAB.

4.2.7 Statistical analysis of dose distribution

The statistics of the surface dose distribution could be analyzed based on the body maps and the triangulated 3D body models following statistical analysis algorithms introduced in the study of Cherenkov TSET dosimetry[62], in which the dose distribution is represented by dose histograms to get a sense of the coverage. The mean and standard deviation of the dose distribution were also derived from the histogram. In this study, we additionally used a Dose Area Histogram (DAH) to display the values, which is similar to the definition of dose volume histogram in conventional treatment planning[120].

4.3 Results

The angular dose profileswere produced by Monte Carlo electron beam simulation, shown for different radius cylinders and summed over several ranges in depth. The values of dose were summed for two depth ranges: 0 to 5mm and 5mm to 10mm, as plotted in Figure 4.3(a) and (b) respectively. For superficial depths (a), the relative dose magnitude increases with the angle of incidence, peaking near the tangential beam angles near 70° to 80° with a slight dependence upon the radii, and then decreasing to zero. The phamtoms with larger radii above 15cm, showed flat relative dose responses to the incident beam angle, presumably due to more material for build up in the tangent areas. When summed over a deeper range of depths (Figure 4.4(b)), the dose values also peaked at tangential angles for small phantom radii, however the peaks were not as pronounced as observed for more superficial depths. For the superficial depths from 0mm to 5mm, the peak dose was around 10% to 15% greater than the dose value at normal incidence, especially for small radii, while the peak dose was less than 10% greater for depths summed between 5mm and 10mm.

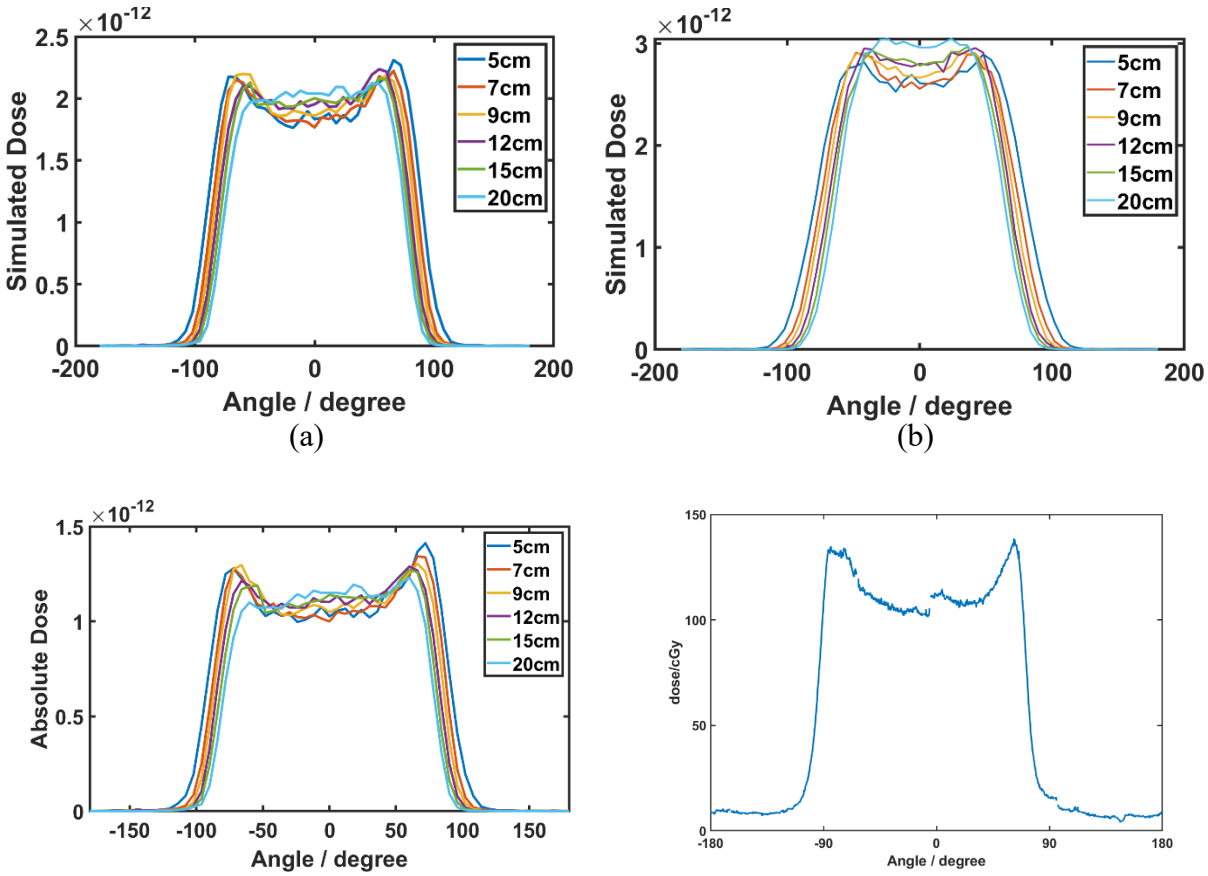


Figure 4.3: The angular dose distribution on cylindrical water phantoms with radii from 5cm to 20cm. (a): The total dose cumulated uniformly from depth of 0mm to 5mm, (b): The total dose cumulated uniformly from depth of 5mm to 10mm, (c): The total dose cumulated uniformly from depth of 0 mm to 3mm,compared with the dose distribution from film measurement around a cylindrical phantom (d) with radius of 15cm.

An experiment using a cylindrical water phantom was completed to measure the surface dose profile using the TSET technique. The shape of the dose distribution in film measurement (Figure 4.3(d)) matched that from the GAMOS simulation, with slight differences in the peak values in the tangential angle region. The dose estimation depends upon the depth below the surface, which makes it hard to choose a single definition of

skin dose for TSET, as summation of the data below the 3mm depth gives different profiles to those in Figure 4.3. In a simulation study, the skin dose was defined as the uniform summation of the dose between depths ranging from the surface to 3mm depth. The angular skin dose distributions were plotted in Figure 4.3(c) for different cylinders. In this study, the range of the depth was from 0 to 3mm, because the aim of the TSET was to treat superficial skin. The superficial dose distribution had peak value at a tangent angle.

As shown in Figures 4.3, for the same angle of incidence, the difference in relative dose value was large across different size cylinders, especially for angles between 60° to 110°. The differences are larger than the defined tolerance of ±10% from TSET guidelines, so a single cosine-dose mapping was not suitable to simulate doses across a patient's body. The radii of curvature of the parts of the patient's body normally ranged from 5cm to 20cm. Therefore, the treatment simulation strategy developed here applied radius of curvature dependent cosine-dose relationships based on the values included in Table .

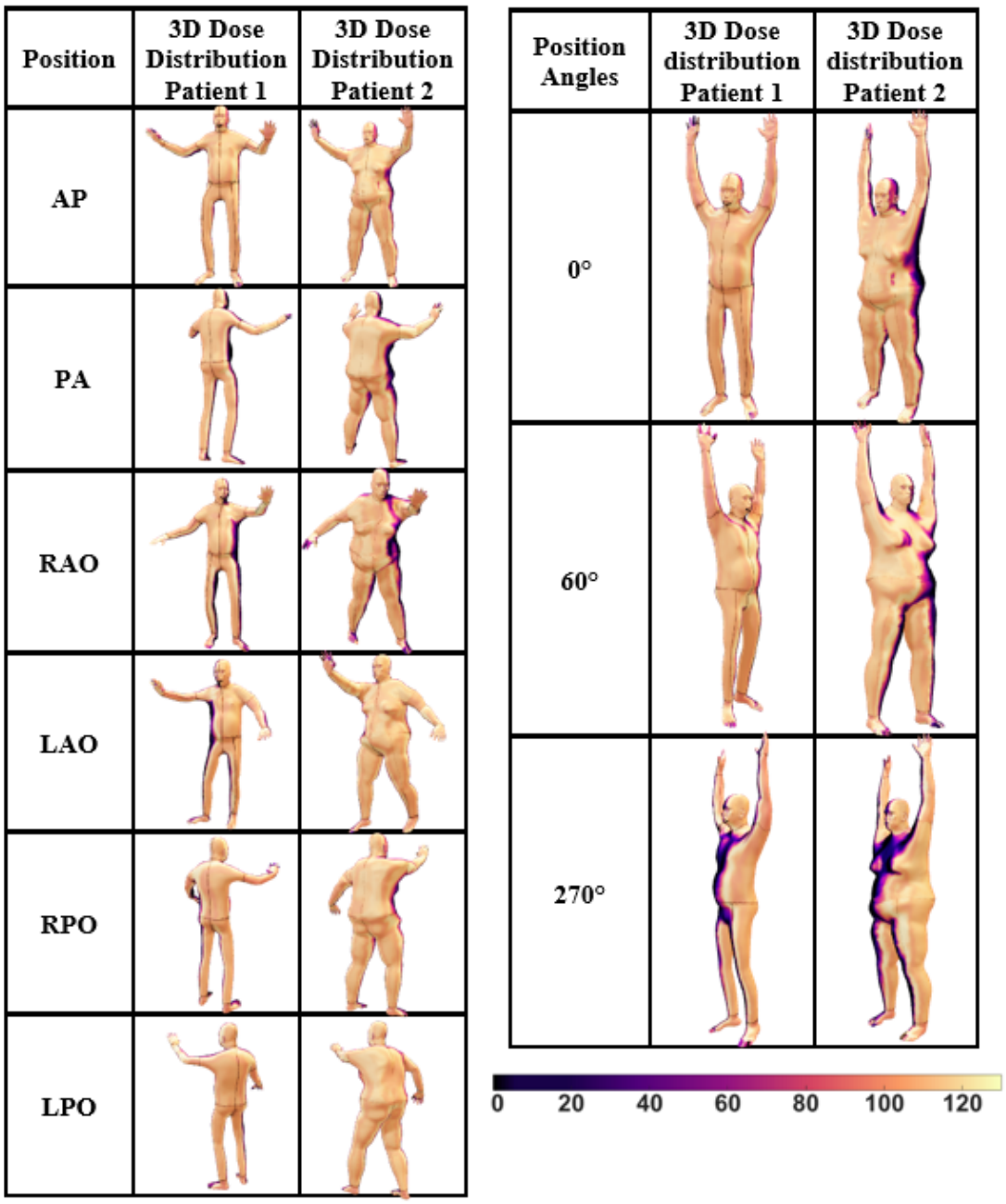
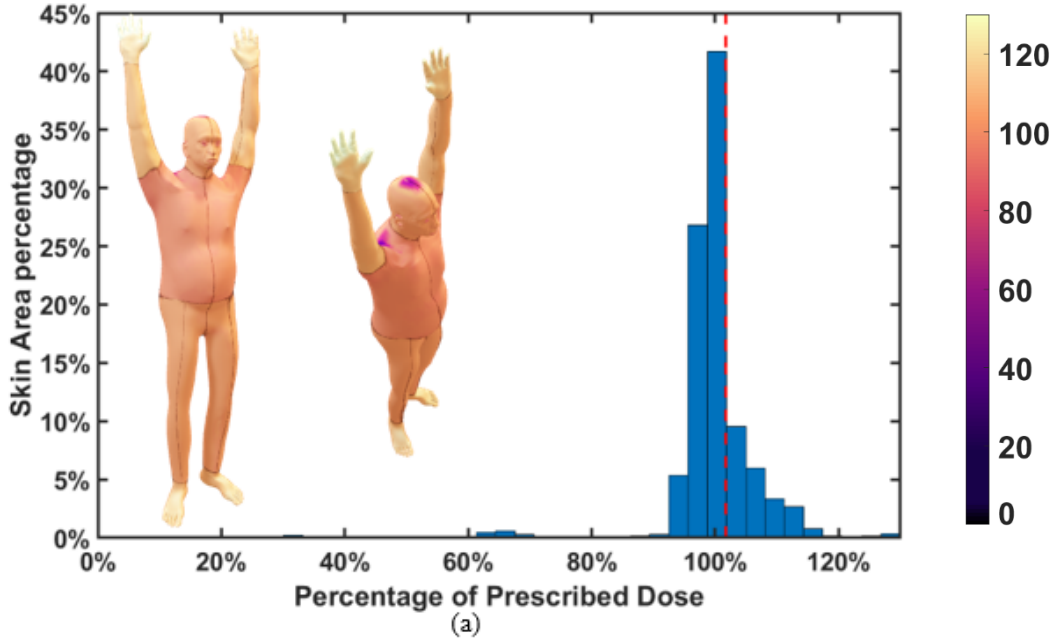


Figure 4.4: Visualization of relative dose distributions for thinner and larger patients in the 6 positions of the Stanford technique (a) and in 3 of the angles of the rotational technique (b).

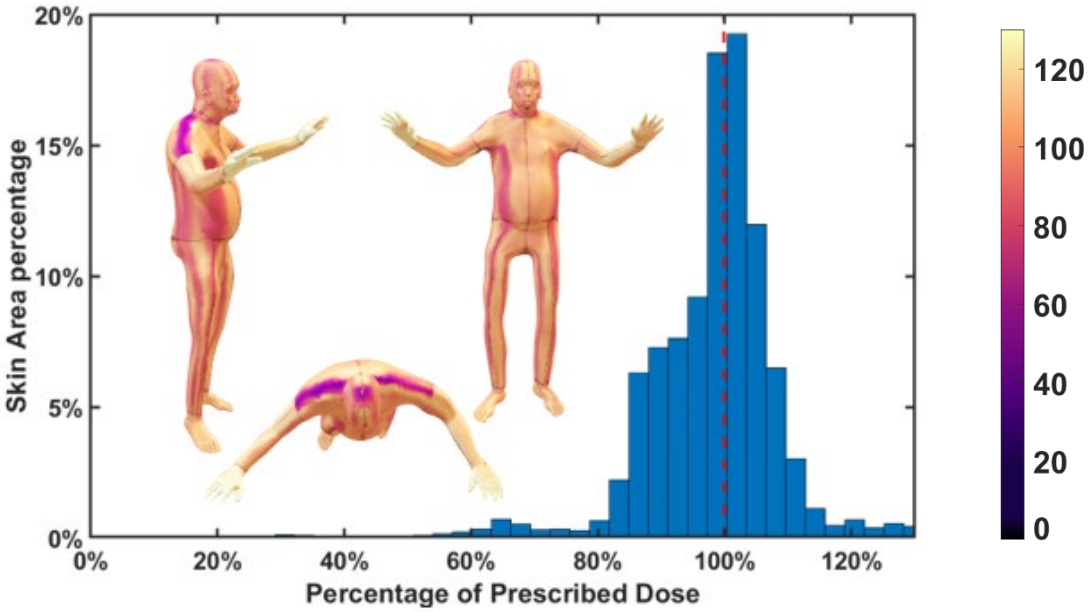
A forward treatment simulation approach was used to estimate the dose distributions for two patients with different levels of body mass in 6 positions of the Stanford techniques, as shown in Fig.4.4(a). The dose was normalized to the dose on the central axis of the incident beam, which was at the height of the belly. In each position, the body regions facing the electron beam had high dose coverage, often above 100%, as the incident angle to these regions was close to the normal. In the regions where the electron beam was tangent to the surface, the value of relative dose first increased to the peak value and then decreased. In the regions that were not exposed to the electron beam, the relative dose was not precisely zero, due to the background dose, which was also illustrated in the angular dose distribution and the cosine-dose relationship of the cylindrical phantoms. The relative dose distribution showed a similar pattern for two patients in the rotation treatment, Figure 4.4(b). However, for some rotation angles, such as 270°, the regions of the tangential incidental beam were larger than that seen in Stanford technique, because of the larger belly region.

The normalized cumulative dose distribution was calculated by summing up the UV maps of the dose distributions for different position, as introduced in an earlier Cherenkov imaging study[62], as displayed in the 3D view of Figure 4.5 (b) and (d). In the Stanford technique, large low dose regions were observed laterally on the body torso, shoulders, triceps, and legs. These regions were covered in fewer positions than the belly regions in Stanford technique, thus were the under-exposed, however the magnitude of this likely depends significantly upon the exact position of the subject and their body shape. In the rotational technique, the dose distribution was more uniform in most regions of the body surface. However, anticipated anomalies were againobserved in the perineum

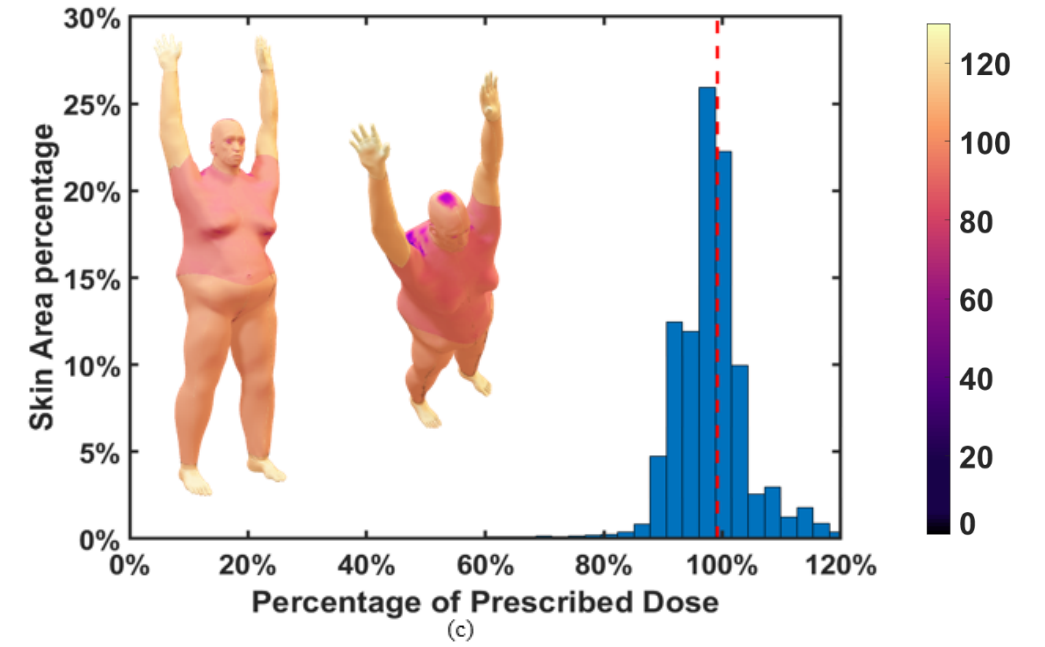
and top of the shoulders because these regions were in the tangential beam for all the rotations. The regions of perineum, and foot soles were also under-exposed in Stanford technique. These regions were usually complemented with boost fields in clinical scenarios.



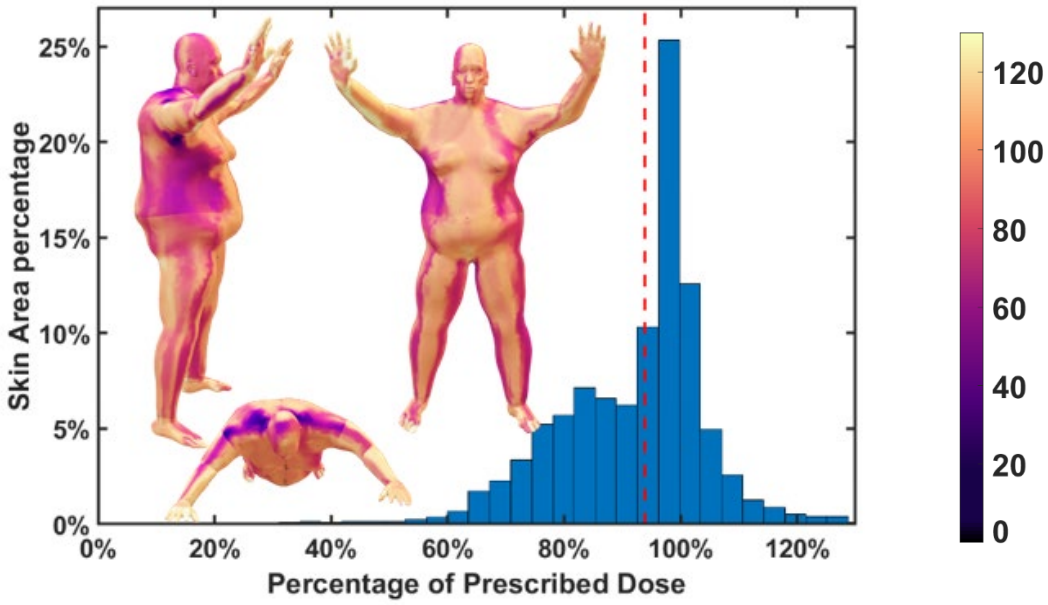
(a)



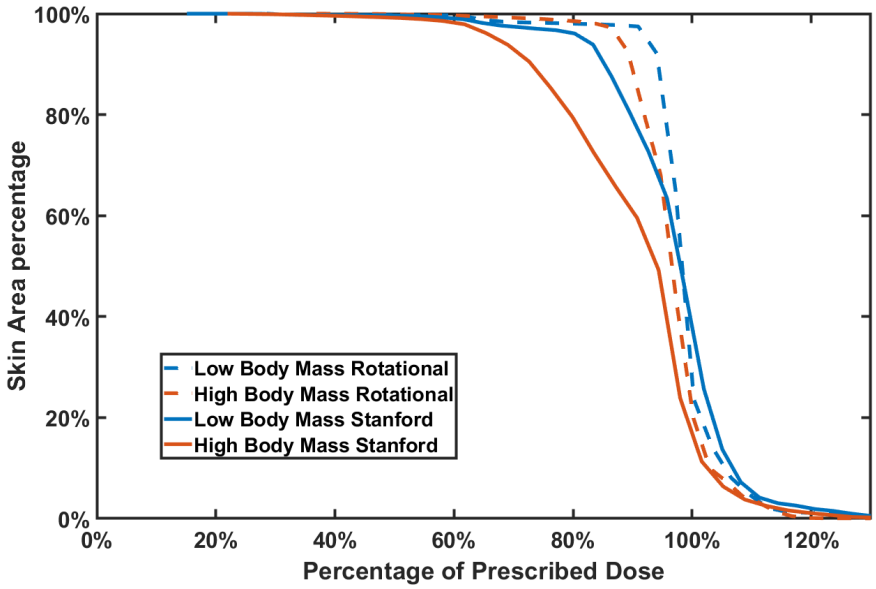
(b)


30%
25%
20%
15%
10%
5%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose
120
100
80
60
40
20
0

(c)


25%
20%
15%
10%
5%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose
120
100
80
60
40
20
0

(d)



(e)

Figure 4.5: The comparison of the cumulative dose distributions, with the skin dose defined by summation of depth dose from 0 to 3mm, for the rotational and Stanford techniques for patients with low body mass (a) and (b), respectively, and similarly for high body mass patients, (c) and (d) respectively, with inlaid dose area distribution histograms. The dose area histogram (DAH) comparison is in (e).

The analysis of surface dose is summarized by the histograms of dose values (4.5(a)-4.5(d)) and the comparison of dose area histograms (DAH) Figure 4.5(e). These data quantify the fact that the rotational technique exhibited a more uniform dose distribution than the Stanford technique for both patients with low and high body mass. In the area dose histogram, the relative dose in most regions was in the range of 100%±10% for rotational technique, while the relative dose of many regions in Stanford technique were not in the tolerance range. The DAH curve, derived from the dose area histogram, showed that rotational technique had a sharper curve near 100%, meaning that most areas had a relative dose around 100%. Statistical analysis also showed that the dose distribution on the low body mass patient was more uniform than that on the high

body mass patient, as the DAH curves of the low body mass patient fells more sharply than that of the high body mass patient in both techniques. In clinical treatment, the primary goal is to ensure that the skin dose is within ±10% of the prescribed dose, and the fractional area of this can be derived from the DAH curve. Table 2 shows the percentage area that is within tolerance of the prescribed dose, showing that the rotational technique would have better dose uniformity and that both methods had better performance in low body mass patient.

	Stanford	**Rotational**
Patient 1 (Low Body Mass)	85.4%	92.8%
Patient 2 (High Body Mass)	57.1%	89.3%

Table 4.2: Percentage of the skin area that fell within the 90% to 110% of the prescribed dose

4.5 Discussion

The TSET treatment plan analysis completed here is a viable approach to simulate and visualize the relative dose distribution on the patient's body. A comparison of the two different techniques was used to illustrate that the dose distribution in rotary technique was more uniform than that in Stanford technique. In the Stanford technique, the low dose anomalies appear to occur in the lateral tangential regions exposed from multiple positions. Lateral regions in areas of the body with the largest radius of curvature are the ones that have the lowest dose, and so this obviously occurs in the largest body mass patients and in nearly all patients in the abdomen regions. The existence of the under-exposed regions match the uneven doses and Cherenkov distributions observed in an

earlier TSET Cherenkov imaging study[62]. These anomalies are thought to be from the low tangent fields that match even with different poses. The anomaly area of the dose distribution in the Stanford technique might be potentially reduced by adjusting the individual position with more rotation. Any patient positioning error or patient movements may further increase low dose areas in the Stanford technique approach[21]. The relative dose distributions were more uniform in the rotational technique because most regions of the body were exposed to a normally incident initial beam at some point in the treatment delivery. Most skin regions were exposed by both normal and the tangential electron fields. There were still some under-exposed regions, such as shoulders, feet soles and perineum in rotational technique.[25] These regions were also under-exposed in the Stanford technique. In clinical treatment, the soles of the feet, perineum and top of the head are commonly compensated with boost fields.[20] However, consideration might be given to the shoulder areas for boost in both the rotational and Stanford techniques.

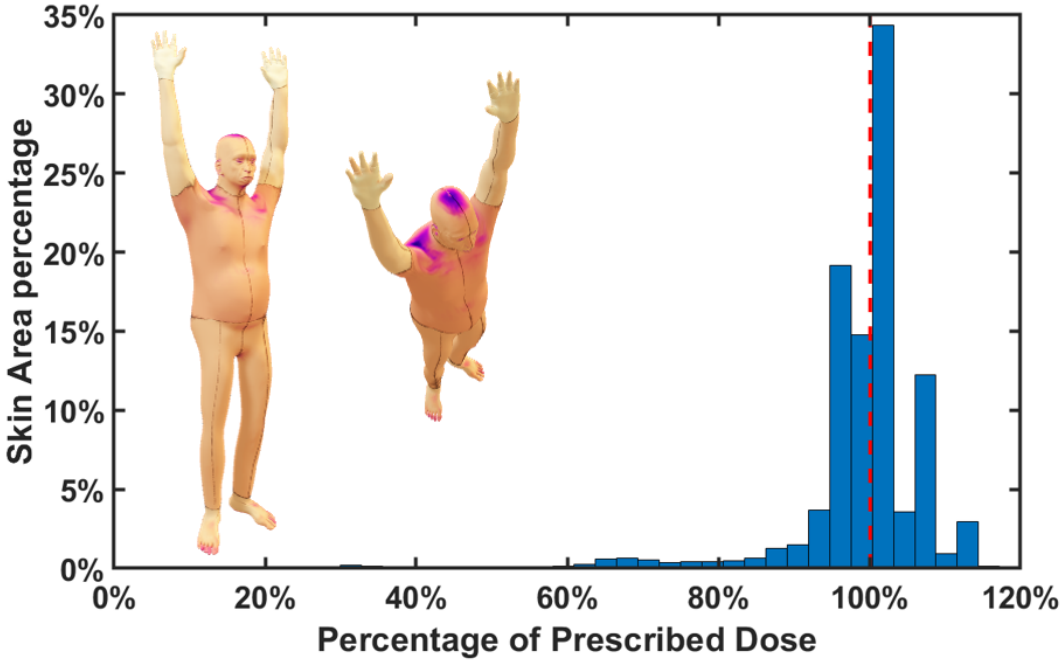
The system was implemented based on the assumption that the relative dose distribution depended upon the angle of incidence of the electron beam and the curvature of the body surfaces. Therefore, the cylindrical water phantoms were assumed as a reasonable approximation to most of the body parts[119]. Some body parts, such as shoulder joints, fingers, and toes, were not similar with the cylinders, thus the dose simulation might have deviation in these regions. However, the area of this these regions were small, and the tangential doses expected to be high, so the overall planned dose statistics would likely not be affected by these deviations in terms of low coverage.

The Monte Carlo dose simulation for the cylindrical phantoms showed that the curvature of the surface affected the relative dose distribution. In this study, the curvatures of different body parts were labeled manually, so there were some regions labelled ambiguously. For example, the shoulder was labeled to have the same curvature as the body trunk. However, the area of the ambiguous region was small, thus would not have big impact on the overall simulation of the dose distribution. In the 3D view of the dose distribution, a difference, represented by the color contrast, could be observed between the upper and the lower parts of the body, however, this was subtle in the belly region and the area with ambiguous curvature was small. Therefore, the discontinuous change in the curvature definition would not have large impact on overall dose distribution simulation and would not change the conclusions of this study.

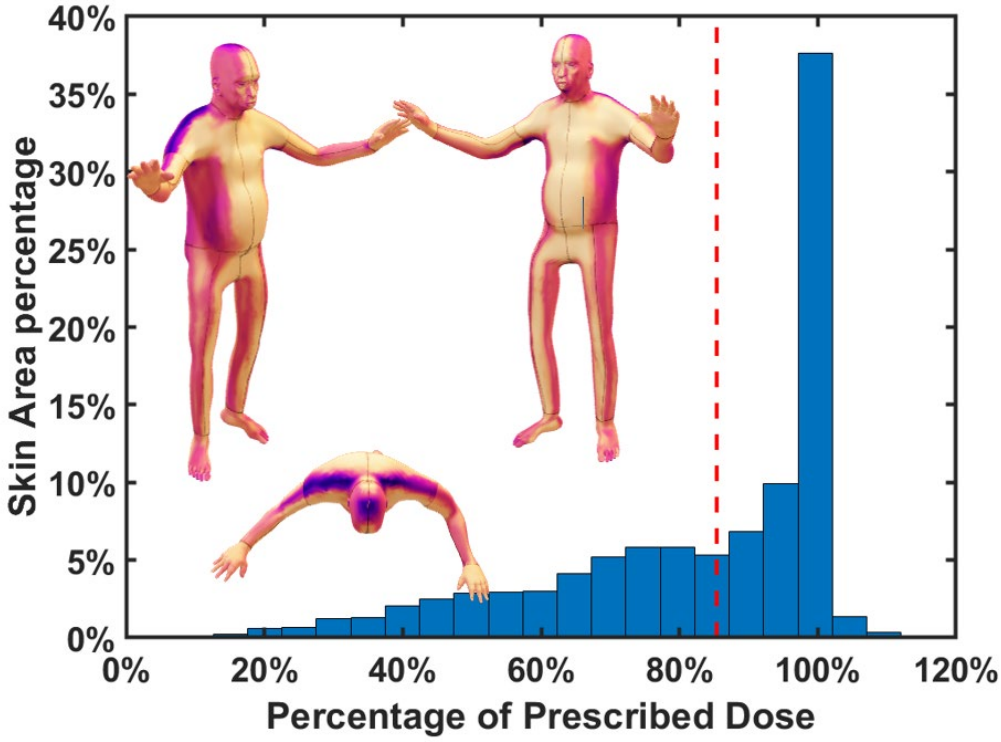
The experiment using a cylindrical water phantom provided a verification for the dose simulation as a function of angle. The shape of the surface dose profile in film was quite similar to that in Monte Carlo simulation, with slight differences of the peak values. The difference was partially attributed to the error of the film measurement, especially in the boundary region between the phantom and the air. Part of the error was also from the limited number of electron particles in Monte Carlo simulation. Thus, in these projected dose simulations the Monte Carlo data were used to define the angular distributions, because they were free from small surface artifacts that tend to plague surface dosimetry experiments.

In this study, the skin dose was defined as a uniform summation of the depth dose from 0 to 3mm at one point on the surface of the phantom or the patient's body. This definition made it easy to evaluate overall dose level at one point on the skin[121]. Treatment plans

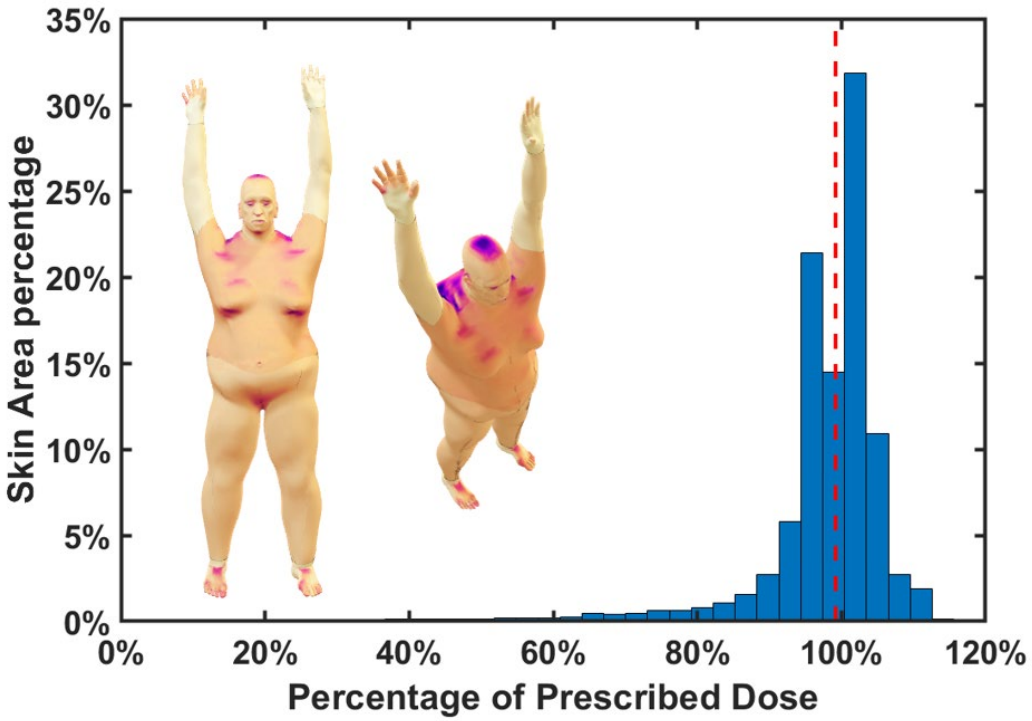
could be created with different definitions of the skin dose as a function of depth. For example, a different depth range could be used to analyze the skin dose. For example, results for the depth range from 0 to 10mm are shown in Figure 6. Overall, the cumulative dose distributions of the depth definition from 0 to 10mm were similar to those summed between 0 and 3mm. For this deeper range in depth, the rotational technique again performed better than the Stanford technique. However, the dose distribution of the depth definition showed a larger variation over the skin, especially in the Stanford technique for the patient with high body mass. This could be attributed to lower dose penetration of the electron beam to the deep regions of the tissue. In clinical dosimetry, only superficial dose, usually less than 5mm in depth, could be measured using classical dosimeters. However, thicker lesions might exist for TSET patients and the dose distribution of the deeper regions could be of interest.[18] With the novel optical dose measurement by Cherenkov, physicists could study the dose distribution contributed from deeper regions of the tissue[21]. Thus, this approach to TSET treatment plan evaluation is flexible an even can be adapted for skin dose definitions with customized weight function to be used for different planning and verification purposes.


35%
30%
25%
20%
15%
10%
5%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose

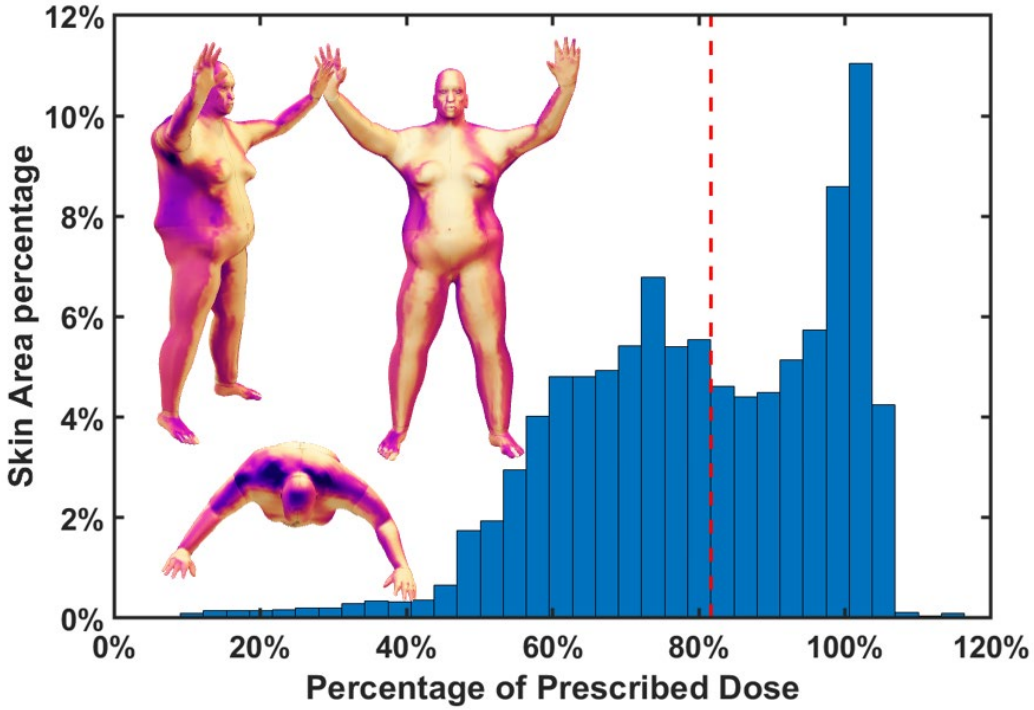
(a)


40%
35%
30%
25%
20%
15%
10%
5%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose

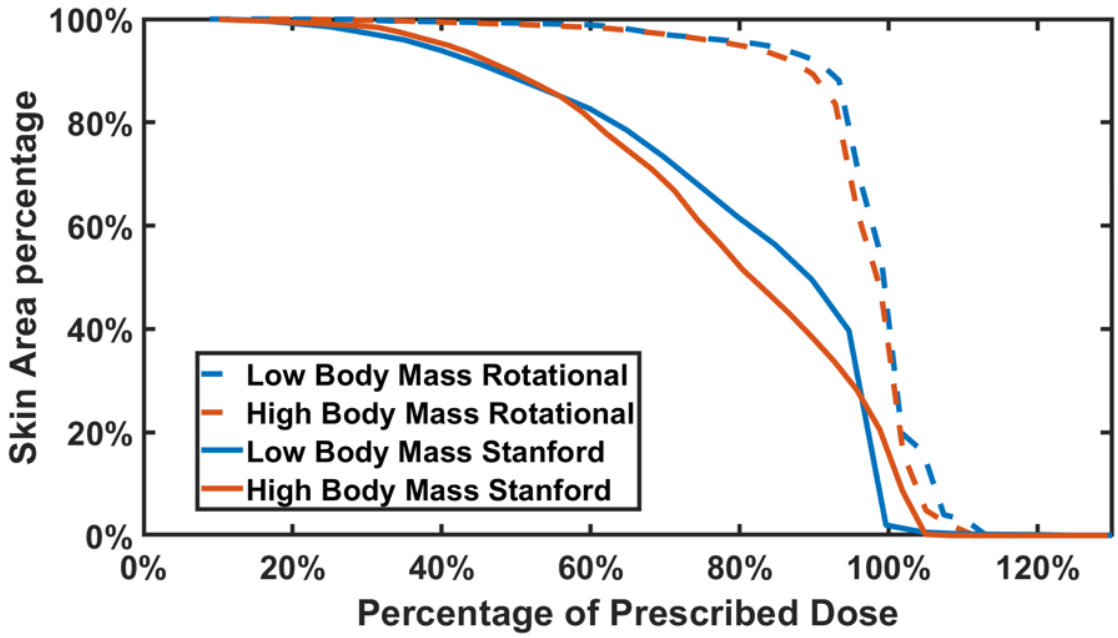
(b)


35%
30%
25%
20%
15%
10%
5%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose

(c)


12%
10%
8%
6%
4%
2%
0%
Skin Area percentage
0%
20%
40%
60%
80%
100%
120%
Percentage of Prescribed Dose

(d)



(e)

Figure 4.6: The comparison of the cumulative dose distributions, with the skin dose defined by summation of depth dose from 0 to 10 mm, for the rotational and Stanford techniques for patients with low body mass ((a) and (b), respectively), and similarly for high body mass ((c) and (d), respectively), with inlaid dose area distribution histograms. The dose area histogram (DAH) comparison is in (e).

This study compared two types of patients, with low and high body mass. The two patients showed a similar conclusion, that the rotational technique performed better than the Stanford technique in terms of dose uniformity. Additionally, both techniques, the dose distribution on the patient with low body mass was more uniform than that of the patient with high body mass. As shown in the DAH plot of Figure 4.6(e), the blue lines, representing the low body mass patient, were sharper than the red lines, in both Stanford and rotational techniques. This difference was attributed to the body curvature. As the radius of the curvature increases, tangential doses decrease, which can affect the larger

body mass patients more than lower body mass patients. Similarly, the Stanford technique tends to fail in lateral coverage because of this issue more than the rotational technique. Additionally, rotational methods that have higher number of rotations of the patient are likely less susceptible to lateral errors at the start/end of the rotation[25], although this was not explicitly verified here.

This study provided two quantification tools, including a dose histogram and dose area histogram, to evaluate the overall dose uniformity of TSET. Other criteria could also be used in the evaluation. For example, if a specific area of the skin, such as the belly or back, might be of interest in the TSET treatment, then the dose distribution in that region could be taken out for analysis by using masking method. In other treatments, some regions of the skin, such as foot soles or perineum might be shielded,[20,25] and these regions could also be masked out in dose distribution analysis. Customization could also be incorporated in the 3D visualization. By applying the thresholding, researchers could isolate the regions of under- and over-exposure. In this way, the physicists could apply appropriate boost fields and shielding to these regions.

In this study, the treatment plan simulation used 3D body models that were created from 3D surface meshes that were created from optical scans of the patients during their TSET treatment[62]. In future research, the 3D mesh could be taken prior to the treatment, with the help of a custom camera solution. A fast 3D body scanner could potentially complete scanning within seconds for each position, making it possible to get the patient's 3D body model during the patient pre-treatment visit or during their first treatment. In this way, treatment simulation could be done before the initial treatment and provide feedback such as over- and under-exposed regions, such that individualized or adaptive planning for the

treatment could occur. The 3D body scanner might ideally also help the therapists to position the patient during each treatment, similar to surface imaging approached applied in breast treatment[122]. Most of the treatment simulation work could be done automatically through software, such as the dose simulation for individual positions, dose distribution accumulation, 3D visualization, and statistical analysis. As in the previous TSET Cherenkov study, manual input was needed to create the animated 3D body model to fit the 3D mesh of different positions in Stanford or rotational technique. Here it was estimated that the workload of editing the 3D body model was similar to contouring target volumes and internal organs in classical volumetric treatment planning, and substantially less work than adaptive planning[62]. Therefore, it seems feasible to incorporate custom surface dose simulation into a clinical TSET workflow and may be especially beneficial for complex patient cases.

4.5 Conclusions

A simulation and analysis approach were introduced that used body surface visualization and surface dose kernel projection tools to quantify the relative dose uniformity of coverage in TSET, with the help of a 3D body scan of the patient. This approach can be used to simulate the relative dose distribution before the treatment to provide feedback to optimize or adapt the treatment plan as needed. We found that the rotational technique outperformed the Stanford technique based on the dose homogeneity for two different body mass patients. In terms of the general uniformity, approximately 90% of the skin area had a dose in the accepted tolerance range of ±10% of the prescribed dose in rotational techniques, while the percentage of the skin area within the tolerance range was observed to range from 50% to 85% for Stanford technique depending on patient size,

with the main area of low dose coverage observed in the lateral abdominal regions of the patient. Interestingly, this discrepancy for high body mass patients appears to be attributable to the loss of tangent dose at low angle of curvature. Future use of this approach to analyze dose coverage is possible as a routine planning tool.

Chapter 5: Conclusion Remarks

Cherenkov imaging is a unique tool to extract the radiation beam shape projected on the tissues or physical objects. Compared with film measurement of the beam shape, the contour of the Cherenkov distribution matched the beam shape configured by the Linac beam shaping modules, such as the jaws and MLCs, with a discrepancy of less than 2 millimeters or 1% of the jaw size. The Cherenkov imaging system was also able to take images of the light field and extract its shape to perform a parallel light field QA process if needed. This can be used to regularly or systematically compare the shapes of the light field, the beam as seen by Cherenkov, and the radiation dose shape as measured by film. The benefit of Cherenkov imaging is the simplicity of measurement once the camera is set up and calibrated, and the ability to image dynamic beam shape patterns to verify the Linac beam shaping modules functionality. Therefore, Cherenkov imaging could be potentially used as the tool to conduct and automate a large number of QA tasks.

The discrepancy seen between the Cherenkov and the light field measurement was attributed to the light scattering penumbra effect of the Cherenkov emission in the edges of the beam field, not the actual radiation penumbra. The light field has sharper transition of the light intensity in the edges, like the dose distribution from the film measurement, while the edge changed more gradually in the Cherenkov distribution on the surface of the plastic phantom. This penumbra effect was likely due to elastic scattering of the Cherenkov emission inside the plastic phantom and could be potentially reduced by changing different the phantom material to a less translucent material and/or with image processing techniques. Even if the discrepancy exists between the Cherenkov and the light field, the value of deviation is small and within the tolerance of the QA

guideline, when an edge detection algorithm is calibrated as demonstrated here. Thus, the Cherenkov imaging system even as presented, is an acceptable tool to fulfill system QA requirement.

The combined system of Cherenkov imaging and 3D body scanning system was a novel application to perform skin dosimetry for the TSET patients. Through the method of the projective texture mapping, the dose distribution of each position, which is estimated from the Cherenkov image, could be overlaid onto the 3D model of the patient's position. The use of the 2D UV map to sum all the texture values made it feasible to accumulate the dose distributions of all positions, and then relay this back onto the full 3D body view.

In both the 3D visualization and the statistical analysis of the patient's cumulative dose distribution, regions of the body trunk exhibited more uniform dose distribution. However, low and high dose regions were observed in the lateral sides of the patient's body, which resulted from overlap and under-exposure in the field, when a summed view of the 6 different positions was seen. If this was applied to treatment planning, it could be possible to rely on these types of results to adjust the patient's positions and reduce the regions of abnormal radiation exposure in future treatments. With predictive modeling it would even be possible to predict body coverage in real time as the patient is positioned and give feedback to the therapists to adjust their position.

Minor errors might arise from the occlusion of the camera position, deviations of camera calibration and projective texture mapping. These errors could be potentially reduced in the future by installation of multiple Cherenkov cameras and 3D body scanners, so that the occluded regions of the individual camera or scanner could be

compensated by the others. One such system is being prototyped now from TC2 body scanner, with 12 PrimeSense structured light cameras surrounding the patient to be scanned. The system acquires the 3D contour meshes of the patient in different view and stitch the contour meshes together using 3D imaging processing techniques. Compared with the single scanner used in the TSET Cherenkov dosimetry study, the scanner system with multiple scanners extracts more complete 3D structure of the patients with greater details.

The TSET treatment planning analysis used the 3D models from the TSET Cherenkov dosimetry study. For the Stanford technique, the results of the treatment planning system showed that the dose distribution was more uniform in the patient's body trunk, but under-exposed regions were observed on the lateral sides of the patient body. This appeared to be amplified in the case of a patient with a larger body habitus, v.s. a thinner patient. In the analysis of the dose distribution of the depth from 0 to 3mm in Stanford technique, around 85% of the skin area of the patient with low body mass patient had the dose within the tolerance of ±10% of the prescribed dose, while only 57% of the skin area is within tolerance for the patient high body mass. The simulation results in the Stanford technique matched the observations in the study of the TSET Cherenkov dosimetry and physicists might be aware of the potential for low dose regions the dose distribution on the lateral sides of the patient, as these regions were in the boundaries of the different positions in the Stanford technique.

The TSET treatment planning study explicitly compared the Stanford and rotational techniques. Based upon the 3D visualization and the statistical analysis of the dose distribution, it was found that the TSET patient had more uniform dose distribution

over the body surface in rotational technique, especially in the lateral sides of the body. Around 90% of skin regions of the patients with both high and low body mass, had dose within ±10% of the prescribed dose in rotational technique. However, the body mass could affect the dose uniformity in Stanford techniques. The patient with high body mass had more abnormal regions of dose distribution, in which only 65% of the skin had the dose value within ±10% of the prescribed dose, compared with 85% of the skin for low body mass patient. Overall, the rotational technique should have better uniformity in dose distribution than the Stanford technique. This remains to be confirmed in patient studies.

The angle-dose relationship of the TSET treatment planning system was based upon the Monte Carlo dose simulation in the water cylindrical phantoms of different radii. However, the curvature radius the body were labelled manually for different parts. This hand-labelling process may contribute to dose simulation errors, as the angle-dose conversion varied for different curvatures, although this was not thought to alter the conclusions of the study. Still, future studies may incorporate the automatic calculation of curvature radius in the patient's 3D model to reduce the dose simulation errors.

The studies of the TSET treatment planning and Cherenkov dosimetry provided a proposed workflow to verify the dose distribution of TSET before and after the treatment. The abnormal regions of the dose distributions were observed in both treatment planning and Cherenkov dosimetry studies, which could potentially help physicists and dosimetrists to optimize the positions designed in all electron skin delivery techniques.

Appendix 1: Summary of TSET patients in the study of TSET dosimetry using Cherenkov imaging and computer animation at UPenn

The workflow using Cherenkov imaging and animation techniques was conducted to perform TSET dosimetry for the patients in University of Pennsylvania Health System (UPHS). More than 20 patients were involved in the study with the Cherenkov imaging and 3D body surface captured during the TSET treatment. The TSET dosimetry analysis was conducted for 6 patients here, choosing those with good quality Cherenkov images and good 3D body surfaces extracted during the treatment. The TSET dosimetry paper [123] listed two patients with different body mass (patient 13 and 21). This appendix section provides the TSET dosimetry results of these patients.

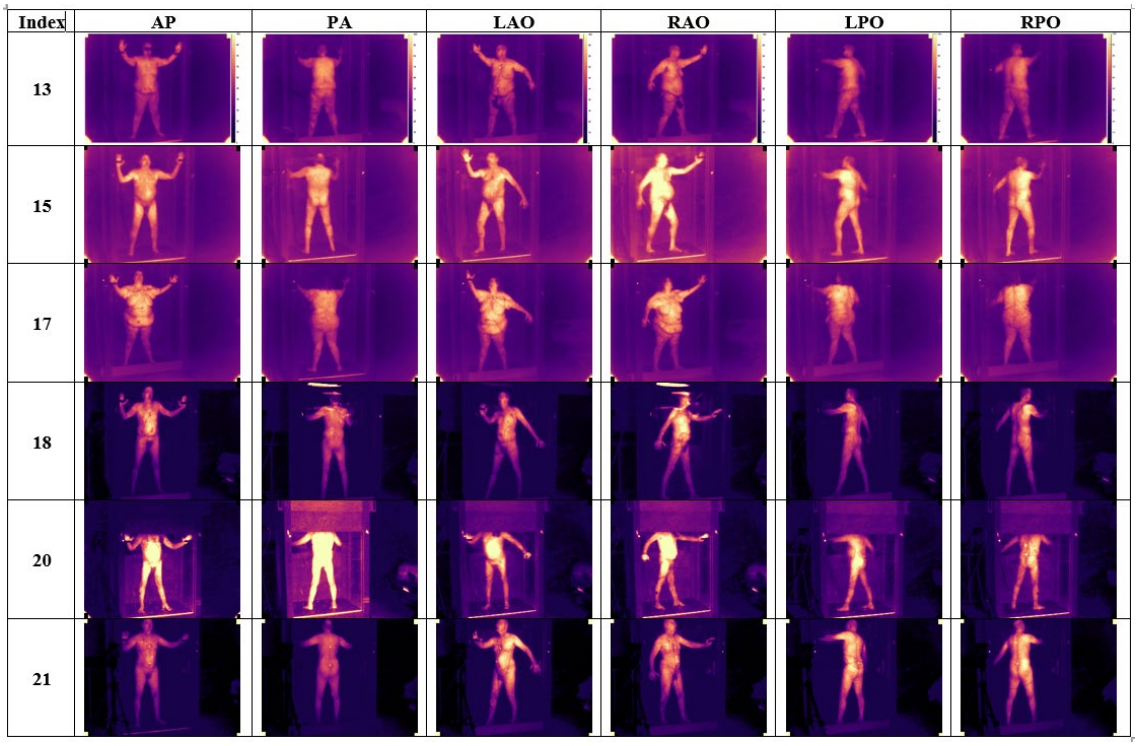


Figure A0.1: Summary of the dose distribution of the patients in UPHS derived from the Cherenkov imaging

A quantitative tabulation was completed for these patients to compare the uniformity of dose distribution extracted from Cherenkov imaging. The dose area histogram was plotted for all patients in Figure A0.2. The dashed curves represented the patients with high body mass, while the solid curves represented the patients with low body mass. This preliminary summary will be further analyzed for areas of the body that can be interpreted as true dose in a future study.

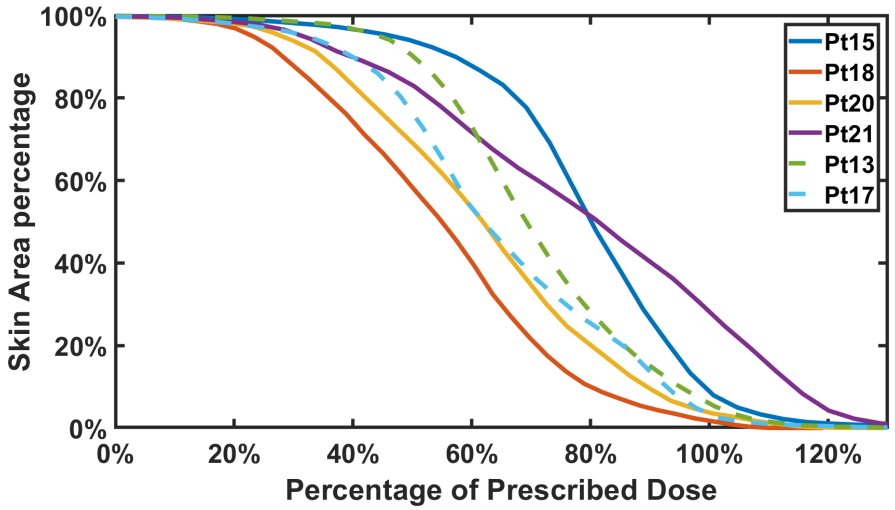


Figure A0.2: The comparison of dose area histogram (DAH) of TSET patients in UPHS Patient 18 and 20 had low body mass. However, their body parts were either blocked by the TSET stand or the shielding plates. So, they have more under-dosed skin regions. Patient 13 and 17 had large body mass, and the occlusion of their own body contributed to the low dose regions. Patient 15 and 20 had medium body mass and had generally better uniformity performance than other patients.

A1.1 TSET dosimetry of patient 13

Patient 13 was the patient with high body mass, that was analyzed in the TSET dosimetry study.[123]

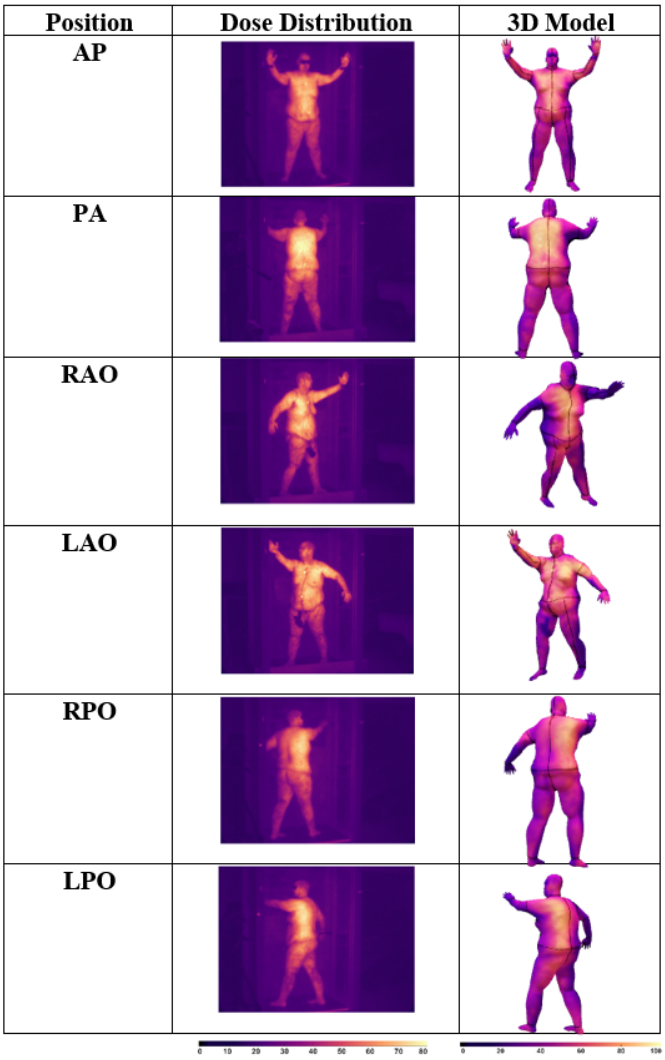


Figure A1.1: The dose distribution in 2D image and 3D models of Patient 13 in different position of Stanford techniques.

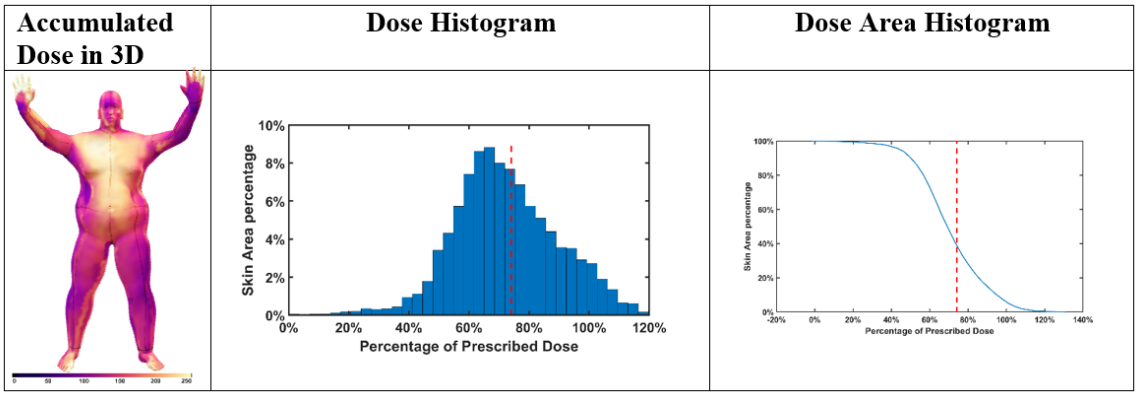


Figure A1.4: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

Patient 13 had prescribed dose of 198.4 cGy in the umbilicus region. The dose coverage of most skin region was less than the prescribed dose due to occlusion of the Cherenkov camera angle.

A1.2 TSET dosimetry of patient 15

The dose distribution was derived from the Cherenkov imaging through the calibration ratio based on the dose reading in the umbilicus region.[22] The 2D dose distribution image were then projected to the 3D models of the patient in each position of Stanford technique.

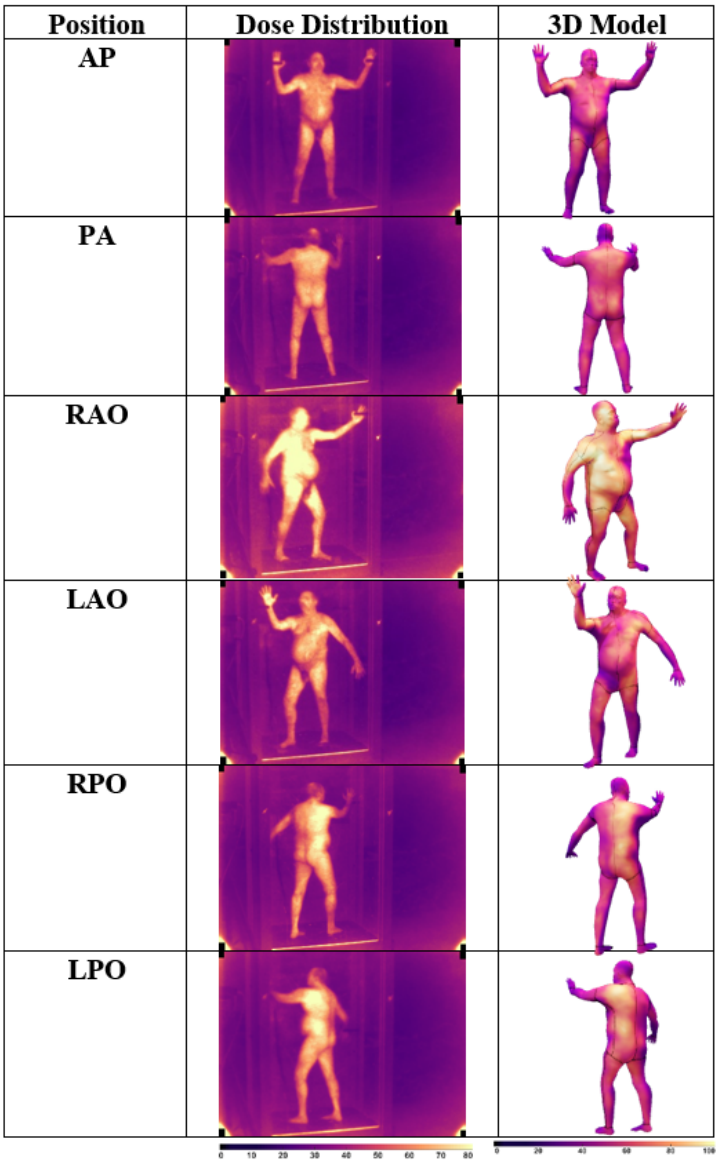


Figure A1.3: The dose distribution in 2D image and 3D models of Patient 15 in different position of Stanford techniques.

The accumulated dose distribution was displayed in the 3D model of the patients and the statistical analysis, including the area dose histogram and dose area histogram, was performed to analyze the uniformity of the TSET dose distribution.

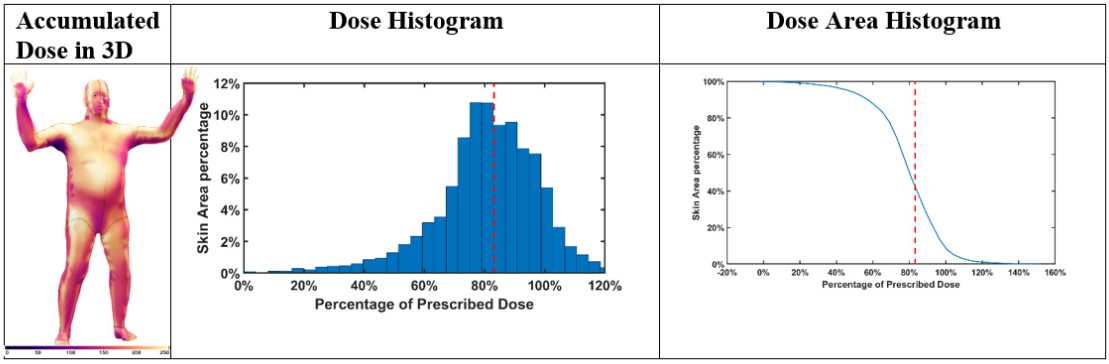


Figure A1.4: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

The prescribed dose for Patient 15 was 205.8cGy, based on the radiation diode reading in the umbilicus region of the patient. From the 3D display and the statistical analysis, most regions had dose coverage less than the prescribed dose. Some regions had an observed significantly lower dose estimation of coverage due to the occlusion of the Cherenkov camera, such as the right side of the belly regions. However, this was an artifact of the imaging, not true dosimetry. Like in the treatment dose planning study, the fluctuation was observed from the lateral side of the patients. However, the histogram had more weight in the lower dose regions, which was derived from the occlusion of the Cherenkov imaging.

A1.3 TSET dosimetry of patient 17

The patient 17 had large body mass and long hair. Some body parts, such as the head and skin fold, were occluded in the Cherenkov camera view.

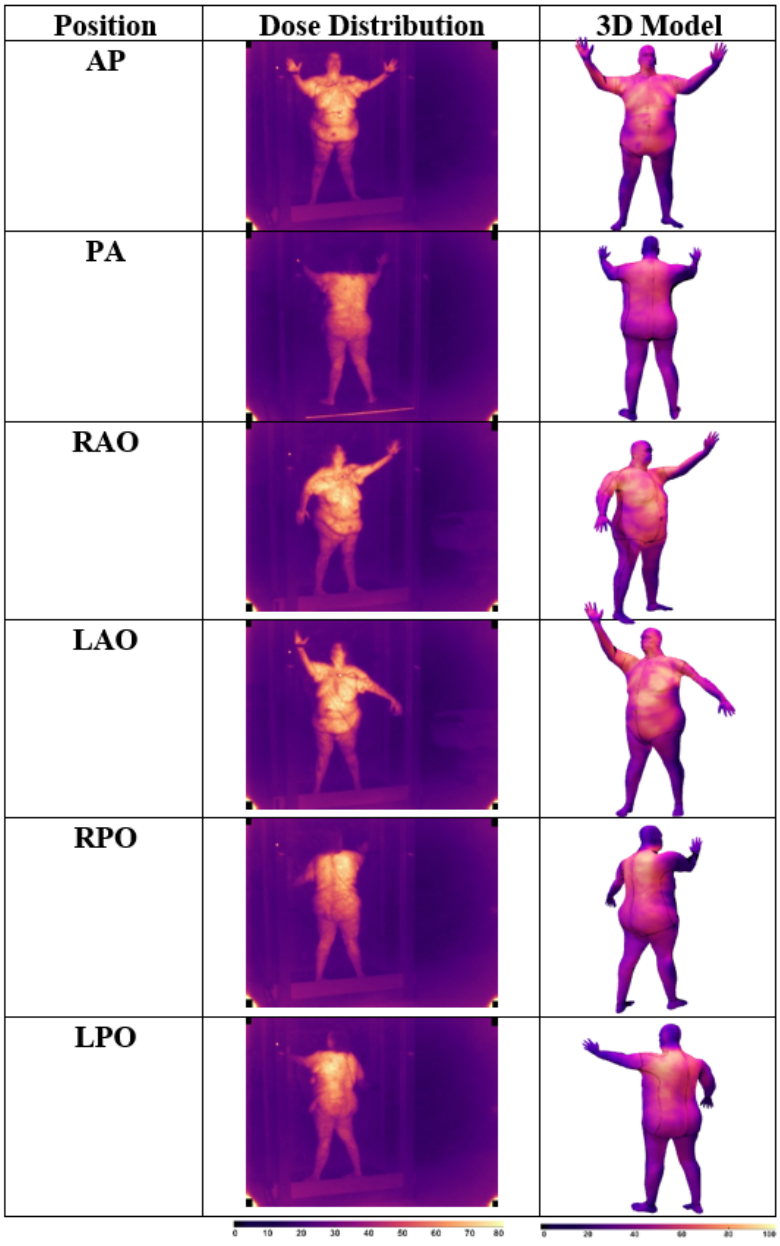


Figure A1.5: The dose distribution in 2D image and 3D models of Patient 17 in different position of Stanford techniques.

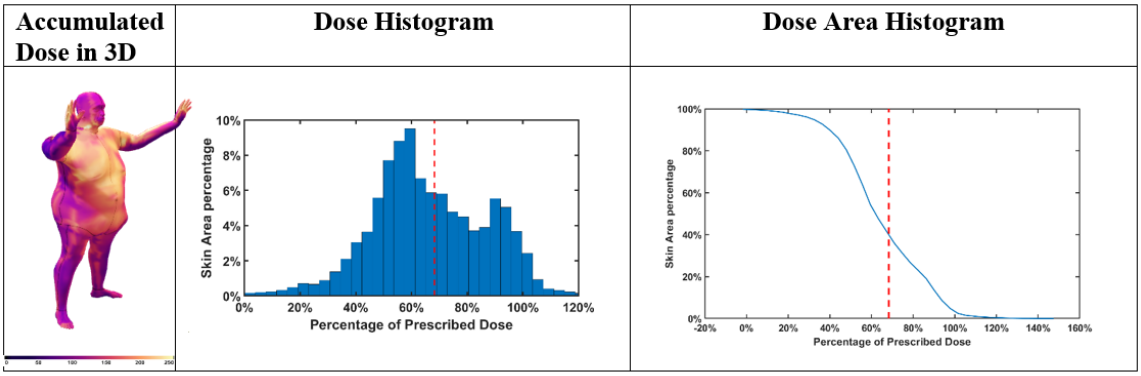


Figure A1.6: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

Patient 17 had a prescribed dose of 197.3cGy in the umbilicus region. Compared with other patients with low body bass, Patient 17 had more skin area with low dose, which can be attributed to the occlusion by their body shape as viewed by the camera.

A1.4 TSET dosimetry of patient 18

Patient 18 had low body mass. However, due to the positioning of the Cherenkov camera and the patient inside the TSET stand, some body parts were occluded by the TSET stand or the patient's body itself. For example, the patient's arms in PA and RPO positions are occluded by the TSET stand. There was also noise of light signal from the ceiling, which may adversely affect the dose reading.

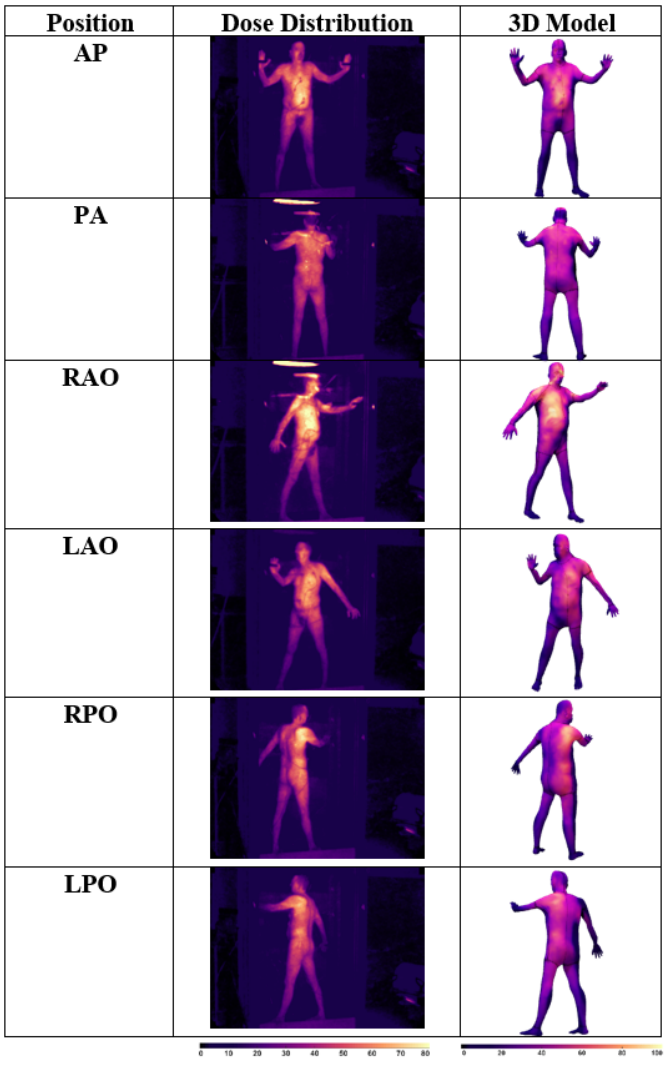


Figure A1.7: The dose distribution in 2D image and 3D models of Patient 18 in different position of Stanford techniques.

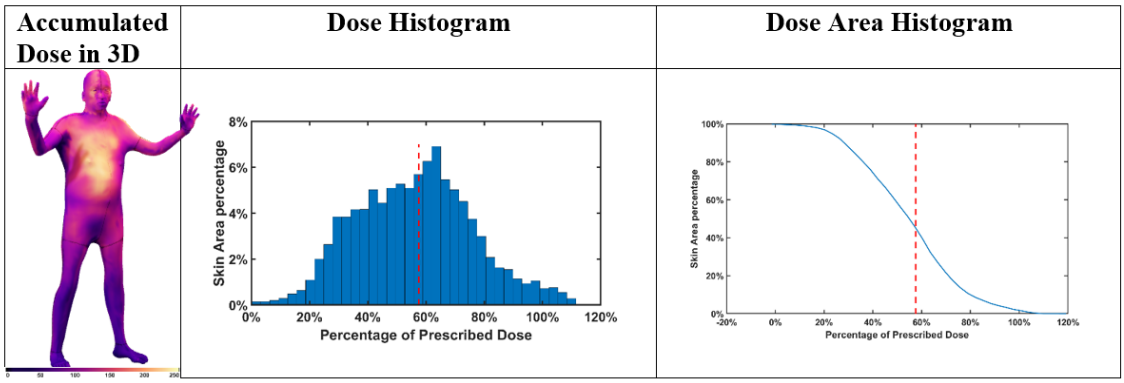


Figure A1.8: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

Patient 18 had a prescribed dose of 185.2 cGy in the umbilicus region. Like Patient 15, Patient 18 had large skin area with low dose due to the occlusion of the Cherenkov camera and the TSET stand.

A1.5 TSET dosimetry of patient 20

Patient 20 had a special treatment in which the head was shielded with a slab of lead plates. The other parts of the body also received higher dose than other patients.

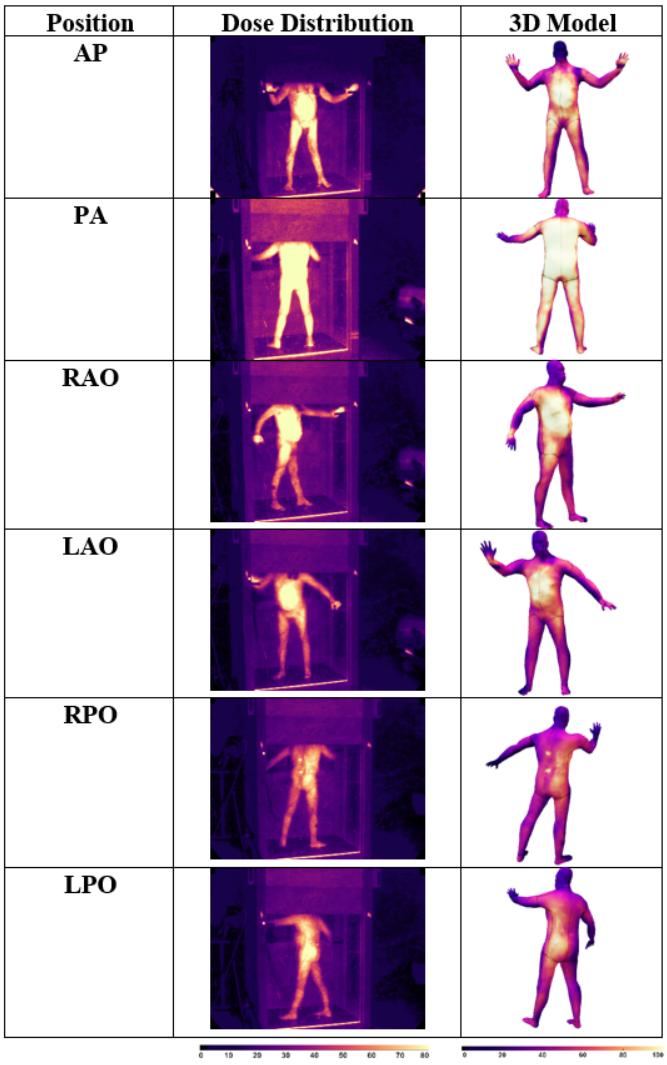


Figure A1.7: The dose distribution in 2D image and 3D models of Patient 20 in the different position of Stanford techniques.

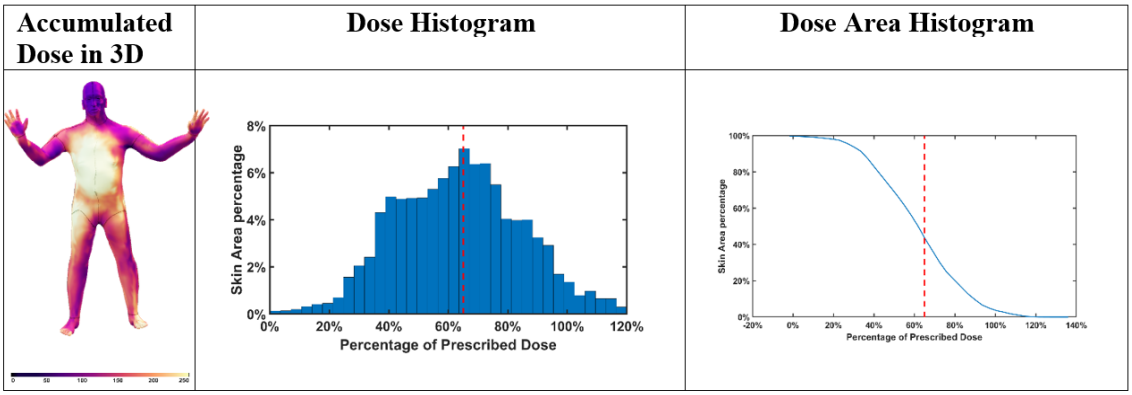


Figure A1.8: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

Patient 20 had a prescribed dose of 251.9 cGy in the umbilicus region. However, the skin area in the head was also included in the statistical analysis, which resulted in more low-dose skin regions.

A1.6 TSET dosimetry of patient 21

Patient 21 was the patient with low body mass in the study of TSET dosimetry.[123]

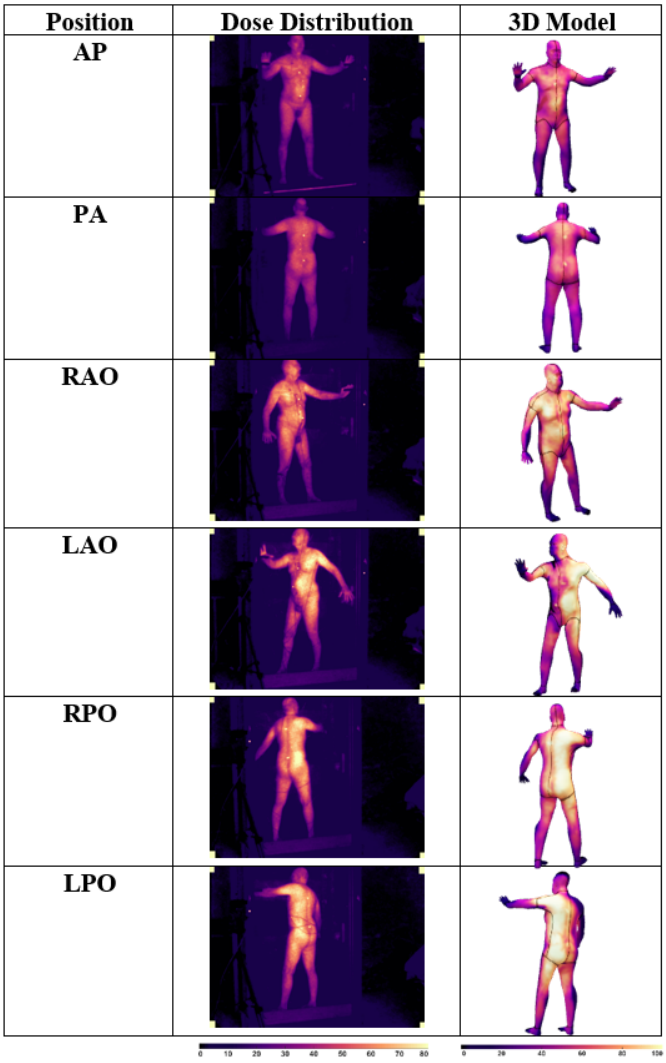


Figure A1.9: The dose distribution in 2D image and 3D models of Patient 21 in different position of Stanford techniques.

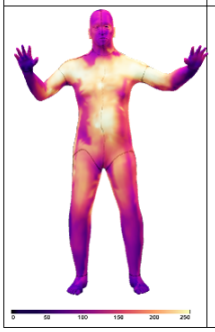
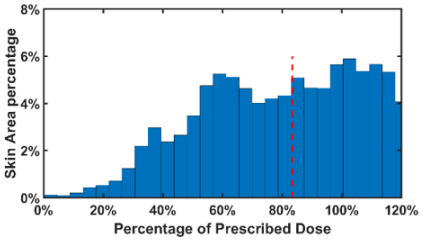
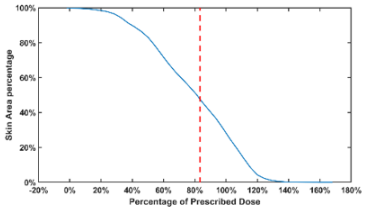
Accumulated Dose in 3D	Dose Histogram	Dose Area Histogram
		

Figure A1.10: The accumulated dose distribution in the 3D model of the patient, the dose histogram and the DAH plot of the dose distribution.

Patient 21 had a prescribed dose of 186.9 cGy in the umbilicus region. Camera angle occlusion was observed in PA, RPO and LPO positions.

Appendix 2: The computer programming scripts in radiation beam shape extraction

The code scripts are programmed for the work in the following publication.

Miao T, Bruza P, Pogue BW, et al. Cherenkov imaging for Linac beam shape analysis as a remote electronic quality assessment verification tool. *Med Phys.* 2019;46(2):811-821.

A2.1 Extraction of corners of the checkerboard image

The corner extraction script is writing in C++, with the support of OpenCV library. It reads an image containing checkerboard pattern in iso-plane, and extracts the pixel location of these corners. The pixel locations are saved in the text file called "location.txt". The cmake text file adds the OpenCV library into the scripts.

```
/*
        File name: main.cpp
        Author: Tianshun Miao
*/
#include <iostream>
#include <fstream>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace cv;
using namespace std;

int main() {
        Size patternsize(5, 6); //number of centers
        Mat frame = imread("Checker_board.png"); //source image with the chess board pattern
        vector<Point2f> centers; //this will be filled by the detected centers

        bool patternfound = findChessboardCorners(frame, patternsize, centers);

        cout << patternfound << endl;
        //drawChessboardCorners(frame, patternsize, Mat(centers), patternfound);
```

```
        ofstream myfile;
        // output the corner information into the .txt files
        myfile.open("location.txt");
        // iterate through every corners
        for (int i = 0; i < centers.size(); i++)
        {
                myfile << centers[i].x << " " << centers[i].y << endl;
        }
        myfile.close();
        return 0;
}
```

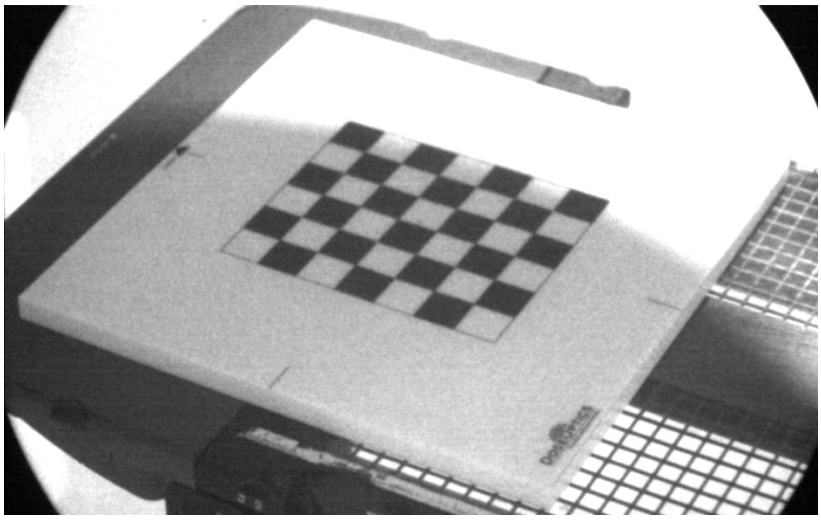


Figure A2.1: The checkerboard image used for camera calibration.

```
# CMakeLists.txt
PROJECT(Find_corner)

# Specify the Opencv Library location
set(OpenCV_DIR "D:/opencv/build/")

# Find and include OpenCV library
find_package(OpenCV REQUIRED)
include_directories(${OpenCV_INCLUDE_DIRS})

# Collect and include all of our source files (.h and .cxx files)
include_directories(${CMAKE_CURRENT_SOURCE_DIR}
${CMAKE_CURRENT_BINARY_DIR})
file(GLOB CXX_FILES *.cpp)

# We are creating an executable project, and including the source files into it
add_executable(Find_corner ${CXX_FILES})

# Set the startup project in Visual Studio to be our project (rather than the extra stuff
CMake builds)
```

```
set_property(DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR} PROPERTY
VS_STARTUP_PROJECT Find_corner)

# Tell the project which libraries to link
target_link_libraries(Find_corner ${OpenCV_LIBS})
```

A2.2 Extraction the camera projective matrix

This Matlab script reads the pixel locations of the corners in the checkerboard images. The real locations of the corners are coded in the script directly. The camera projective matrix is calculated using the camera calibration library in Matlab and is saved in the output file of "tf_matrix.mat".

```
% File name: transformation_extraction.m
% Author: Tianshun Miao
clc
clear all
close all
% Load the Checkerboard images
checkboard_name = 'Checker_board.png';
% Load the pixel locations
location_file = 'location.txt';
im_checker = imread(checkboard_name);
locs = dlmread(location_file);
% Specify the physical locations of the corners
x_vect = 2.31*[1:-1:-3] * 100;
y_vect = 2.31*[3:-1:-3] * 100;
% Build up the matrix of the of the x and y coordinates of the physical
% locations
[x_mat, y_mat] = meshgrid(x_vect, y_vect);
x_mat_t = x_mat';
y_mat_t = y_mat';
% flatten the matrix of the physical locations to the vectors
loc_real = [x_mat_t(:), y_mat_t(:)];
% Use matlab function to get the camera prjective matrix
tform = fitgeotrans(locs, loc_real,'projective');
% Check if the pixel location of the corners are right
figure(1)
imshow(im_checker);
hold on;
line(locs(:, 1), locs(:, 2));
hold off;
```

```
figure(2)
% Check if the transfomration is by display the image after transformation
It = imwarp(im_checker, tform);
imshow(It)
save('tf_matrix.mat', 'tform');
```

A2.3 Extraction of the 2D Cherenkov distribution

The Matlab code in this section converted the Cherenkov image to the 2D Cherenkov distribution based on the camera projective camera matrix. The script displayed 2D Cherenkov distribution with the colormap of the "ice style".[124]

```
% File Name: get_2d_profile.m
% Author: Tianshun Miao
% The colormap function uses the function cmocean, which is extracted at
https://www.mathworks.com/matlabcentral/fileexchange/57773-cmocean-perceptually-uniform-colormaps

clc
clear all
close all
% Read the Cherenkov image and the background image
che_image_name = 'new_images/abs_che/abs_10_ig_70k.PNG';
bkg_image_name = 'new_images/abs_che/abs_bkg_ig_70k.PNG';
% Input the beam size
beam_size = 15;
min_val = 40;
% Read the transfomation matrix
load('tf_matrix.mat');
im_che = imread(che_image_name);
im_bkg = imread(bkg_image_name);
% Get the Cherenkov image without background
im_che = abs(double(im_che) - double(im_bkg));
% Define the direction, v stands for vertical, h stands for horizontla
dire = 'v';
% get the line to profile
% Get the 2d matrix
x_vect = -1500:1:1500;
y_vect = -1500:1:1500;
[x_mat, y_mat] = meshgrid(x_vect, y_vect);
% Get the pixel locations of the points
[x_proj_vect, y_proj_vect] = transformPointsInverse(tform, x_mat(:), y_mat(:));
row_num = size(x_mat, 1);
col_num = size(x_mat, 2);
x_proj_mat = reshape(x_proj_vect, row_num, []);
y_proj_mat = reshape(y_proj_vect, row_num, []);
figure(1)
imshow(im_che, [0, 4000])
hold on;
scatter(x_proj_mat(:), y_proj_mat(:),  'LineWidth', 2)
```

```
% line([p_A_prime(1) p_B_prime(1)], [p_A_prime(2) p_B_prime(2)], 'LineWidth', 2)
% line([p_C_prime(1) p_D_prime(1)], [p_C_prime(2) p_D_prime(2)], 'LineWidth', 2)
hold off;
% Extract the 2D Cherenkov distribution
pixel_mat = interp2(double(im_che), x_proj_mat, y_proj_mat);

% Display the 2D Cherenkov distribution
figure(3)
imagesc(pixel_mat);
cmocean('ice', 2000);
hold on;
line([0, 3000],[1500 1500], 'Color', 'r', 'LineStyle', '--', 'LineWidth', 4)
hold off;
xticklabels = -15:3:15;
xticks = linspace(1, size(pixel_mat, 2), numel(xticklabels));
set(gca, 'XTick', xticks, 'XTickLabel', xticklabels)
yticklabels = -15:3:15;
yticks = linspace(1, size(pixel_mat, 1), numel(yticklabels));
set(gca, 'YTick', yticks, 'YTickLabel', flipud(yticklabels(:)))
set(gca,'FontSize',20)
set(gca,'FontWeight','bold')
set(gca, 'LineWidth', 4);
xlabel('Distance in Horizontal Direction / cm');
ylabel('Distance in Vertical Direction / cm');
axis([0 3000 0 3000]);
axis equal
```

A2.4 Extraction of the size of the Cherenkov pattern

This script first extracted the 2D Cherenkov distribution from the Cherenkov image.

Then, the Cherenkov distribution across the horizontal and vertical line was extracted.

The width of the profile was defined as the full width half maximum or maximum

derivatives.

```
% File Name: get_line_profile.m
% Author: Tianshun Miao
clc
clear all
close all
% Read the Cherenkov image and the background images
che_image_name = 'new_images/abs_che/abs_10_ig_70k.PNG';
bkg_image_name = 'new_images/abs_che/abs_bkg_ig_70k.PNG';
% The jaw size of the linac
beam_size = 10;
% background pixel reading
min_val = 15;
load('tf_matrix.mat');
im_che = imread(che_image_name);
im_bkg = imread(bkg_image_name);
im_che = abs(double(im_che) - double(im_bkg));
% Direction of the line profile, v stands for vertical and h stands for
% horizontal
dire = 'v';
% get the line to profile, input the start and end point of the line
% profile
p_a = [0, 2500];
p_b = [0, -2500];
p_c = [-2500, 0];
p_d = [2500, 0];
if(dire == 'h')
   % Create the regions to extract the line profile
    [x_mat, y_mat] = create_stripe_mat2(p_c, p_d, 30);
else
     [x_mat, y_mat] = create_stripe_mat2(p_a, p_b, 30);
end
% Get the regions of the pixel locations of the line profile
[x_proj_vect, y_proj_vect] = transformPointsInverse(tform, x_mat(:), y_mat(:));
row_num = size(x_mat, 1);
col_num = size(x_mat, 2);
x_proj_mat = reshape(x_proj_vect, row_num, []);
```

```
y_proj_mat = reshape(y_proj_vect, row_num, []);
% Display the line profile region in the Cherenkov plot
figure(1)
imshow(im_che, [0, 4000])
hold on;
scatter(x_proj_mat(:), y_proj_mat(:),  'LineWidth', 2)
line(x_proj_mat(:, floor(col_num/2)), y_proj_mat(:, floor(col_num/2)), 'Color', 'r',
'LineWidth', 3)
% line([p_A_prime(1) p_B_prime(1)], [p_A_prime(2) p_B_prime(2)], 'LineWidth', 2)
% line([p_C_prime(1) p_D_prime(1)], [p_C_prime(2) p_D_prime(2)], 'LineWidth', 2)
hold off;
pixel_mat = interp2(double(im_che), x_proj_mat, y_proj_mat);
% if(dire == 'h')
%     line_profile2 = median(pixel_mat, 1);
% else
%     line_profile2 = median(pixel_mat, 2);
% end
% Extraction of the line profile by take the mean of the strip
 line_profile2 = mean(pixel_mat, 1);
% Median filter the line profile
% Apply median filter to smooth the line profile
line_profile = medfilt1(line_profile2, 17);

windowSize = 10;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
line_profile = filter(b, a, line_profile);
line_length = length(line_profile);
line_leng = 1:line_length;
line_leng = (line_leng - line_length/2)/100;
% Extract the index of the peak values
peak_ind = [2400:2450, 2550:2600];
peak_val = mean(line_profile(peak_ind));
ratio = 0.5;
% Calute the width through fwhm
[fwhm, ind1, ind2]= calc_width(line_profile, (peak_val+min_val)*ratio);

% calcuate the width through maximum derivative.
% diff_profile = diff(line_profile);
% [~, ind1] = min(diff_profile);
% [~, ind2] = max(diff_profile);
% fwhm = abs(ind1 - ind2) / 100;
output_info = strcat('The full width half maximum is ', num2str(fwhm), ' cm');
disp(output_info)
% Display the line profile
figure(2)
```

```
% line_leng(1) = [];
plot(line_leng, line_profile, 'LineWidth', 2);
xlabel('Distance / cm');
ylabel('Cherenkov Intensity');
set(gca, 'FontSize', 20, 'FontWeight', 'bold')
axis([-15 15 0 2000]);
% title('Plot of Derivative/Difference of Pixel Count along Horizontal Direction (10cm x
10cm Cherenkov)', 'FontSize', 20);
hold on;
% vline([-beam_size/2, beam_size/2], 'g');
vline([line_leng(ind1), line_leng(ind2)], 'r', ['Boundary']);
hline([peak_val], 'b', 'Maximum Value');
hline([min_val], 'b', 'Minimum Value');
hline([(peak_val+min_val)*ratio], 'b', 'Half Maximum Value');
hold off;
```

A few functions have been used in extraction the size of the Cherenkov patterns.

```
% File Name: create_stripe_mat2.m
% Author: Tianshun Miao
function [ x_mat, y_mat ] = create_stripe_mat2( p1, p2, d_w )
%CREATE_STRIPE_MAT2 Summary of this function goes here
% This function is to extract the read locations of the strip region
% between two points, defined as p1 and p2. The wdth of the strip is
% defined as d_w. The output are two matrix represents the x and y
% coodinates of the points in the strip.
%   Detailed explanation goes here
nump = round(norm(p1 - p2));
line_vect = (p1 - p2) / norm(p1 - p2);
norm_line_vect = [line_vect(2) -line_vect(1)];
x_vect = linspace(p1(1), p2(1), nump);
y_vect = linspace(p1(2), p2(2), nump);
x_mat1 = repmat(x_vect, 2*d_w+1, 1);
y_mat1 = repmat(y_vect, 2*d_w+1, 1);
rang = -d_w:d_w;
leng = length(x_vect);
cal_x_mat = repmat(rang', 1, leng) * norm_line_vect(1);
cal_y_mat = repmat(rang', 1, leng) * norm_line_vect(2);
x_mat = x_mat1 + cal_x_mat;
y_mat = y_mat1 + cal_y_mat;

end
```

The following function calculate the width of a Cherenkov line profile.

```
% File Name: calc_width.m
```

```
% Author: Tianshun Miao
function [ wid_val, ind1, ind2 ] = calc_width( line_vect, y_val)
%CALC_WIDTH Summary of this function goes here
% This function calculate the width of the linear curve. The threshold is
% defined as y_val. The function itlp is a solver to calcuate the closeset
% the index for which line_vect value cloeset to y_val
%   Detailed explanation goes here
leng = length(line_vect);
ind1 = itpl(line_vect(1:floor(leng/2)), y_val);
ind2 = itpl(line_vect(floor(leng/2)+1:end), y_val);
ind2 = floor(leng/2) + ind2;
wid_val = abs(ind2 - ind1) / 100;
end

% File Name: itpl.m
% Author: Tianshun Miao

function [ ind ] = itpl( y_vect, x )
%ITPL Summary of this function goes here
% This is a solver with y_vect = x
%   Detailed explanation goes here
abs_y = abs(y_vect - x);
[~, ind] = min(abs_y);
end
```

A2.5 Cherenkov distribution with complex shapes configured by MLC

The Matlab scripts displayed the patterns of the Cherenkov distribution of a dynamic MLC treatment plan. It also read the dynalog files, which has the information of the MLC leaf positions.

The first Matlab scripts read the Cherenkov images and converted them into Cherenkov distribution in real physical space using the transformation matrix extracted through the checkerboard pattern.

```
% File Name: processing_image.m
% Author: Tianshun Miao
clc
clear all
close all
% Read the Cherenkov image files, which is saved as .mat file
image_file_name = 'complex_shape_data.mat';
load(image_file_name);
% Read the prjective transformation matrix
load('tf_matrix.mat');
% Prelocate the Cherenkov distribution, which as 60 frames
im_che = complex_shape_data(:, :, 60);
% The matrix of x and y coordinates
x_vect = -1000:1:1000;
y_vect = -1000:1:1000;
[x_mat, y_mat] = meshgrid(x_vect, y_vect);
% Get the pixel locations of the real physical location points
[x_proj_vect, y_proj_vect] = transformPointsInverse(tform, x_mat(:), y_mat(:));
row_num = size(x_mat, 1);
col_num = size(x_mat, 2);
x_proj_mat = reshape(x_proj_vect, row_num, []);
y_proj_mat = reshape(y_proj_vect, row_num, []);
% Convert the Cherenkov image to Cherenkov distribution for every frame and
% saved the frame as images
for i = 1:size(complex_shape_data, 3)
    i
    im_che = complex_shape_data(:, :, i);
    pixel_mat = uint16(interp2(double(im_che), x_proj_mat, y_proj_mat));
    file_name = strcat('processed_images/', num2str(i), '.PNG');
    imwrite(pixel_mat, file_name);
end
```

The second part of the Matlab scripts read the dynalog file and create images of the MLC patterns. This was a test script to verify the dynalog input functionality.

```
% File Name: read_dynalog.m
% Author: Tianshun Miao
clc
clear all
close all
% Read the dynalog file
dynalog_file_name1 = 'dynalogs_20180424/A20180424174955_032210.dlg';
dynalog_file_name2 = 'dynalogs_20180424/B20180424174955_032210.dlg';
dyna_data1 = dlmread(dynalog_file_name1, ',', 6, 0);
dyna_data2 = dlmread(dynalog_file_name2, ',', 6, 0);
% Output image folder
folder_name = 'before_jaw/';
tot_leaf = 60;
time_size = size(dyna_data1, 1);
% The time step
h = figure;
set(h, 'Visible', 'off');
% For each frame get the mlc patterns
for i = 1:time_size
i
   for j = 0:tot_leaf-1 % j is leaf index
      x_pos1 = -dyna_data1(i, 4*j + 16) / 100 * 1.96078;
      x_pos2 = -dyna_data2(i, 4*j + 16) / 100 * 1.96078;
      y_pos = leaf_pos(j+1) * 10;
      drawrectangle([], -100, y_pos(1), x_pos2+100, y_pos(2) - y_pos(1));
      drawrectangle([], 100, y_pos(1), -(x_pos1+100), y_pos(2) - y_pos(1));
   end
axis equal
xlim([-100 100]);
ylim([-100 100]);
file_name = strcat(folder_name, num2str(i), '.png');
saveas(gca, file_name);
clf
end
```

The function of leaf_pos read the index of the leaf in the dyanlog file and gave the location of the leaf in vertical direction.

```
function [y_loc] = leaf_pos(ind)
```

```
%LEAF_POS Summary of this function goes here
% y_loc records the position of given mlc leaf in the vertical direction
%   Detailed explanation goes here
y_loc = zeros(1, 2);
if(ind == 1)
    y_loc(1) = 0;
    y_loc(2) = 1.4;
elseif(ind <= 10)
    y_loc(1) = 1.4 + (ind - 2) * 1;
    y_loc(2) = 1.4 + (ind - 1) * 1;
elseif(ind <= 50)
    y_loc(1) = 10.4 + (ind - 12) * 0.5;
    y_loc(2) = 10.4 + (ind - 11) * 0.5;
elseif(ind <= 60)
    y_loc(1) = 30.4 + (ind - 52) * 1;
    y_loc(2) = 30.4 + (ind - 51) * 1;
else
    y_loc(1) = 39.4;
    y_loc(2) = 40.8;
end
y_loc(1) = y_loc(1) - 20.4;
y_loc(2) = y_loc(2) - 20.4;
end

function [] = drawrectangle( I, x, y, width, height )
%DRAWRECTANGLE Summary of this function goes here
% The function displays the a red rectangle box in the image specified by
% I. The position of the box was the decided by the parameters x and y (The
% top-left side of the box). The size of the box is specified by the
% parameters width and height.
%   Detailed explanation goes here

% Determine the number of rectangle we need to draw
box_num = size(x, 1);

% Tell if there are mutiple rectangles I need to draw for the image
if(isscalar(width))
    width = width * ones(box_num, 1);
    height = height * ones(box_num, 1);
end

% Input the properties of the retangle
% According to homework handout: Color 'r', linestyle '-' and linewidth 0.5
prop1 = 'color';
% prop1_value = [0, 0, 0];
prop1_value = 'r';
```

```
prop2 = 'linestyle';
prop2_value = '-';
prop3 = 'linewidth';
prop3_value = 1.5;

% Step 2: Display the image;
if(~isempty(I))
imshow(I);
end
% Step 3: Draw a overload rectangle
hold on;
for i = 1:box_num
   line([x(i) (x(i) + width(i))], [y(i) y(i)], prop1, prop1_value, ...
      prop2, prop2_value, prop3, prop3_value);  % First draw line up
   line([x(i) (x(i) + width(i))], [(y(i)+height(i)) ...
      (y(i)+height(i))], prop1, prop1_value, ...
      prop2, prop2_value, prop3, prop3_value); % second draw line dow
   line([(x(i)+width(i)) (x(i) + width(i))], [y(i) ...
      y(i)+height(i)], prop1, prop1_value, ...
      prop2, prop2_value, prop3, prop3_value); % third draw right line
   line([x(i) x(i)], [y(i) y(i)+height(i)], prop1, prop1_value, ...
      prop2, prop2_value, prop3, prop3_value); % last drow line
end
hold off;

end
```

In the third part of the Matlab scripts, the Cherenkov distribution and the MLC patterns were overlaid in the same frames, with the frames converted to the movie file.

```
% File Name:draw_leaf.m
% Author: Tianshun Miao
clc
clear all
close all
%% Read folder names
che_folder = 'processed_vid_file/';
che_file_num = 380;
che_time_stamp_file = 'che_raw_video/time_s1.txt';
% dynalog file name
time_step = 50e-3;
dynalog_file_name1 = 'dynalogs_20180424/A20180424174955_032210.dlg';
dynalog_file_name2 = 'dynalogs_20180424/B20180424174955_032210.dlg';
dyna_data1 = dlmread(dynalog_file_name1, ',', 6, 0);
dyna_data2 = dlmread(dynalog_file_name2, ',', 6, 0);
tot_leaf = 60;
```

```
%% Read Cherenkov file
che_time_file = dlmread(che_time_stamp_file);
s1_t = che_time_file(:, 2);
t0 = s1_t(1);
for i = 1:che_file_num
    i
    che_file_name = strcat(che_folder, num2str(i), '.PNG');
    che_img = flipud(imread(che_file_name));
    t1 = s1_t(i) - t0;
    dyna_t = round(t1 / time_step) + 4;
    f = figure('visible','off');
    imshow(che_img)
    set(gca,'position',[0 0 1 1],'units','normalized')
    hold on;
    for j = 0:tot_leaf-1 % j is leaf index
        x_pos1 = -dyna_data1(dyna_t, 4*j + 16) / 100 * 1.96078 * 10;
        x_pos2 = -dyna_data2(dyna_t, 4*j + 16) / 100 * 1.96078 * 10;
        y_pos = (leaf_pos(j+1) + 0.5) * 10 * 10;
        drawrectangle([], 0, y_pos(1) + 1000, x_pos2+1000, y_pos(2) - y_pos(1));
        drawrectangle([], 2000, y_pos(1) + 1000, -(x_pos1+1000), y_pos(2) - y_pos(1));
    end
    hold off;
    frame_data = getframe(gca);
    save_file_name = strcat('vid_leaf/', num2str(i), '.png');
    imwrite(frame_data.cdata, save_file_name);
end

% File Name:vid_creation.m
% Author: Tianshun Miao
clc
clear all
close all
map = cmocean('ice', 400);
v = VideoWriter('processed_shape.mp4', 'MPEG-4');
v.FrameRate = 10;
open(v)
for i = 1:380
    i
    image_name = strcat('vid_leaf/', num2str(i), '.PNG');
    im1 = imread(image_name);
    writeVideo(v,im1);
end
close(v)
```

A2.6 Star shot analysis based on Cherenkov imaging and film measurement

The first part of the Matlab scripts conducted the start shot analysis of the Cherenkov images. The scripts read the star shot pattern from individual images and converted to the Cherenkov distribution in physical space. Then the scripts extract the center lines of the Cherenkov strips and performed star shot analysis on the intersections of the lines.

```
% File Name:create_shot.m
% Author: Tianshun Miao
clc
clear all
close all

% load shot from all angles
load('img30.mat');
load('img90.mat');
load('img150.mat');
load('img180.mat');
load('img240.mat');
load('img300.mat');
load('tf_matrix.mat');
% combine all the images
im_total = img30 + img90 + img150 + img180 + img240 + img300;
figure(1)
imagesc(im_total);
%  back projection
x_vect = -1500:1:1500;
y_vect = -1500:1:1500;
[x_mat, y_mat] = meshgrid(x_vect, y_vect);
[x_proj_vect, y_proj_vect] = transformPointsInverse(tform, x_mat(:), y_mat(:));
row_num = size(x_mat, 1);
col_num = size(x_mat, 2);
x_proj_mat = reshape(x_proj_vect, row_num, []);
y_proj_mat = reshape(y_proj_vect, row_num, []);
pixel_mat = interp2(double(im_total), x_proj_mat, y_proj_mat);
figure(2)
imagesc(pixel_mat)

% visualize
c_map = cmocean('ice', 500);
shot_img = ind2rgb(uint16(pixel_mat), c_map);
imshow(shot_img)
save('shot_cherenkov.mat', 'pixel_mat');
```

```
% File Name:shot_analysis_che.m
% Author: Tianshun Miao
clc
clear all
close all

load('shot_cherenkov.mat');
film_ratio = 1 / 100;  % the distance / pixel ratio of film reading
im_d = pixel_mat;
% get the location on the circle
radius = [800; 1000; 1200]; %just an example
cent = size(im_d) / 2; % center axis
theta = linspace(0,2*pi, 360);%you can increase this if this isn't enough yet
x=radius*cos(theta) + cent(2);
y=radius*sin(theta) + cent(1);
figure(1) % plot out the location of the circle
image(im_d);
hold on;
scatter(x(:), y(:));
hold off;
axis equal
figure(2) % plot out the profile in polar coordinte
angular_prof = interp2(im_d, x, y);
polarplot(theta, angular_prof);
% Start extract the line profile
w = gausswin(8);
angular_prof2 = angular_prof;
prof_ave = mean(angular_prof2, 2);
angular_prof_p = angular_prof2 - repmat(prof_ave, [1, size(angular_prof2, 2)]);
figure(3)
plot(theta, angular_prof_p);
rec_0 = zeros(3, 24);
rec_sign = zeros(3, 24); % 1 means increase, 0 means decrease
ind = ones(3, 1);
for i = 1:3
    for j = 2:size(theta, 2)
        if(angular_prof_p(i, j) > 0 && angular_prof_p(i, j - 1) < 0)
            rec_0(i, ind(i)) = (theta(j-1) + theta(j))/2;
            rec_sign(i, ind(i)) = 1;
            ind(i) = ind(i) + 1;
        elseif(angular_prof_p(i, j) < 0 && angular_prof_p(i, j - 1) > 0)
            rec_0(i, ind(i)) = (theta(j-1) + theta(j))/2;
            rec_sign(i, ind(i)) = 0;
            ind(i) = ind(i) + 1;
        end
    end
```

```
end
% get the cent point of the axis
if(rec_sign(1, 1) == 0)
    rec_0 = circshift(rec_0, 1, 2);
end
if(rec_0(1, 1) > rec_0(1, 2))
   rec_0(:, 1) = rec_0(:, 1) - 2*pi;
end
x_int=repmat(radius, [1, size(rec_0, 2)]).* cos(rec_0) + cent(2);
y_int=repmat(radius, [1, size(rec_0, 2)]).* sin(rec_0) + cent(1);
x_cent = (x_int(:, 1:2:end) + x_int(:, 2:2:end)) / 2;
y_cent = (y_int(:, 1:2:end) + y_int(:, 2:2:end)) / 2;
figure(4)
image(im_d)
hold on;
scatter(x_int(:), y_int(:));
scatter(x_cent(:), y_cent(:), 'r');
hold off;
% get the line profile
xcent = [x_cent(:, 1:6); x_cent(:, 7:12)];
ycent = [y_cent(:, 1:6); y_cent(:, 7:12)];
pfit = zeros(2, 6);
for i = 1:6
    pfit(:, i) = polyfit(xcent(:, i), ycent(:, i), 1)';
end
x_vect = 1000:0.1:2000;
vect_siz = size(x_vect, 2);
y_mat = zeros(6, vect_siz);
for i = 1:6
   y_mat(i, :) = polyval(pfit(:, i)', x_vect);
end
figure(5)
image(im_d)
hold on;
for i = 1:6
   plot(x_vect, y_mat(i, :), 'LineWidth', 2)
end
hold off;
axis equal
```

The second part of the scrips conducted the star shot analysis from the film measurement.

The center lines extracted in the film measurement were overlaid with the center lines in

the Cherenkov image.

```
% File Name:shot_analysis_film.m
% Author: Tianshun Miao
clc
clear all
close all
load('shot_cherenkov.mat');
shot_film = imread('img028.tif');
load('poly_data.mat');
film_ratio = 20.32 / 576;  % the distance / pixel ratio of film reading
im_s = fliplr(shot_film(225:576, 406:971));
im_r = double(im_s(:, :, 1));
im_d = polyval(p4, im_r);
% get the location on the circle
radius = [100; 125; 150]; %just an example
cent = size(im_d) / 2; % center axis
theta = linspace(0,2*pi, 500);%you can increase this if this isn't enough yet
x=radius*cos(theta) + cent(2);
y=radius*sin(theta) + cent(1);
figure(1) % plot out the location of the circle
image(im_d);
hold on;
scatter(x(:), y(:));
hold off;
axis equal
figure(2) % plot out the profile in polar coordinte
angular_prof = interp2(im_d, x, y);
polarplot(theta, angular_prof);

% Start extract the line profile
prof_ave = mean(angular_prof, 2);
angular_prof_p = angular_prof - repmat(prof_ave, [1, size(angular_prof, 2)]);
figure(3)
plot(theta, angular_prof_p);
rec_0 = zeros(3, 24);
rec_sign = zeros(3, 24); % 1 means increase, 0 means decrease
ind = ones(3, 1);
for i = 1:3
    for j = 2:size(theta, 2)
        if(angular_prof_p(i, j) > 0 && angular_prof_p(i, j - 1) < 0)
            rec_0(i, ind(i)) = (theta(j-1) + theta(j))/2;
            rec_sign(i, ind(i)) = 1;
            ind(i) = ind(i) + 1;
        elseif(angular_prof_p(i, j) < 0 && angular_prof_p(i, j - 1) > 0)
            rec_0(i, ind(i)) = (theta(j-1) + theta(j))/2;
            rec_sign(i, ind(i)) = 0;
            ind(i) = ind(i) + 1;
```

```
        end
    end
end
% get the cent point of the axis
if(rec_sign(1, 1) == 0)
    rec_0 = circshift(rec_0, 1, 2);
end
if(rec_0(1, 1) > rec_0(1, 2))
   rec_0(:, 1) = rec_0(:, 1) - 2*pi;
end
x_int=repmat(radius, [1, size(rec_0, 2)]).* cos(rec_0) + cent(2);
y_int=repmat(radius, [1, size(rec_0, 2)]).* sin(rec_0) + cent(1);
x_cent = (x_int(:, 1:2:end) + x_int(:, 2:2:end)) / 2;
y_cent = (y_int(:, 1:2:end) + y_int(:, 2:2:end)) / 2;
figure(4)
image(im_d)
hold on;
scatter(x_int(:), y_int(:));
scatter(x_cent(:), y_cent(:), 'r');
hold off;
% get the line profile
xcent = [x_cent(:, 1:6); x_cent(:, 7:12)];
ycent = [y_cent(:, 1:6); y_cent(:, 7:12)];
pfit = zeros(2, 6);
for i = 1:6
    pfit(:, i) = polyfit(xcent(:, i), ycent(:, i), 1)';
end
x_vect = 100:0.1:500;
vect_siz = size(x_vect, 2);
y_mat = zeros(6, vect_siz);
for i = 1:6
   y_mat(i, :) = polyval(pfit(:, i)', x_vect);
end
figure(5)
image(im_d)
hold on;
for i = 1:6
   plot(x_vect, y_mat(i, :), 'LineWidth', 2)
end
hold off;
```

Appendix 3: The computer programming scripts in TSET Cherenkov dosimetry

The code scripts are programmed for the work in the following publication.
Miao T, Petroccia H, Xie Y, et al. Computer animation body surface analysis of total skin electron radiation therapy dose homogeneity via Cherenkov imaging. *J Med.Imaging.* 2020;7(3):034002.

A3.1 2D Cherenkov distribution converting to the dose distribution

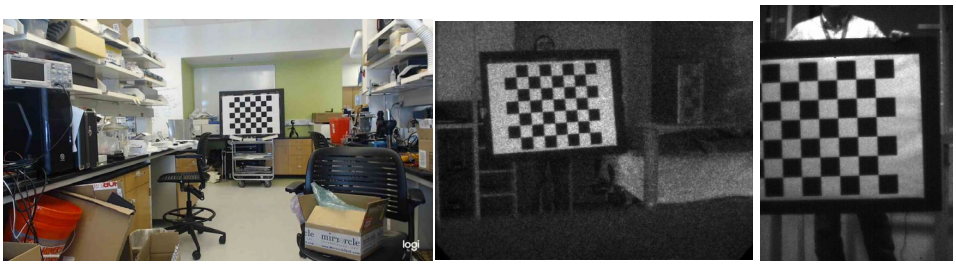
The Matlab script converted the Cherenkov images into the images of dose distribution through the calibration factor calculated from the dose measurement by diodes in the umbilical region of the patient in AP position. The script did the flat-field correction before the conversion, with the flat-field kernel made by imaging of a flat-field screen panel.

```
% File name: transformation_extraction.m
% Author: Tianshun Miao
clc
clear all
close all
% Read the Cherenkov image
che_img = imread('che_color/AP_che.PNG');
% Load the flat field correction panel
load('flat_sm_Solver2_0deg_08052019.mat');
ff_kernal = flat_sm_Solver2_0deg_08052019;
% Flat field correction of the Cherenkov images
im_corr = double(che_img) ./ff_kernal;
imagesc(im_corr, [0 1.7e4])
axis equal
% The conversion ratio is from the calibration
che_ratio =67/1.4e4;
dose_map = im_corr * che_ratio;
figure(2)
imagesc(dose_map, [0, 80])
axis equal
% unit conversion used for the c++ program
dose_map_16 = uint16(100*dose_map);
```

```
% Get colormap
cmap = cmocean('magma', 100);
dose_img = ind2rgb(round(dose_map), cmap);
figure(3)
imshow(dose_img)
% colorbar('southoutside')
colorbar('Ticks',[0 25 50 75 100],...
        'TickLabels',{'0','25','50','75','100'}, 'location','southoutside')
colormap(cmap);
caxis([0 100]);
set(gca, 'fontweight', 'bold', 'FontSize', 22)
figure(4)
imagesc(double(che_img)*10)
cmap2 = cmocean('magma', 17e4);
colormap(cmap2);
colorbar('Ticks',[0 40000 80000 120000 160000],...
        'TickLabels',{'0','40000','80000','120000','160000'}, 'location','southoutside')
caxis([0 17e4]);
axis equal
set(gca, 'fontweight', 'bold', 'FontSize', 22)
imwrite(dose_map_16, 'dose_map/AP_dose_map.PNG');
```

A3.2 Camera calibration of the TSET Cherenkov camera

The camera calibration extracted the information of the camera position, orientation and other properties, such as focal length. The procedures were similar as the projective transformation matrix extraction in light field study. However, the distance between the camera and the foreground objects, which were patients and slab phantoms, were different in the setup of TSET and the light field study. In TSET, the distance was around 4 meters, and a larger checkerboard pattern was used to make sure the corners were detected using the computer vision algorithm.



(a) (b) (c)

Figure A3.1: Calibration for the test purpose in the Dartmouth lab (a), and for calibration purpose in the Linac room at Penn Medicine (b) and DHMC (c)

In the script, the physical locations of the corners were measured and recorded. Then, the images of the checkerboard pattern were processed to extract the pixel locations of the corners. These two steps were performed in Matlab. The two piece of information were fed into the C++ script to extract the camera matrix with the Opencv library. This way, the format of the camera calibration matrix can be directly used in the projective texture mapping.

The calibration procedure was conducted both in Dartmouth and in University of Pennsylvania. The calibration made in Dartmouth was used for the test with water cylindrical phantom.

```
% File name: create_world_points.m
% Author: Tianshun Miao
clc
clear all
close all
% Load the data of the locations of the corners
load('intrin_corner_data.mat')
% Create the locations for the real physical location in unit of cm
x_vect = 30:-10:-30;
y_vect = -30:10:20;
x_vect = 10 * x_vect;
y_vect = 10 * y_vect;
[x_mat, y_mat] = meshgrid(x_vect, y_vect);
x_loc = x_mat(:);
y_loc = y_mat(:);
imagePoints = zeros(42, 2, 13);
for i = 1:13
    imagePoints(:, 1, i) = x_rec(:, i);
```

```
    imagePoints(:, 2, i) = y_rec(:, i);
end
worldPoints = [x_loc, y_loc];
file_name = 'world_points.txt';
dlmwrite(file_name, worldPoints);

% File name: create_img_points.m
% Author: Tianshun Miao
clc
clear all
close all
img_num = 11;
x_rec = zeros(42, img_num);
y_rec = zeros(42, img_num);

for i = 1:11
    img_name = strcat('tset_intrinsic/', num2str(i), '.PNG');
    img = imread(img_name);
    [imagePoints,boardSize] = detectCheckerboardPoints(img);
    x_rec(:, i) = imagePoints(:, 1);
    y_rec(:, i) = imagePoints(:, 2);
end
file_name1 = 'img_points.txt';
% Image number is 11
for i = 1:11
    dlmwrite(file_name1, imagePoints(:, :, i), '-append');
end

/*
        File Name: main.cpp
        Author: Tianshun Miao
*/
#include <opencv2/core/core.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
using namespace cv;
int main()
{
        // First to read the image points, and world points from txt file
        ifstream world_p, img_p;
        world_p.open("../data/dartmouth/world_points.txt");
```

```
img_p.open("../data/dartmouth/img_points.txt");
int img_num = 11; // number of images
int p_num = 42;  // number of corners
Size2i img_siz;
img_siz.height = 1600;  // image sensor pixel size
img_siz.width = 1200;
string buf;
Vec3f p_buf;
Vec2f p2_buf;
vector<Point3f> world_rec;
vector<Point2f> img_rec;
vector<vector<Point3f>> world_dict;
vector<vector<Point2f>> img_dict;
stringstream ss;
char p_h;
// read the world points
for (int i = 0; i < p_num; i++)
{
        getline(world_p, buf);
        ss.str(buf);
        ss >> p_buf[0] >> p_h >> p_buf[1] >> p_h >> p_buf[2];
        cout << p_buf << endl;
        world_rec.push_back(p_buf);
        ss.clear();
        buf.clear();
}
// read the image points
for (int i = 0; i < img_num; i++)
{
        for (int j = 0; j < p_num; j++)
        {
                getline(img_p, buf);
                ss.str(buf);
                ss >> p2_buf[0] >> p_h >> p2_buf[1];
                img_rec.push_back(p2_buf);
                ss.clear();
                buf.clear();
        }
        img_dict.push_back(img_rec);
        img_rec.clear();
        world_dict.push_back(world_rec);
}
world_p.close();
img_p.close();
// define the parameters for camera calibration
Mat K;
```

```
        Mat D;
        vector< Mat > rvecs, tvecs;
        int flag = 0;
        flag |= CALIB_FIX_PRINCIPAL_POINT;
        calibrateCamera(world_dict, img_dict, img_siz, K, D, rvecs, tvecs, flag);
        // Get the parameters that can be used in vtk code: quote from Michael Jermym
        // calculate camera info from extrinsic matrix: C = -transpose(R)*T
        Mat vec_r = rvecs[img_num - 1];
        Mat vec_t = tvecs[img_num - 1];

        cv::Mat R;
        cv::Rodrigues(vec_r, R);
        cv::Mat cam_translation = -R.t() * vec_t;
        cv::Mat zvec = (cv::Mat_<double>(3, 1) << 0, 0, 1);
        cv::Mat cam_rotation = R.t()*zvec;

        // viewing angle is 2*arctan(h/(2f)) where f is focal length in y, h is height in
pixels
        Mat mat_camera = K;
        int height = 1200;
        int width = 1600;
        double fy = mat_camera.at<double>(1, 1);
        double angle_view = (double) 2.0 * atan(height / (2.0 * fy));
        angle_view *= 180.0 / 3.14159;
        // compute camera properties
        double position[3]{ cam_translation.at<double>(0),
cam_translation.at<double>(1), cam_translation.at<double>(2) };
        double focal_point[3];
        double focal_vec[3]{ R.at<double>(2, 0), R.at<double>(2, 1), R.at<double>(2, 2)
};
        double view_up[3]{ -R.at<double>(1, 0), -R.at<double>(1, 1), -R.at<double>(1,
2) };

        double dist = sqrt(position[0] * position[0] + position[1] * position[1] +
position[2] * position[2]);

        focal_point[0] = position[0] + dist * focal_vec[0];
        focal_point[1] = position[1] + dist * focal_vec[1];
        focal_point[2] = position[2] + dist * focal_vec[2];

        FileStorage fs("camera_calib_complete.yml", FileStorage::WRITE);
        fs << "cam_mat" << K;
        fs << "dist_mat" << D;
        fs << "came rotation matrix" << R;
        fs << "cam_translation_vect" << cam_translation;
        fs << "cam_rotation_vect" << vec_r;
```

```
        ofstream vtk_cam_list;
        vtk_cam_list.open("vtk_cam_list.txt");
        vtk_cam_list << "camera position: " << position[0] << " " << position[1] << " "
<< position[2] << endl;
        vtk_cam_list << "focal point: " << focal_point[0] << " " << focal_point[1] << " "
<< focal_point[2] << endl;
        vtk_cam_list << "view up angle: " << view_up[0] << " " << view_up[1] << " " <<
view_up[2] << endl;
        vtk_cam_list << "angle of view: " << angle_view << endl;
        vtk_cam_list.close();
        return EXIT_SUCCESS;
}
# CMakeLists.txt
PROJECT(Camera_calib)

# Specify the Opencv Library location
set(OpenCV_DIR "D:/opencv/build/")

# Find and include OpenCV library
find_package(OpenCV REQUIRED)
include_directories(${OpenCV_INCLUDE_DIRS})

# Collect and include all of our source files (.h and .cxx files)
include_directories(${CMAKE_CURRENT_SOURCE_DIR}
${CMAKE_CURRENT_BINARY_DIR})
file(GLOB CXX_FILES *.cpp)

# We are creating an executable project, and including the source files into it
add_executable(Camera_calib ${CXX_FILES})

# Set the startup project in Visual Studio to be our project (rather than the extra stuff
CMake builds)
set_property(DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR} PROPERTY
VS_STARTUP_PROJECT camera_)

# Tell the project which libraries to link
target_link_libraries(Camera_calib ${OpenCV_LIBS})
```

A3.3 Projective Texture mapping to VTK format

The C++ scripts read the 3D file of the patient and 2D dose image. The it overlaid the 2D dose image to the 3D model. The output of the script was in VTK format. The scripts used the proprietary library from DoseOptics, which was commercially available through the CDose software.

```
/*
        File Name: ui.h
        Author: Tianshun Miao, with help from Michael Jermyn
*/

// Prevent this header file from being included multiple times
#pragma once

// VTK header files
#include <vtkPNGReader.h>
#include <vtkInteractorStyleTrackballCamera.h>
#include <vtkActor.h>
#include <vtkImageMapToWindowLevelColors.h>
#include <vtkImageMapper.h>
#include <vtkImageMapper3D.h>
#include <vtkImageActor.h>
#include <vtkImageData.h>
#include <vtkProperty.h>
#include <vtkDICOMImageReader.h>
#include <QVTKOpenGLWidget.h>
#include <vtkGenericOpenGLRenderWindow.h>
#include <vtkSmartVolumeMapper.h>
#include <vtkNew.h>
#include <vtkVolumeProperty.h>
#include <vtkVolume.h>
#include <vtkRenderWindow.h>
#include <vtkRenderer.h>
#include <vtkPolyDataMapper.h>
#include <vtkCamera.h>
#include <vtkCoordinate.h>
#include <vtkImageFlip.h>
#include <vtkPolyData.h>
#include <vtkPNGReader.h>
#include <vtkUnsignedShortArray.h>
#include <vtkDataSet.h>
```

```
#include <vtkPointData.h>
#include <vtkProperty.h>
#include <vtkProperty2D.h>
#include <vtkOBJReader.h>
#include <vtkPolyDataMapper.h>
#include <vtkPointData.h>
#include <vtkXMLPolyDataWriter.h>
#include <vtkImageActor.h>
#include <vtkSmartPointer.h>
#include <vtkImageViewer2.h>
#include <vtkActor2D.h>

// Qt header files
#include <QPointer>
#include <QFileDialog.h>
#include <QMainWindow.h>
#include <QLayout.h>
#include <QPushButton.h>
#include <QVTKWidget.h>
#include <QSlider>
#include <QLabel>
#include <QSurfaceFormat>
#include <QStaticText>
#include <QTextEdit>

// OpenCV header files
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/objdetect.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
// Add the projection function
#include "dov_projection.h"

// Class that represents the main window for our application
class ui : public QMainWindow
{
        Q_OBJECT
private:
```

```
        // Declaration of  CLASS VARIABLES HERE
        QPushButton * but1;
        QPushButton * but2;
        QPushButton * but3;
        QPushButton * but4;
        QPushButton *bg_but, *tx_but, *obj_but;
        QTextEdit * txt_cam_angle;
        QTextEdit *bg_path, *tx_path, *obj_path;
        QTextEdit *output_file_name;
        QLabel *Text_bg, *Text_tx, *Text_obj;
        QLabel *output_file, *View_angle;
        QSlider *sl1;
        QLabel *label1;
        QVBoxLayout *layout_vertical;
        QHBoxLayout *layout_horizontal, *layout_horizontal_bg, *layout_horizontal_tx,
*layout_horizontal_obj;
        QPointer<QVTKOpenGLWidget> viewport;
        vtkNew<vtkPNGReader> img_reader, img_reader_tx;
        vtkNew<vtkGenericOpenGLRenderWindow> window1;
        QWidget *widget;
        vtkNew<vtkVolume> vol;
        vtkNew<vtkRenderer> ren1;
        vtkNew<vtkRenderer> ren2;
        vtkSmartPointer<vtkPolyData> surface;
        vtkNew<vtkOBJReader> obj_reader;
        vtkNew<vtkPolyDataMapper> mapper;
        vtkNew<vtkActor> actor;
        vtkNew<vtkActor> actor2;
        vtkNew<vtkCamera> cam1;
        vtkNew<vtkRenderWindowInteractor> renderWindowInteractor;
        vtkNew<vtkPolyDataMapper> poly_mapper;
        vtkNew<vtkInteractorStyleTrackballCamera> style;
        vtkNew<vtkImageData> image;
        vtkSmartPointer<vtkImageData> img, img_tx;
        vtkSmartPointer<vtkMapper> mpper;
        vtkNew<vtkActor2D> actor2d;
        vtkSmartPointer<vtkImageActor> img_actor;
        vtkSmartPointer<vtkImageMapper> imageMapper;
        vtkNew<vtkImageActor> imgactor;
        vtkNew<vtkCoordinate> coord;
        vtkNew<vtkImageFlip> flipYFilter;
        vtkNew<vtkImageFlip> flipZFilter;
        vtkPolyData * surf;
        vtkNew<vtkPNGReader> reader2;
        vtkRenderWindow * rendw;
```

```
        int num_p;
public:

// Constructor (happens when created)
        ui()
        {
                // Resize the window
                this->resize(1800, 800); // CHANGE THIS
                img = vtkSmartPointer<vtkImageData>::New();
                img_tx = vtkSmartPointer<vtkImageData>::New();
                img_actor = vtkSmartPointer<vtkImageActor>::New();
                imageMapper = vtkSmartPointer<vtkImageMapper>::New();
                surface = vtkSmartPointer<vtkPolyData>::New();
                // Create the "central" (primary) widget for the window
                QWidget *widget = new QWidget();
                this->setCentralWidget(widget);

                // Create your widgets
                // Configuration of vtkopegl window

        vtkOpenGLRenderWindow::SetGlobalMaximumNumberOfMultiSamples(0);

        QSurfaceFormat::setDefaultFormat(QVTKOpenGLWidget::defaultFormat());
                viewport = new QVTKOpenGLWidget();
                viewport->SetRenderWindow(window1.Get());
                viewport->GetRenderWindow()->SetSize(800, 600);
                // configuration of button etc.
                but1 = new QPushButton("Load OBJ");
                but1->setFixedSize(150, 40);
                but2 = new QPushButton("Export Camera");
                but2->setFixedSize(150, 40);
                but3 = new QPushButton("Export Texture");
                but3->setFixedSize(150, 40);
                but4 = new QPushButton("Update Cam");
                but4->setFixedSize(150, 40);
                txt_cam_angle = new QTextEdit();
                txt_cam_angle->setFixedSize(110, 40);
                // Layout the widgets
                // YOUR CODE HERE
                layout_vertical = new QVBoxLayout();
                layout_horizontal = new QHBoxLayout();
                layout_horizontal_bg = new QHBoxLayout();
                layout_horizontal_tx = new QHBoxLayout();
                layout_horizontal_obj = new QHBoxLayout();
                widget->setLayout(layout_vertical);
                layout_vertical->addWidget(viewport);
```

```
                // the lines corresponding to read the background image file, texture image
file and obj file etc.
                layout_vertical->addLayout(layout_horizontal_bg);
                layout_vertical->addLayout(layout_horizontal_tx);
                layout_vertical->addLayout(layout_horizontal_obj);
                QFont font_label;
                font_label.setBold(true);
                font_label.setPointSize(10);
                Text_bg = new QLabel("Background Image");
                Text_bg->setFont(font_label);
                Text_tx = new QLabel("Texture Image");
                Text_tx->setFont(font_label);
                Text_obj = new QLabel("3D Obj File");
                Text_obj->setFont(font_label);
                Text_bg->setFixedSize(180, 40);
                Text_tx->setFixedSize(180, 40);
                Text_obj->setFixedSize(180, 40);
                layout_horizontal_bg->addWidget(Text_bg);
                layout_horizontal_tx->addWidget(Text_tx);
                layout_horizontal_obj->addWidget(Text_obj);
                layout_horizontal_bg->setAlignment(Qt::AlignLeft);
                layout_horizontal_tx->setAlignment(Qt::AlignLeft);
                layout_horizontal_obj->setAlignment(Qt::AlignLeft);
                // Add the text edit box to show the file path
                bg_path = new QTextEdit;
                tx_path = new QTextEdit;
                obj_path = new QTextEdit;
                bg_path->setFixedSize(800, 40);
                tx_path->setFixedSize(800, 40);
                obj_path->setFixedSize(800, 40);
                layout_horizontal_bg->addWidget(bg_path);
                layout_horizontal_tx->addWidget(tx_path);
                layout_horizontal_obj->addWidget(obj_path);
                bg_but = new QPushButton("Browse");
                tx_but = new QPushButton("Browse");
                obj_but = new QPushButton("Browse");
                bg_but->setFixedSize(150, 40);
                tx_but->setFixedSize(150, 40);
                obj_but->setFixedSize(150, 40);
                layout_horizontal_bg->addWidget(bg_but);
                layout_horizontal_tx->addWidget(tx_but);
                layout_horizontal_obj->addWidget(obj_but);
                // add output and angle of view label
                output_file = new QLabel("Output File Name");
                output_file->setFixedSize(180, 40);
                output_file->setFont(font_label);
```

```
                View_angle = new QLabel("Angle of View");
                View_angle->setFont(font_label);
                View_angle->setFixedSize(180, 40);
                layout_horizontal_bg->addSpacing(20);
                layout_horizontal_bg->addWidget(output_file);
                layout_horizontal_tx->addSpacing(20);
                layout_horizontal_tx->addWidget(View_angle);
                output_file_name = new QTextEdit;
                output_file_name->setFixedSize(110, 40);
                layout_horizontal_bg->addWidget(output_file_name);
                layout_horizontal_tx->addWidget(txt_cam_angle);
                layout_horizontal_tx->addWidget(but4);
                // Add the widge to load and do the texture
                layout_vertical->addLayout(layout_horizontal);
                layout_horizontal->addWidget(but1);
                layout_horizontal->addWidget(but2);
                layout_horizontal->addWidget(but3);
                //layout_horizontal->addWidget(txt_cam_angle);
                //layout_horizontal->addWidget(but4);
                // Configuration of the camera
                /*cam1->SetPosition(-124.736, -104.852, -654.272);
                cam1->SetFocalPoint(-12.5727, 11.141, 5.87349);
                cam1->SetViewUp(0.529516, -0.797376, 0.289491);
                cam1->SetViewAngle(2);
*/
                //cam1->SetPosition(-109.83, 80, 442.673);
                //cam1->SetFocalPoint(30.56, 83.6, 68.49);
                //cam1->SetViewUp(0.073, 0.9967, 0.0330712);   // calibration is 0.073
0.9967 0.033712
                //cam1->SetViewAngle(24.8);
                cam1->SetPosition(-109.83, -40.6, 442.673);
                cam1->SetFocalPoint(30.56, -36.4, 68.49);
                cam1->SetViewUp(0.073, 0.9967, 0.0330712);   // calibration is 0.073
0.9967 0.033712
                cam1->SetViewAngle(24.8);
                //cam1->SetPosition(45.13, -13.70, 492.5);
                //cam1->SetFocalPoint(19.12, -8.04, -1.53);
                //cam1->SetViewUp(0.03304, 0.999, 0);   // calibration is 0.073 0.9967
0.033712
                //cam1->SetViewAngle(17.5);

                // Configure coordinate
                coord->SetCoordinateSystemToWorld();
                rendw = viewport->GetRenderWindow();
                // Connected widget signals to slots
```

```
		connect(but1, SIGNAL(released()), this, SLOT(load_obj()));
		connect(but2, SIGNAL(released()), this, SLOT(Export_Camera()));
		connect(but3, SIGNAL(released()), this, SLOT(Export_Texture()));
		connect(but4, SIGNAL(released()), this, SLOT(Update_Cam()));
		connect(bg_but, SIGNAL(released()), this, SLOT(Load_bg()));
		connect(tx_but, SIGNAL(released()), this, SLOT(Load_tx()));
		connect(obj_but, SIGNAL(released()), this, SLOT(Load_3d()));
		// Display the window
		this->show();
	}

public slots: // This tells Qt we are defining slots here

	// YOUR CODE HERE
	// the load data code loads the dicom ct files
	void load_obj()
	{
		// load the image file in the background
		//std::string img_file;
		//img_file = "input_data/dartmouth_data/bg_pos2.PNG";
		//img_reader->SetFileName(img_file.c_str());
		img_reader->SetFileName(bg_path->toPlainText().toLocal8Bit().data());
		img_reader->Update();
		img = img_reader->GetOutput();
		imgactor->SetInputData(img);
		ren1->AddActor(imgactor);
		// load the image file as the texture
		//std::string img_tx_file;
		//img_tx_file = "input_data/dartmouth_data/che_pos2.PNG";
		//img_reader_tx->SetFileName(img_tx_file.c_str());
		img_reader_tx->SetFileName(tx_path-
>toPlainText().toLocal8Bit().data());
		img_reader_tx->Update();
		img_tx = img_reader_tx->GetOutput();
		// fill the gap in the render window
		double scale = 0.51;
		double origin[3];
		double spacing[3];
		int extent[6];
		img->GetOrigin(origin);
		img->GetSpacing(spacing);
		img->GetExtent(extent);
		vtkSmartPointer<vtkCamera> cam2 =
			ren1->GetActiveCamera(); // Use the renderer for the image in this
line
		cam2->ParallelProjectionOn();
```

```
                double xc = origin[0] + 0.5*(extent[0] + extent[1])*spacing[0];
                double yc = origin[1] + 0.5*(extent[2] + extent[3])*spacing[1];
                double yd = (extent[3] - extent[2] + 1)*spacing[1];
                double d = cam2->GetDistance();
                cam2->SetParallelScale(scale * yd);
                cam2->SetFocalPoint(xc, yc, 0.0);
                cam2->SetPosition(xc, yc, d);
                cam2->SetViewUp(0, 1, 0);
                // load the obj file
                //std::string obj_file;
                //obj_file = "input_data/dartmouth_data/model_pos2.obj";
                //obj_reader->SetFileName(obj_file.c_str());
                obj_reader->SetFileName(obj_path->toPlainText().toLocal8Bit().data());
                obj_reader->Update();
                poly_mapper->SetInputConnection(obj_reader->GetOutputPort());
                surface = obj_reader->GetOutput();
                actor->SetMapper(poly_mapper);
                ren2->AddActor(actor);
                ren2->SetActiveCamera(cam1);
                // combine renderer into render window
                ren1->SetLayer(0);
                ren1->InteractiveOff();
                ren2->SetLayer(1);
                renderWindowInteractor->SetRenderWindow(rendw);
                renderWindowInteractor->SetInteractorStyle(style);
                rendw->SetNumberOfLayers(2);
                rendw->AddRenderer(ren1);
                rendw->AddRenderer(ren2);
                rendw->Render();
                renderWindowInteractor->Start();
        }

void Export_Camera()
        {
                ren1->Render();
                ofstream camera_data;
                camera_data.open("Camera_data.txt");
                double cam_pos1, cam_pos2, cam_pos3;
                cam1->GetPosition(cam_pos1, cam_pos2, cam_pos3);
                double cam_focal1, cam_focal2, cam_focal3;
                cam1->GetFocalPoint(cam_focal1, cam_focal2, cam_focal3);
                double cam_viewup1, cam_viewup2, cam_viewup3;
                cam1->GetViewUp(cam_viewup1, cam_viewup2, cam_viewup3);
                double cam_angle_view = cam1->GetViewAngle();
                camera_data << "Position: "  << cam_pos1 << " " << cam_pos2 << " " <<
cam_pos3 << std::endl;
```

```
		camera_data << "focal point: " << cam_focal1 << " " << cam_focal2 << "
" << cam_focal3 << std::endl;
		camera_data << "view up: " << cam_viewup1 << " " << cam_viewup2 <<
" " << cam_viewup3 << std::endl;
		camera_data << "angle of view: " << cam_angle_view << std::endl;
		camera_data.close();
	}
	void Export_Texture()
	{
		dov::projection proj;
		ren1->Render();
		vtkSmartPointer<vtkCamera> cam_test = ren1->GetActiveCamera();
		// Filp the images
		vtkSmartPointer<vtkImageFlip> flipXFilter =
			vtkSmartPointer<vtkImageFlip>::New();
		flipXFilter->SetFilteredAxis(0); // flip x axis
		flipXFilter->SetInputData(img_tx);
		flipXFilter->Update();

		vtkSmartPointer<vtkImageFlip> flipYFilter =
			vtkSmartPointer<vtkImageFlip>::New();
		flipYFilter->SetFilteredAxis(1); // flip Y axis
		flipYFilter->SetInputConnection(flipXFilter->GetOutputPort());
		flipYFilter->Update();
		vtkSmartPointer<vtkImageData> img2 = flipYFilter->GetOutput();

		proj.set_image(img2); // The image to project [vtkImageData]
		proj.set_surface(surface); // The surface [vtkPolyData]
		proj.set_camera(cam1); // The camera [vtkCamera]
		proj.cumulative = false; // If you have multi-frame image data you are
projecting, this is whether to accumulate them
		proj.visible_only = true; // Whether to only project to the first intersection
		vtkSmartPointer<vtkPolyData> surface_projected = proj.project(); //
Perform the projection, and return the output surface
		// The output surface has scalar fields "s0", "s1", etc. for each image slice
provided
		surface_projected->GetPointData()->SetActiveScalars("s0");
		// Export to the vtp xml file
		vtkSmartPointer<vtkXMLPolyDataWriter> writer =
vtkSmartPointer<vtkXMLPolyDataWriter>::New();
		QString output_name = output_file_name->toPlainText();
		output_name = output_name + ".vtp";
		writer->SetFileName(output_name.toLocal8Bit().data());
		writer->SetInputData(surface_projected);
		writer->Write();
	}
```

```
        void Update_Cam()
        {
                double cam_angle = txt_cam_angle->toPlainText().toDouble();
                cam1->SetViewAngle(cam_angle);
                rendw->Render();
                //std::cout << "The cam_angle is " << cam_angle << std::endl;
        }
        void Load_bg()
        {
                QString bg_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                bg_path->setText(bg_file_path);
        }
        void Load_tx()
        {
                QString tx_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                tx_path->setText(tx_file_path);
        }
        void Load_3d()
        {
                QString obj_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                obj_path->setText(obj_file_path);
        }
};
```

A3.4 Conversion from the 3D dose texture in VTK format to 2D UV image format

The dose distribution was represented as the 3D point cloud in the VTK format, which could not be used to accumulate to the dose distribution for different position. The C++ and Matlab scripts converted the 3D dose distribution in VTK format to the UV map, specified in the .obj file. The C++ script read the .obj file and built up the mapping relationship between the 3D vertices and the UV vertices. The C++ script used the external opensource library: tinyobjleader, which can be extracted at: https://github.com/tinyobjloader/tinyobjloader. The Matlab saved the UV map of the dose distribution into the variable called "pixel_mat". For each positions, the research needed to manually change the name of the input files and saved the output file into the appropriate matlab format file corresponding to the position of the patient.

```
/*
        File Name: main.cpp
        Author: Tianshun Miao, with help from Michael Jermyn
*/

#include <iostream>
#include <fstream>
#define TINYOBJLOADER_IMPLEMENTATION // define this in only *one* .cc
#include "tiny_obj_loader.h"
using namespace std;

int main()
{
        // Input obj file
        std::string inputfile =
"C:/Users/Turing/Dropbox/research/pt13_stanford_overrotate/3d_model/RPO_model.obj
";
        std::string tex_loc_file = "tex_location_list.txt";   // list of 2d UV vertices
        std::string face_vert_list = "face_vert_list.txt";    // list of faces with verice indices
        std::string face_tex_list = "face_tex_list.txt";      // list of faces with uv indices
        std::string vert_loc_list = "vert_location_list.txt";  // list of 3d vertice locations
        // create 4 files used for conversion
```

```
	ofstream tex_list, face_vert, face_tex, vert_list;
	tex_list.open(tex_loc_file);
	face_vert.open(face_vert_list);
	face_tex.open(face_tex_list);
	vert_list.open(vert_loc_list);
	tinyobj::attrib_t attrib;
	std::vector<tinyobj::shape_t> shapes;
	std::vector<tinyobj::material_t> materials;

	std::string warn;
	std::string err;

	bool ret = tinyobj::LoadObj(&attrib, &shapes, &materials, &warn, &err,
inputfile.c_str(), NULL, false, true);
	for (int i = 0; i < attrib.texcoords.size()/2; i++)
	{
		tex_list << attrib.texcoords[2 * i] << " ";
		tex_list << attrib.texcoords[2 * i + 1] << endl;
	}
	for (int i = 0; i < attrib.vertices.size() / 3; i++)
	{
		vert_list << attrib.vertices[3 * i] << " ";
		vert_list << attrib.vertices[3 * i + 1] << " ";
		vert_list << attrib.vertices[3 * i + 2] << endl;
	}
	for (int i = 0; i < shapes[0].mesh.indices.size()/4; i++)
	{
		// write the vertex correspondence into the text file
		face_vert << shapes[0].mesh.indices[4 * i].vertex_index << " ";
		face_vert << shapes[0].mesh.indices[4 * i + 1].vertex_index << " ";
		face_vert << shapes[0].mesh.indices[4 * i + 2].vertex_index << " ";
		face_vert << shapes[0].mesh.indices[4 * i + 3].vertex_index << endl;
		// write the texture correspondence into the text file
		face_tex << shapes[0].mesh.indices[4 * i].texcoord_index << " ";
		face_tex << shapes[0].mesh.indices[4 * i + 1].texcoord_index << " ";
		face_tex << shapes[0].mesh.indices[4 * i + 2].texcoord_index << " ";
		face_tex << shapes[0].mesh.indices[4 * i + 3].texcoord_index << endl;
	}

	face_vert.close();
	face_tex.close();
	tex_list.close();
	vert_list.close();
	system("Pause");
}
```

```
% File name: General_steps.m
% Author: Tianshun Miao
%% Clear the workspace
clc
clear all
close all
tic
%% Input files of the vtk, obj information
% Read vtk data file
data_vtk = dlmread('vtk_texture_file/dose_texture/lpo_dose.txt', ',', 1, 0);
val = data_vtk(:, 7);
pos = data_vtk(:, end-2:end);
% load the positon of the obj vertex
vert_list = dlmread('LPO/vert_location_list.txt');

% load the correspondence of faces, with index of four corners in the 3D
% space
f_vert_list = dlmread('LPO/face_vert_list.txt');

% load the correspondence of faces, with index of the four corners in the
% texture coordinate
f_tex_list = dlmread('LPO/face_tex_list.txt');

% then load the list of texture point coordinate
tex_list = dlmread('LPO/tex_location_list.txt');

%% Assign the texture to the vertices defined in the obj file use the kd tree
% create kdtree searcher
vtk_num = size(data_vtk, 1);
vtk_lut = KDTreeSearcher(pos);

% create texture to point
ind_nn = knnsearch(vtk_lut, vert_list);
pixel_val_list = val(ind_nn); % The list of texture for the vertices in obj

% The product is the pixel_val_list

%% Create LUT of Vertices from 3D point to 2D texture location
% Flatten the matrix to vector
vert_vect = f_vert_list(:);
tex_vect = f_tex_list(:);
% Create the lut
max_num = max(vert_vect)+1;
vert_tex_dict = zeros(max_num, 5);
vert_tex_dict(:, 2:end) = NaN;
temp_vect = zeros(1, 4);
```

```
for i = 1:size(vert_vect, 1)
    vert_ind = vert_vect(i);
    tex_ind = tex_vect(i);
    dict_ind = vert_ind + 1;
    % Check if the texture index is already in the list
    temp_vect = vert_tex_dict(dict_ind, 2:end);
    flag_same = sum(temp_vect == tex_ind);
    if(flag_same == 0)
        vert_tex_dict(dict_ind, 1) = vert_tex_dict(dict_ind, 1)+1;
        vert_tex_dict(dict_ind, vert_tex_dict(dict_ind, 1)+1) = tex_ind;
    end
end

%% Assign value to the vertices in texture coordinate
t_num = size(tex_list, 1);              % Total number of Texture Vertices
val_tex_list = zeros(t_num, 2);         % Output: the texture assigned
v_num = size(pixel_val_list, 1);        % The total number of 3D vertices
for i = 1:v_num
    for j = 1:vert_tex_dict(i, 1)       % One 3D point corresponding to multiple 2D Texture
        tex_ind = vert_tex_dict(i, j+1)+1;
        val_tex_list(tex_ind, 1) = val_tex_list(tex_ind, 1) + 1;
        val_tex_list(tex_ind, 2) = pixel_val_list(i);
    end
end

%% Create the center of the faces
f_num = size(f_tex_list, 1);
face_coord = zeros(f_num, 2);
for i = 1:f_num
    ind = f_tex_list(i, :);
    ind = ind + 1; % convert from c++ index to matlab index
    coord4 = tex_list(ind, :);   % get the 4 coordinates of the quad
    f_coord = mean(coord4, 1);
    face_coord(i, :) = f_coord;
end
% Build up the kd tree of the face centers
face_coord2 = face_coord * 1000;
face_lut = KDTreeSearcher(face_coord2);

%% Making texture
% Make the scatter point interpolant
val_tex = val_tex_list(:, 2);
F = scatteredInterpolant(1000*tex_list(:, 1), 1000*tex_list(:, 2), val_tex);
k_num = 6;                          % nearest neighbor number
pixel_mat = zeros(1000, 1000);      % intiating the texture map
flag_mat = zeros(1000, 1000);
```

```
f_tex = f_tex_list + 1;
parfor i = 1:1000
   for j = 1:1000
      pixel_loc = [j, i];
      % First check if the location is in the region of foreground
      f_indx = knnsearch(face_lut, pixel_loc, 'K', k_num);
      forground_flag = false;
      for k_ind = 1:k_num
         f_tex_ind = f_indx(k_ind);
         corner_ind = f_tex(f_tex_ind, :);
         corner_loc = 1000 * tex_list(corner_ind, :);
         forground_flag = inpolygon(j, i, corner_loc(:, 1), corner_loc(:, 2));
         % If it is the foreground pixel, then assign the texture based on the four
         % corners
         if(forground_flag)
           corner_val = val_tex(corner_ind);
%              pixel_mat(i, j) = F(j, i);
%              pixel_mat(i, j) = 1;
            flag_mat(i, j) = 1;
           break;
         end
      end
   end
end
pixel_list = zeros(1000^2, 2);
counter = 1;
for i = 1:1000
   for j = 1:1000
   if (flag_mat(i, j) == 1)
      pixel_list(counter, :) = [j, i];
      counter = counter + 1;
   end
   end
end
pixel_list(counter:end, :) = [];

pixel_val = F(pixel_list(:, 1), pixel_list(:, 2));
for x = 1:size(pixel_list, 1)
    j = pixel_list(x, 1);
    i = pixel_list(x, 2);
    pixel_mat(i, j) = pixel_val(x);
end
toc
```

A3.5 Dose distribution accumulated from all positions through the 2D UV map

The step of the dose distribution accumulation was straight forward by summing up the 2D UV map distribution for all positions because they shared the same the UV map. The function of "edge_padding" added the padding to the boundary of the UV map for display purpose to film the cutting-seam from the UV map generation.

```
% File name: create_uv_map.m
% Author: Tianshun Miao
clc
clear all
close all
% load mat file of the uv map
load('ap_uv.mat');
load('pa_uv.mat');
load('rao_uv.mat');
load('lao_uv.mat');
load('rpo_uv.mat');
load('lpo_uv.mat');
% convert with ratio
rat_val = 10;
ap_val = ap_uv * rat_val;
pa_val = pa_uv * rat_val;
rao_val = rao_uv * rat_val;
lao_val = lao_uv * rat_val;
rpo_val = rpo_uv * rat_val;
lpo_val = lpo_uv * rat_val;

% for visualization only
ap_pad = edge_padding(ap_val);
pa_pad = edge_padding(pa_val);
rao_pad = edge_padding(rao_val);
lao_pad = edge_padding(lao_val);
rpo_pad = edge_padding(rpo_val);
lpo_pad = edge_padding(lpo_val);
% Get colormap
cmap = cmocean('magma', 1.7e5);

% convert uv map to image
ap_map = ind2rgb(round(ap_pad), cmap);
pa_map = ind2rgb(round(pa_pad), cmap);
```

```
rao_map = ind2rgb(round(rao_pad), cmap);
lao_map = ind2rgb(round(lao_pad), cmap);
rpo_map = ind2rgb(round(rpo_pad), cmap);
lpo_map = ind2rgb(round(lpo_pad), cmap);
% imwrite(ap_map, 'ap_map.PNG');
% imwrite(pa_map, 'pa_map.PNG');
% imwrite(rao_map, 'rao_map.PNG');
% imwrite(lao_map, 'lao_map.PNG');
% imwrite(rpo_map, 'rpo_map.PNG');
% imwrite(lpo_map, 'lpo_map.PNG');
% % get accumulative colormap
accu_val = ap_val + pa_val + rao_val + lao_val + rpo_val + lpo_val;
accu_pad = edge_padding(accu_val);
cmap2 = cmocean('magma', 5e5);
accu_color = ind2rgb(round(accu_pad), cmap2);
% imwrite(accu_color, 'accu_map.PNG');

function [uv_map2] = edge_padding(uv_map)
%EDGE_PADDING Summary of this function goes here
%   Detailed explanation goes here
img_patch = zeros(3, 3);
uv_map2 = uv_map;
for i = 2:999
   for j = 2:999
      if(uv_map(i, j) == 0)
         img_patch = uv_map(i-1:i+1, j-1:j+1);
         uv_map2(i, j) = max(img_patch(:));
      end
   end
end
end
```

A3.6 Statistical analysis of the area dose distribution

The statistical analysis creates the area dose distribution from the UV map of the cumulative dose distribution and the 3D model of the patient. The 3D model, which was in .obj file, was first triangulated using the C++ scripts, with the library of the "tinyobjloader". The Matlab scripts calculated the area of the triangle element and the mean dose value of the element. The area dose distribution was represented as the histogram as the output. In creating the histogram, the Matlab script used the function of "histwc". It was an opensource function and could be extracted at https://www.mathworks.com/matlabcentral/fileexchange/42493-generate-weighted-histogram.

```
/*
        File Name: main.cpp
        Author: Tianshun Miao

#include <iostream>
#include <fstream>
#define TINYOBJLOADER_IMPLEMENTATION // define this in only *one* .cc
#include "tiny_obj_loader.h"
using namespace std;

int main()
{
        std::string inputfile = "accu_3d_uv.obj";
        std::string tex_loc_file = "tex_location_list.txt";
        std::string face_vert_list = "face_vert_list.txt";
        std::string face_tex_list = "face_tex_list.txt";
        std::string vert_loc_list = "vert_location_list.txt";
        ofstream tex_list, face_vert, face_tex, vert_list;
        tex_list.open(tex_loc_file);
        face_vert.open(face_vert_list);
        face_tex.open(face_tex_list);
        vert_list.open(vert_loc_list);
        tinyobj::attrib_t attrib;
        std::vector<tinyobj::shape_t> shapes;
        std::vector<tinyobj::material_t> materials;
```

```
	std::string warn;
	std::string err;

	bool ret = tinyobj::LoadObj(&attrib, &shapes, &materials, &warn, &err,
inputfile.c_str(), NULL, true, true);
	for (int i = 0; i < attrib.texcoords.size() / 2; i++)
	{
		tex_list << attrib.texcoords[2 * i] << " ";
		tex_list << attrib.texcoords[2 * i + 1] << endl;
	}
	for (int i = 0; i < attrib.vertices.size() / 3; i++)
	{
		vert_list << attrib.vertices[3 * i] << " ";
		vert_list << attrib.vertices[3 * i + 1] << " ";
		vert_list << attrib.vertices[3 * i + 2] << endl;
	}
	// in triangulated case
	for (int i = 0; i < shapes[0].mesh.indices.size() / 3; i++)
	{
		// write the vertex correspondence into the text file
		face_vert << shapes[0].mesh.indices[3 * i].vertex_index << " ";
		face_vert << shapes[0].mesh.indices[3 * i + 1].vertex_index << " ";
		face_vert << shapes[0].mesh.indices[3 * i + 2].vertex_index << endl;
		// write the texture correspondence into the text file
		face_tex << shapes[0].mesh.indices[3 * i].texcoord_index << " ";
		face_tex << shapes[0].mesh.indices[3 * i + 1].texcoord_index << " ";
		face_tex << shapes[0].mesh.indices[3 * i + 2].texcoord_index << endl;
	}

	face_vert.close();
	face_tex.close();
	tex_list.close();
	system("Pause");
}
```

```
% File name: area_analysis.m
% Author: Tianshun Miao
clc
clear all
close all
% Load and covert the UV map
uv_map = load('accu.mat').im_total;
% load the positon of the obj vertex
vert_list = dlmread('texture_files/AP_triangulated/vert_location_list.txt');

% load the correspondence of faces, with index of four corners in the 3D
```

```
% space
f_vert_list = dlmread('texture_files/AP_triangulated/face_vert_list.txt');

% load the correspondence of faces, with index of the four corners in the
% texture coordinate
f_tex_list = dlmread('texture_files/AP_triangulated/face_tex_list.txt');

% then load the list of texture point coordinate
tex_list = dlmread('texture_files/AP_triangulated/tex_location_list.txt');

% Convert c index to  matlab index
f_tex_list = f_tex_list + 1;
f_vert_list = f_vert_list + 1;

% Get the number of faces
f_num = size(f_tex_list, 1);

% Initiate the area vector of all the faces
area_vect = zeros(f_num, 1);
%% First get the area of the triangle
for i = 1:f_num
   % Get the index of the vertices
   ind = f_vert_list(i, :);
   % Get the 3D location of the vertices
   v_loc = vert_list(ind, :);
   % Get the area
   area_vect(i)= 1/2*norm(cross(v_loc(2, :)-v_loc(1, :), v_loc(3, :)-v_loc(1, :)));
end

%%  Get the dose reading of the three vertices in the UV map
f_reading_vect = zeros(f_num, 1);
f_cent_vect = zeros(f_num, 2);
zero_mat = zeros(3, 2);
for i = 1:f_num-7
   % Get the index of the uv vertices
   ind = f_tex_list(i, :);
   % Get the locations of the uv vertices
   uv_loc = round(max(zero_mat, tex_list(ind, :) * 1000))+1;
   reading1 = uv_map(uv_loc(1, 2), uv_loc(1, 1));
   reading2 = uv_map(uv_loc(2, 2), uv_loc(2, 1));
   reading3 = uv_map(uv_loc(3, 2), uv_loc(3, 1));
   f_reading_vect(i) = mean([reading1 reading2 reading3]);
   f_cent_vect(i, :) = mean(uv_loc, 1);
end

%% plot the coverage map to verify the zero values
```

```
zero_ind = f_reading_vect == 0;
figure(1)
imagesc(uv_map)
axis equal
hold on;
% scatter(round(tex_list(:, 1)*1000)+1, round(tex_list(:, 2)*1000)+1);
scatter(f_cent_vect(zero_ind, 1), f_cent_vect(zero_ind, 2));
hold off;
% plot the histogram
figure(2)
% The reference dose is 198.3
ref_dose = 198.3;
valid_area = area_vect(~zero_ind);
valid_dose = f_reading_vect(~zero_ind);
[histw, intervals] = histwc(valid_dose, valid_area, 40);
interval_percent = intervals / ref_dose * 100;
bar(interval_percent, histw, 'barWidth',1)
% ax = axes;
xtickformat('%g%%');
xlabel('Percentage of Prescribed Dose');
ylabel('Skin Area / cm^2');
% get the statistics
dose_mean = sum(valid_area .* valid_dose) / sum(valid_area);
dose_std = std(valid_dose, valid_area);
hold on;
mean_line=vline(dose_mean/ref_dose * 100, 'r');
set(mean_line,'linewidth',3)
% upper_line = dose_mean + dose_std;
% lower_line = dose_mean - dose_std;
% vline(lower_line, 'b');
% vline(upper_line, 'b');
% border_x = [lower_line, upper_line, upper_line, lower_line];
% border_y = [0 0 1200 1200];
% fill(border_x, border_y, [0.01 0.01 0.01]);
hold off;
% alpha(.5)
set(gca, 'FontSize', 18, 'FontWeight', 'bold', 'LineWidth', 2)
```

Appendix 4: The programming scripts in TSET treatment simulation

The code scripts are programmed for the work in the following work that is currently submitted for review.

Miao T, Zhang R, Jermyn M, Bruza P, Zhu TC, Pogue BW, Gladsone DJ, and Williams BB. Computational dose visualization & distribution comparison in total skin electron treatment illustrates superior coverage by the rotational technique

A4.1 Electron beam Monte Carlo simulation on the cylindrical phantom

The Monte Carlo simulation software (GAMOS) was used to simulate the electron beam radiation on the cylindrical phantom. The phantom was partitioned into the voxels of 1mm×1mm×1mm, and the dose value was recorded in each voxel as the output file. The scripts were in GAMOS format.

```
# File name: sd.in
# Author: Tianshun Miao and Rongxiao Zhang
# RANDOMIZE RESULTS
/gamos/random/setSeeds 1001 1001

# SET SIMULATION GLOBAL VARIABLES
/control/verbose 0
/run/verbose 0
/gamos/analysis/fileFormat CSV

# READ GEOMETRY FROM TEXT FILE
/gamos/setParam GmGeometryFromText:FileName sd.geom
/gamos/setParam GmGeometryFromText:FileNameParallel parallel_world.geom 1
/gamos/geometry GmGeometryFromText

# DECLARE BASIC EM PHYSICS PROCESSES
/gamos/physicsList GmEMPhysics
/run/initialize

#/gamos/setParam GmScoringUA:FirstEvent 1
```

```
#/gamos/setParam GmScoringUA:EachNEvent 10000
#/gamos/setParam GmPhysicsAddCerenkov:TrackSecondariesFirst 1
#/gamos/setParam      G4OpCerenkov:Method         interpolate
#/gamos/physics/addPhysics cerenkov
#/gamos/physics/addPhysics tissue-optics
/gamos/physics/addParallelProcess

# ADD SIMULATION SOURCE
/gamos/generator GmGenerator
/gamos/generator/addSingleParticleSource square_e e- 6.*MeV
/gamos/generator/positionDist square_e GmGenerDistPositionSquare 7.5*cm 0 0 50 0 0
1
/gamos/generator/directionDist square_e GmGenerDistDirectionConst 0 0 1

## LIST DIFFERENT DATA RECORDING MECHANISMS
/gamos/scoring/createMFDetector detector voxel

# DOSE SCORER
/gamos/scoring/addScorer2MFD DoseScorer GmG4PSDoseDeposit detector
/gamos/scoring/scoreErrors DoseScorer FALSE
/gamos/scoring/printer DosePrinter GmPSPrinterCSVFile
/gamos/scoring/addPrinter2Scorer DosePrinter DoseScorer
#/gamos/scoring/printByEvent DosePrinter FALSE

# CHERENKOV PHOTON NUMBER SCORER

# USER ACTIONS
/gamos/userAction GmCountTracksUA
/gamos/setParam GmCountTracksUA:EachNEvent 10000
/gamos/userAction GmCountProcessesUA

#/vis/scene/endOfEventAction accumulate 1000
#/control/execute /ihome/rzhang/gamos/GAMOS.4.0.0/examples/visVRML2FILE.in
/run/beamOn 10000000

// File name: sd.geom
//Author: Tianshun Miao and Rongxiao Zhang
// BUILD GEOMETRY

:VOLU world BOX 2500.*mm 2500.*mm 4000.*mm G4_AIR
:VIS world OFF

:ROTM RM0 0. 0. 0.
:ROTM RM1 0. 90. 0.
:VOLU inclusion TUBE 0*mm 100.*mm 100.*mm G4_WATER
:COLOUR inclusion 1. 0. 0.
```

```
:PLACE inclusion 1 world RM1 0. 0. 3150.
//---------------------------------------------------------------//
// File name: Parallel_world.geom
// Author: Tianshun Miao and Rongxiao Zhang
:VOLU container BOX 150*mm 150*mm 150*mm G4_AIR
:VOLU voxel BOX 0.5*mm 0.5*mm 0.5*mm G4_AIR
:VIS voxel OFF
:PLACE container 1 world RM0 0. 0. 3150.
:PLACE_PARAM voxel 1 container PHANTOM 300. 300. 300. 1.0 1.0 1.0
```

A4.2 Conversion from the voxeled dose distribution to the angular dose distribution

The Matlab scripts read the voxeled dose distribution from the Monte Carlo simulation and the extracted the angular dose distribution in the central regions of the cylinder. The angular dose distribution was converted to the function of the dose v.s. consine value and parameterized using a 9^{th} order polynomial function for the later treatment simulation.

```
% File name: circ_analysis.m
% Author: Tianshun Miao
clc
clear all
close all

load('dose_profile.mat');

% Plot depth dose curve
plane_profile = dose_prof;
dpp_data = mean(plane_profile(145:155, 50:150), 1);
ind = 0:1:100;
figure(1)
plot(ind, dpp_data, 'LineWidth', 2);
set(gca, 'FontSize', 14, 'FontWeight', 'Bold');
xlabel('Distance / mm');
ylabel('Dose / relative Count');

% Get circule positions
cent = 150;
radi = 50;
surface_thickness = 0:30;
radi_range = radi - surface_thickness;
radi_leng = length(radi_range);

angle_range = -180:180;
angle_leng = length(angle_range);

cir_dose_rec = zeros(radi_leng, angle_leng);
loc_x = zeros(radi_leng, angle_leng);
loc_y = zeros(radi_leng, angle_leng);

% Get dose reading for every profile
for i = 1:radi_leng
```

```
  for j =  1:angle_leng
     loc_x(i, j) = cent - radi_range(i)*cosd(angle_range(j));
     loc_y(i, j) = cent + radi_range(i)*sind(angle_range(j));
     cir_dose_rec(i, j) = interp2(plane_profile, loc_x(i, j), loc_y(i, j));
  end
end

figure(1)
cir_dose_1 = sum(cir_dose_rec(1:1, :), 1);
plot(angle_range, cir_dose_1, 'LineWidth', 2);
set(gca, 'FontSize', 14, 'FontWeight', 'Bold');
xlabel('angle / degree');
ylabel('Relative Dose count');
title('Dose Accumulate Depth 1mm');
save('cir_profile.mat', 'angle_leng', 'angle_range', 'cir_dose_rec', 'plane_profile',
'radi_leng', 'radi_range');

% File name: bin_work.m
% Author: Tianshun Miao
clc
clear all
close all

load('cir_profile.mat');
% bin the data to smooth the curve
bin_ind = -180:6:180;
bin_vect = zeros(31, 61);
bin_num = 61;
bin_vect(:, 1) = sum(cir_dose_rec(:, 358:360), 2) + ...
                 sum(cir_dose_rec(:, 1:4), 2);
bin_vect(:, end) = bin_vect(:, 1);
for i = 2:bin_num-1
   bin_vect(:, i) = sum(cir_dose_rec(:, 6*(i-1)-3:6*(i-1)+3), 2);
end

% 1: Plot dose vs angle for different depth, from 1mm, 3mm, 5mm, 10mm
depth_ind = [1, 3, 5, 10];
dose_temp = zeros(4, bin_num);
for i = 1:length(depth_ind)
   dose_prof = bin_vect(depth_ind(i), :);
   dose_temp(i, :) = dose_prof / mean(dose_prof(29:33));
end
figure(1)
plot(bin_ind, dose_temp(1, :), bin_ind, dose_temp(2, :), ...
   bin_ind, dose_temp(3, :), bin_ind, dose_temp(4, :), 'LineWidth', 2);
```

```
legend('1mm', '3mm', '5mm', '10mm');
xlabel('Angle / degree');
ylabel('Normalized Dose');
title('Normalized Angular Dose Distribution for Different Depth in Cylinder Water
Phantom');
set(gca, 'FontSize', 18, 'FontWeight', 'Bold', 'LineWidth', 2)

% 2: Acumulative Dose Curve
figure(2)
accu_dose_1_5 = sum(bin_vect(1:5, :), 1);
accu_dose_1_3 = sum(bin_vect(1:3, :), 1);
% accu_dose_1_5 = accu_dose_1_5 / mean(accu_dose_1_5(29:33));
accu_dose_5_10 = sum(bin_vect(5:10, :), 1);
% accu_dose_5_10 = accu_dose_5_10 / mean(accu_dose_5_10(29:33));
accu_dose_1_10 = sum(bin_vect(1:10, :), 1);
% accu_dose_1_10 = accu_dose_1_10 / mean(accu_dose_1_10(29:33));
plot(bin_ind, accu_dose_1_5, bin_ind, accu_dose_5_10, ...
    bin_ind, accu_dose_1_10, 'LineWidth', 2);
legend('1-5mm', '6-10mm', '1-10mm');
xlabel('Angle / degree');
ylabel('Normalized Dose');
title('Normalized Angular Acumulative Depth Dose Distribution');
set(gca, 'FontSize', 18, 'FontWeight', 'Bold', 'LineWidth', 2)

% 3: different exponential weighed curve
% Get the weight vector depending on the depth of the tissue
figure(3)
R0 = 50;   % The radius of the cylinder in mm
mu_eff = [0.15 0.175 0.2, 0.225];
w_vect = exp(- (R0 - radi_range)'* mu_eff);
che_vect = w_vect' * bin_vect;
plot(bin_ind, che_vect, 'LineWidth', 2)
legend('\mu^{eff} = 0.15 mm ^{-1}', '\mu^{eff} = 0.175 mm ^{-1}', '\mu^{eff} = 0.2
mm ^{-1}', '\mu^{eff} = 0.225 mm ^{-1}');
xlabel('Angle / degree');
ylabel('Dose Count');
title('Angular Exponentially-Weighted Depth Dose Distribution');
set(gca, 'FontSize', 18, 'FontWeight', 'Bold', 'LineWidth', 2)

figure(4)
plot(R0 - radi_range, w_vect, 'LineWidth', 2)
xlabel('Depth / mm');
ylabel('Relative Weight');
legend('\mu^{eff} = 0.15 mm ^{-1}', '\mu^{eff} = 0.175 mm ^{-1}', '\mu^{eff} = 0.2
mm ^{-1}', '\mu^{eff} = 0.225 mm ^{-1}');
title('Exponentially-Weighted Curve for Different Effective Absorption Coefficient');
```

```
set(gca, 'FontSize', 18, 'FontWeight', 'Bold', 'LineWidth', 2)

% Normalize the curve
figure(5)
che_vect_norm = che_vect;
for i = 1:4
   che_vect_norm(i, :) = che_vect(i, :) / mean(che_vect(i, 29:33));
end
plot(bin_ind, che_vect_norm', 'LineWidth', 2)
xlabel('Angle / degree');
ylabel('Relative Dose');

% take the average of the normalized
figure(6)
che_vect_mean = (mean(che_vect_norm, 1) ...
              + fliplr(mean(che_vect_norm, 1))) / 2;
dose_10_mean = (accu_dose_1_10 ...
              + fliplr(accu_dose_1_10)) / 2;
dose_3_mean = (accu_dose_1_3 ...
    + fliplr(accu_dose_1_3)) / 2;
plot(bin_ind, che_vect_mean', bin_ind, dose_3_mean, 'LineWidth', 2)
xlabel('Angle / degree');
ylabel('Relative Dose');

% Take the regions of interst and get the cosie value of it
figure(7)
che_positive = che_vect_mean(31:end);
bin_positive = bin_ind(31:end);
dose_positive = dose_3_mean(31:end);
cos_positive = cosd(bin_positive);
plot(cos_positive, che_positive, cos_positive, dose_positive, 'LineWidth', 2)
xlabel('Cosine Angle ');
ylabel('Relative Dose');
set(gca, 'FontSize', 18, 'FontWeight', 'Bold', 'LineWidth', 2)
save('ab_dose_5.mat', 'bin_ind', 'accu_dose_1_3');

% File name: bin_work.m
% Author: Tianshun Miao
clc
clear all
close all
prof_5 = load('ab_dose_5.mat');
prof_6 = load('ab_dose_6.mat');
prof_7 = load('ab_dose_7.mat');
prof_8 = load('ab_dose_8.mat');
prof_9 = load('ab_dose_9.mat');
```

```
prof_10 = load('ab_dose_10.mat');
prof_12 = load('ab_dose_12.mat');
prof_14 = load('ab_dose_14.mat');
prof_15 = load('ab_dose_15.mat');
prof_20 = load('ab_dose_20.mat');
cos_prof_5 = load('cos_prof_5.mat');
cos_prof_6 = load('cos_prof_6.mat');
cos_prof_7 = load('cos_prof_7.mat');
cos_prof_8 = load('cos_prof_8.mat');
cos_prof_9 = load('cos_prof_9.mat');
cos_prof_10 = load('cos_prof_10.mat');
cos_prof_12 = load('cos_prof_12.mat');
cos_prof_14 = load('cos_prof_14.mat');
cos_prof_15 = load('cos_prof_15.mat');
cos_prof_20 = load('cos_prof_20.mat');
bin_ind = prof_10.bin_ind;
cos_positive = cos_prof_10.cos_positive;

figure(1)
plot(bin_ind, prof_5.accu_dose_1_3,  ...
    bin_ind, prof_7.accu_dose_1_3, ...
    bin_ind, prof_9.accu_dose_1_3, ...
    bin_ind, prof_12.accu_dose_1_3, ...
    bin_ind, prof_15.accu_dose_1_3, bin_ind, prof_20.accu_dose_1_3, 'LineWidth', 3);
legend('5cm', '7cm', '9cm', '12cm', '15cm', '20cm');
xlabel('Angle / degree');
ylabel('Absolute Dose');
% title('Absolute Dose Distribution Summing from 1mm to 10mm');
set(gca, 'FontSize', 26, 'FontWeight', 'Bold', 'LineWidth', 3)
axis([-180 180 0 1.5e-12]);
% grid on;

figure(2)
plot(cos_positive,cos_prof_5.dose_positive,  ...
    cos_positive,cos_prof_7.dose_positive, ...
    cos_positive,cos_prof_9.dose_positive, ...
    cos_positive,cos_prof_12.dose_positive, ...
    cos_positive,cos_prof_15.dose_positive, cos_positive,cos_prof_20.dose_positive,
'LineWidth', 3);
legend('5cm', '7cm', '9cm',  '12cm', '15cm', '20cm');
xlabel('Cosine of Angle');
ylabel('Simulated Dose');
%title('Angular Acumulative Depth Dose Distribution from 1mm to 5mm');
set(gca, 'FontSize', 26, 'FontWeight', 'Bold', 'LineWidth', 3)

% make the fitted curve
```

```
norm_fact = 1.07e-12;
cos_val_5 = cos_prof_5.dose_positive / norm_fact;
cos_val_6 = cos_prof_6.dose_positive / norm_fact;
cos_val_7 = cos_prof_7.dose_positive / norm_fact;
cos_val_8 = cos_prof_8.dose_positive / norm_fact;
cos_val_9 = cos_prof_9.dose_positive / norm_fact;
cos_val_10 = cos_prof_10.dose_positive / norm_fact;
cos_val_12 = cos_prof_12.dose_positive / norm_fact;
cos_val_14 = cos_prof_14.dose_positive / norm_fact;
cos_val_15 = cos_prof_15.dose_positive / norm_fact;
cos_val_20 = cos_prof_20.dose_positive / norm_fact;

cos_fit_5 = fit(cos_positive',cos_val_5','poly9');
cos_fit_6 = fit(cos_positive',cos_val_6','poly9');
cos_fit_7 = fit(cos_positive',cos_val_7','poly9');
cos_fit_8 = fit(cos_positive',cos_val_8','poly9');
cos_fit_9 = fit(cos_positive',cos_val_9','poly9');
cos_fit_10 = fit(cos_positive',cos_val_10','poly9');
cos_fit_12 = fit(cos_positive',cos_val_12','poly9');
cos_fit_14 = fit(cos_positive',cos_val_14','poly9');
cos_fit_15 = fit(cos_positive',cos_val_15','poly9');
cos_fit_20 = fit(cos_positive',cos_val_20','poly9');
save('cos_curve_fit.mat', 'cos_fit_5', 'cos_fit_6', 'cos_fit_7',...
    'cos_fit_8', 'cos_fit_10', 'cos_fit_12', 'cos_fit_14', 'cos_fit_15',...
    'cos_fit_20');
```

A4.3 Extraction of the cosine value of the incidental angle to the patient's surface

The C++ script, similar to the projective texture mapping in TSET dosimetry study, was used to extract the cosine value of angle between the incidental beam and the surface normal. The values were saved for every 3D vortices in VTK format, and the format was then converted to 2D UV images using the Matlab scripts. The C++ script also used the proprietary library, which was commercially available from DoseOptics, LLC. Because the C++ script cannot handle signed datatype properly, the script was run twice for both positive and negative cosine values of each position. The conversion from the VTK format to the 2D UV image was done by the same Matlab script used in the TSET Cherenkov dosimetry study.

```
/*
        File Name: ui.h
        Author: Tianshun Miao, with help from Michael Jermyn
*/

// Prevent this header file from being included multiple times
#pragma once

// VTK header files
#include <vtkPNGReader.h>
#include <vtkInteractorStyleTrackballCamera.h>
#include <vtkActor.h>
#include <vtkImageMapToWindowLevelColors.h>
#include <vtkImageMapper.h>
#include <vtkImageMapper3D.h>
#include <vtkImageActor.h>
#include <vtkImageData.h>
#include <vtkProperty.h>
#include <vtkDICOMImageReader.h>
#include <QVTKOpenGLWidget.h>
#include <vtkGenericOpenGLRenderWindow.h>
#include <vtkSmartVolumeMapper.h>
#include <vtkNew.h>
```

```
#include <vtkVolumeProperty.h>
#include <vtkVolume.h>
#include <vtkRenderWindow.h>
#include <vtkRenderer.h>
#include <vtkPolyDataMapper.h>
#include <vtkCamera.h>
#include <vtkCoordinate.h>
#include <vtkImageFlip.h>
#include <vtkPolyData.h>
#include <vtkPNGReader.h>
#include <vtkUnsignedShortArray.h>
#include <vtkDataSet.h>
#include <vtkPointData.h>
#include <vtkProperty.h>
#include <vtkProperty2D.h>
#include <vtkOBJReader.h>
#include <vtkPolyDataMapper.h>
#include <vtkPointData.h>
#include <vtkXMLPolyDataWriter.h>
#include <vtkImageActor.h>
#include <vtkSmartPointer.h>
#include <vtkImageViewer2.h>
#include <vtkActor2D.h>

// Qt header files
#include <QPointer>
#include <QFileDialog.h>
#include <QMainWindow.h>
#include <QLayout.h>
#include <QPushButton.h>
#include <QVTKWidget.h>
#include <QSlider>
#include <QLabel>
#include <QSurfaceFormat>
#include <QStaticText>
#include <QTextEdit>

// OpenCV header files
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/objdetect.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
```

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
// Add the projection function
#include "dov_projection.h"
// Class that represents the main window for our application
class ui : public QMainWindow
{
	Q_OBJECT
private:
	// Declaration of  CLASS VARIABLES HERE
	QPushButton * but1;
	QPushButton * but2;
	QPushButton * but3;
	QPushButton * but4;
	QPushButton *bg_but, *tx_but, *obj_but;
	QTextEdit * txt_cam_angle;
	QTextEdit *bg_path, *tx_path, *obj_path;
	QTextEdit *output_file_name;
	QLabel *Text_bg, *Text_tx, *Text_obj;
	QLabel *output_file, *View_angle;
	QSlider *sl1;
	QLabel *label1;
	QVBoxLayout *layout_vertical;
	QHBoxLayout *layout_horizontal, *layout_horizontal_bg, *layout_horizontal_tx,
*layout_horizontal_obj;
	QPointer<QVTKOpenGLWidget> viewport;
	vtkNew<vtkPNGReader> img_reader, img_reader_tx;
	vtkNew<vtkGenericOpenGLRenderWindow> window1;
	QWidget *widget;
	vtkNew<vtkVolume> vol;
	vtkNew<vtkRenderer> ren1;
	vtkNew<vtkRenderer> ren2;
	vtkSmartPointer<vtkPolyData> surface;
	vtkNew<vtkOBJReader> obj_reader;
	vtkNew<vtkPolyDataMapper> mapper;
	vtkNew<vtkActor> actor;
	vtkNew<vtkActor> actor2;
	vtkNew<vtkCamera> cam1;
	vtkNew<vtkRenderWindowInteractor> renderWindowInteractor;
	vtkNew<vtkPolyDataMapper> poly_mapper;
	vtkNew<vtkInteractorStyleTrackballCamera> style;
	vtkNew<vtkImageData> image;
	vtkSmartPointer<vtkImageData> img, img_tx;
	vtkSmartPointer<vtkMapper> mpper;
```

```
        vtkNew<vtkActor2D> actor2d;
        vtkSmartPointer<vtkImageActor> img_actor;
        vtkSmartPointer<vtkImageMapper> imageMapper;
        vtkNew<vtkImageActor> imgactor;
        vtkNew<vtkCoordinate> coord;
        vtkNew<vtkImageFlip> flipYFilter;
        vtkNew<vtkImageFlip> flipZFilter;
        vtkPolyData * surf;
        vtkNew<vtkPNGReader> reader2;
        vtkRenderWindow * rendw;
        int num_p;
public:

        // Constructor (happens when created)
        ui()
        {
                // Resize the window
                this->resize(1800, 800); // CHANGE THIS
                img = vtkSmartPointer<vtkImageData>::New();
                img_tx = vtkSmartPointer<vtkImageData>::New();
                img_actor = vtkSmartPointer<vtkImageActor>::New();
                imageMapper = vtkSmartPointer<vtkImageMapper>::New();
                surface = vtkSmartPointer<vtkPolyData>::New();
                // Create the "central" (primary) widget for the window
                QWidget *widget = new QWidget();
                this->setCentralWidget(widget);

                // Create your widgets
                // Configuration of vtkopegl window

        vtkOpenGLRenderWindow::SetGlobalMaximumNumberOfMultiSamples(0);

        QSurfaceFormat::setDefaultFormat(QVTKOpenGLWidget::defaultFormat());
                viewport = new QVTKOpenGLWidget();
                viewport->SetRenderWindow(window1.Get());
                viewport->GetRenderWindow()->SetSize(800, 600);
                // configuration of button etc.
                but1 = new QPushButton("Load OBJ");
                but1->setFixedSize(150, 40);
                but2 = new QPushButton("Export Camera");
                but2->setFixedSize(150, 40);
                but3 = new QPushButton("Export Texture");
                but3->setFixedSize(150, 40);
                but4 = new QPushButton("Update Cam");
                but4->setFixedSize(150, 40);
                txt_cam_angle = new QTextEdit();
```

```
        txt_cam_angle->setFixedSize(110, 40);
        // Layout the widgets
        // YOUR CODE HERE
        layout_vertical = new QVBoxLayout();
        layout_horizontal = new QHBoxLayout();
        layout_horizontal_bg = new QHBoxLayout();
        layout_horizontal_tx = new QHBoxLayout();
        layout_horizontal_obj = new QHBoxLayout();
        widget->setLayout(layout_vertical);
        layout_vertical->addWidget(viewport);
        // the lines corresponding to read the background image file, texture image
file and obj file etc.
        layout_vertical->addLayout(layout_horizontal_bg);
        layout_vertical->addLayout(layout_horizontal_tx);
        layout_vertical->addLayout(layout_horizontal_obj);
        QFont font_label;
        font_label.setBold(true);
        font_label.setPointSize(10);
        Text_bg = new QLabel("Background Image");
        Text_bg->setFont(font_label);
        Text_tx = new QLabel("Texture Image");
        Text_tx->setFont(font_label);
        Text_obj = new QLabel("3D Obj File");
        Text_obj->setFont(font_label);
        Text_bg->setFixedSize(180, 40);
        Text_tx->setFixedSize(180, 40);
        Text_obj->setFixedSize(180, 40);
        layout_horizontal_bg->addWidget(Text_bg);
        layout_horizontal_tx->addWidget(Text_tx);
        layout_horizontal_obj->addWidget(Text_obj);
        layout_horizontal_bg->setAlignment(Qt::AlignLeft);
        layout_horizontal_tx->setAlignment(Qt::AlignLeft);
        layout_horizontal_obj->setAlignment(Qt::AlignLeft);
        // Add the text edit box to show the file path
        bg_path = new QTextEdit;
        tx_path = new QTextEdit;
        obj_path = new QTextEdit;
        bg_path->setFixedSize(800, 40);
        tx_path->setFixedSize(800, 40);
        obj_path->setFixedSize(800, 40);
        layout_horizontal_bg->addWidget(bg_path);
        layout_horizontal_tx->addWidget(tx_path);
        layout_horizontal_obj->addWidget(obj_path);
        bg_but = new QPushButton("Browse");
        tx_but = new QPushButton("Browse");
        obj_but = new QPushButton("Browse");
```

```
                bg_but->setFixedSize(150, 40);
                tx_but->setFixedSize(150, 40);
                obj_but->setFixedSize(150, 40);
                layout_horizontal_bg->addWidget(bg_but);
                layout_horizontal_tx->addWidget(tx_but);
                layout_horizontal_obj->addWidget(obj_but);
                // add output and angle of view label
                output_file = new QLabel("Output File Name");
                output_file->setFixedSize(180, 40);
                output_file->setFont(font_label);
                View_angle = new QLabel("Angle of View");
                View_angle->setFont(font_label);
                View_angle->setFixedSize(180, 40);
                layout_horizontal_bg->addSpacing(20);
                layout_horizontal_bg->addWidget(output_file);
                layout_horizontal_tx->addSpacing(20);
                layout_horizontal_tx->addWidget(View_angle);
                output_file_name = new QTextEdit;
                output_file_name->setFixedSize(110, 40);
                layout_horizontal_bg->addWidget(output_file_name);
                layout_horizontal_tx->addWidget(txt_cam_angle);
                layout_horizontal_tx->addWidget(but4);
                // Add the widge to load and do the texture
                layout_vertical->addLayout(layout_horizontal);
                layout_horizontal->addWidget(but1);
                layout_horizontal->addWidget(but2);
                layout_horizontal->addWidget(but3);
                //layout_horizontal->addWidget(txt_cam_angle);
                //layout_horizontal->addWidget(but4);
                // Configuration of the camera
                /*cam1->SetPosition(-124.736, -104.852, -654.272);
                cam1->SetFocalPoint(-12.5727, 11.141, 5.87349);
                cam1->SetViewUp(0.529516, -0.797376, 0.289491);
                cam1->SetViewAngle(2);
*/
                //cam1->SetPosition(-109.83, 80, 442.673);
                //cam1->SetFocalPoint(30.56, 83.6, 68.49);
                //cam1->SetViewUp(0.073, 0.9967, 0.0330712);   // calibration is 0.073
0.9967 0.033712
                //cam1->SetViewAngle(24.8);

                // this is for the bucket test.
                //cam1->SetPosition(45.13, -13.70, 492.5);
                //cam1->SetFocalPoint(19.12, -8.04, -1.53);
                //cam1->SetViewUp(0.03304, 0.999, 0);   // calibration is 0.073 0.9967
0.033712
```

```
		//cam1->SetViewAngle(16);

		// camera parameters for tset treatment planning
		cam1->SetPosition(0.0, 0.0, 442.4);
		cam1->SetFocalPoint(0, 0, 0);
		cam1->SetViewUp(0, 1, 0);
		cam1->SetViewAngle(25.0);

		// Configure coordinate
		coord->SetCoordinateSystemToWorld();
		rendw = viewport->GetRenderWindow();
		// Connected widget signals to slots

		connect(but1, SIGNAL(released()), this, SLOT(load_obj()));
		connect(but2, SIGNAL(released()), this, SLOT(Export_Camera()));
		connect(but3, SIGNAL(released()), this, SLOT(Export_Texture()));
		connect(but4, SIGNAL(released()), this, SLOT(Update_Cam()));
		connect(bg_but, SIGNAL(released()), this, SLOT(Load_bg()));
		connect(tx_but, SIGNAL(released()), this, SLOT(Load_tx()));
		connect(obj_but, SIGNAL(released()), this, SLOT(Load_3d()));
		// Display the window
		this->show();

	}
public slots: // This tells Qt we are defining slots here

	// YOUR CODE HERE
	// the load data code loads the dicom ct files
	void load_obj()
	{
		// load the image file in the background
		//std::string img_file;
		//img_file = "input_data/dartmouth_data/bg_pos2.PNG";
		//img_reader->SetFileName(img_file.c_str());
		img_reader->SetFileName(bg_path->toPlainText().toLocal8Bit().data());
		img_reader->Update();
		img = img_reader->GetOutput();
		imgactor->SetInputData(img);
		ren1->AddActor(imgactor);
		// load the image file as the texture
		//std::string img_tx_file;
		//img_tx_file = "input_data/dartmouth_data/che_pos2.PNG";
		//img_reader_tx->SetFileName(img_tx_file.c_str());
		img_reader_tx->SetFileName(tx_path-
>toPlainText().toLocal8Bit().data());
		img_reader_tx->Update();
```

```
		img_tx = img_reader_tx->GetOutput();
		// fill the gap in the render window
		double scale = 0.51;
		double origin[3];
		double spacing[3];
		int extent[6];
		img->GetOrigin(origin);
		img->GetSpacing(spacing);
		img->GetExtent(extent);
		vtkSmartPointer<vtkCamera> cam2 =
			ren1->GetActiveCamera(); // Use the renderer for the image in this
line
		cam2->ParallelProjectionOn();
		double xc = origin[0] + 0.5*(extent[0] + extent[1])*spacing[0];
		double yc = origin[1] + 0.5*(extent[2] + extent[3])*spacing[1];
		double yd = (extent[3] - extent[2] + 1)*spacing[1];
		double d = cam2->GetDistance();
		cam2->SetParallelScale(scale * yd);
		cam2->SetFocalPoint(xc, yc, 0.0);
		cam2->SetPosition(xc, yc, d);
		cam2->SetViewUp(0, 1, 0);
		// load the obj file
		//std::string obj_file;
		//obj_file = "input_data/dartmouth_data/model_pos2.obj";
		//obj_reader->SetFileName(obj_file.c_str());
		obj_reader->SetFileName(obj_path->toPlainText().toLocal8Bit().data());
		obj_reader->Update();
		poly_mapper->SetInputConnection(obj_reader->GetOutputPort());
		surface = obj_reader->GetOutput();
		actor->SetMapper(poly_mapper);
		ren2->AddActor(actor);
		ren2->SetActiveCamera(cam1);
		// combine renderer into render window
		ren1->SetLayer(0);
		ren1->InteractiveOff();
		ren2->SetLayer(1);
		renderWindowInteractor->SetRenderWindow(rendw);
		renderWindowInteractor->SetInteractorStyle(style);
		rendw->SetNumberOfLayers(2);
		rendw->AddRenderer(ren1);
		rendw->AddRenderer(ren2);
		rendw->Render();
		renderWindowInteractor->Start();
	}

	void Export_Camera()
```

```
	{
		ren1->Render();
		ofstream camera_data;
		camera_data.open("Camera_data.txt");
		double cam_pos1, cam_pos2, cam_pos3;
		cam1->GetPosition(cam_pos1, cam_pos2, cam_pos3);
		double cam_focal1, cam_focal2, cam_focal3;
		cam1->GetFocalPoint(cam_focal1, cam_focal2, cam_focal3);
		double cam_viewup1, cam_viewup2, cam_viewup3;
		cam1->GetViewUp(cam_viewup1, cam_viewup2, cam_viewup3);
		double cam_angle_view = cam1->GetViewAngle();
		camera_data << "Position: "  << cam_pos1 << " " << cam_pos2 << " " <<
cam_pos3 << std::endl;
		camera_data << "focal point: " << cam_focal1 << " " << cam_focal2 << "
" << cam_focal3 << std::endl;
		camera_data << "view up: " << cam_viewup1 << " " << cam_viewup2 <<
" " << cam_viewup3 << std::endl;
		camera_data << "angle of view: " << cam_angle_view << std::endl;
		camera_data.close();
	}
	void Export_Texture()
	{
		dov::projection proj;
		ren1->Render();
		vtkSmartPointer<vtkCamera> cam_test = ren1->GetActiveCamera();
		// Filp the images
		vtkSmartPointer<vtkImageFlip> flipXFilter =
			vtkSmartPointer<vtkImageFlip>::New();
		flipXFilter->SetFilteredAxis(0); // flip x axis
		flipXFilter->SetInputData(img_tx);
		flipXFilter->Update();

		vtkSmartPointer<vtkImageFlip> flipYFilter =
			vtkSmartPointer<vtkImageFlip>::New();
		flipYFilter->SetFilteredAxis(1); // flip Y axis
		flipYFilter->SetInputConnection(flipXFilter->GetOutputPort());
		flipYFilter->Update();
		vtkSmartPointer<vtkImageData> img2 = flipYFilter->GetOutput();

		proj.set_image(img2); // The image to project [vtkImageData]
		proj.set_surface(surface); // The surface [vtkPolyData]
		proj.set_camera(cam1); // The camera [vtkCamera]
		proj.cumulative = false; // If you have multi-frame image data you are
projecting, this is whether to accumulate them
		proj.visible_only = true; // Whether to only project to the first intersection
```

```
                vtkSmartPointer<vtkPolyData> surface_projected = proj.project(); //
Perform the projection, and return the output surface
                // The output surface has scalar fields "s0", "s1", etc. for each image slice
provided
                surface_projected->GetPointData()->SetActiveScalars("s0");
                // Export to the vtp xml file
                vtkSmartPointer<vtkXMLPolyDataWriter> writer =
vtkSmartPointer<vtkXMLPolyDataWriter>::New();
                QString output_name = output_file_name->toPlainText();
                output_name = output_name + ".vtp";
                writer->SetFileName(output_name.toLocal8Bit().data());
                writer->SetInputData(surface_projected);
                writer->Write();
        }
        void Update_Cam()
        {
                double cam_angle = txt_cam_angle->toPlainText().toDouble();
                cam1->SetViewAngle(cam_angle);
                rendw->Render();
                //std::cout << "The cam_angle is " << cam_angle << std::endl;
        }
        void Load_bg()
        {
                QString bg_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                bg_path->setText(bg_file_path);
        }
        void Load_tx()
        {
                QString tx_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                tx_path->setText(tx_file_path);
        }
        void Load_3d()
        {
                QString obj_file_path = QFileDialog::getOpenFileName(this,
QDir::currentPath());
                obj_path->setText(obj_file_path);
        }
};
```

A4.4 The TSET dose simulation based on the cosine value

The first part of the Matlab scripts helped the author to label the curvature radius of each part of the body. It was based on the flood-fill algorithm, in which the author manually selected one point and the script filled the area containing this point with the curvature radius value. The second part of the Matlab scripts simulated the relative dose distribution on the UV map according to the cosine values and the curvature radius. In the final part, the scripts accumulated the dose distributions from different positions, for the Stanford and rotational techniques.

```
% File name: generate_masks.m
% Author: Tianshun Miao
clc
clear all
close all
% The uv_total_mask is the regions of the UV map that is used for texture
% the 3D model
tot_mask = logical(flipud(load('uv_total_mask').uv_dist));

negative_mask = ~logical(tot_mask);
% Take regions of trunk
test_mask1 = imfill(negative_mask, [793 199; 732 631]);
region_trunk = test_mask1 - negative_mask;
% Grab regions of legs
test_mask2 = imfill(negative_mask, [262 199; 312 597]);
region_leg = test_mask2 - negative_mask;
% Grab regions of arms
test_mask3 = imfill(negative_mask, [553 231; 557 450]);
region_arm = test_mask3 - negative_mask;
% Grab regions of head
test_mask4 = imfill(negative_mask, [822 901; 811 744]);
region_head = test_mask4 - negative_mask;
% Grab regions of hands
test_mask5 = imfill(negative_mask, [267 823; 408 829]);
region_hand = test_mask5 - negative_mask;
% Grab regions of feet
test_mask6 = imfill(negative_mask, [129 866; 593 895]);
region_foot = test_mask6 - negative_mask;

% Combine uv with different cylinder radius
```

```
radius_dist = edge_padding (region_trunk * 20 + region_leg * 14 + ...
            region_arm * 9 + region_head * 14 + ...
            region_hand * 6 + region_foot * 7);
cmap = cmocean('magma', 15);
radius_uv = ind2rgb(round(radius_dist), cmap);
figure(1)
imagesc(negative_mask)
axis equal
figure(2)
imagesc(test_mask1)
axis equal
figure(3)
imshow(radius_uv)
colorbar
colormap(cmap);
caxis([0 20])
set(gca, 'FontSize', 14, 'FontWeight', 'bold');
axis equal


% File name: convert_dose.m
% Author: Tianshun Miao
clc
clear all
close all
cos_uv = load('results/LPO_pos.mat').uv_dist ...
        - load('results/LPO_neg.mat').uv_dist;
output_file_name = 'simu_dose/LPO_dose.mat';
cos_uv = edge_padding(cos_uv/10);
load('pt13_radius_label.mat');
load('cos_curve_fit.mat');
cmap = cmocean('magma', 130);
radius_dist = flipud(radius_dist);
% 1: Trunk dose, radius = 20
mask_trunk = radius_dist == 20;
dose_trunk = reshape(feval(cos_fit_20, cos_uv/100), [1000, 1000]);
dose_trunk(dose_trunk < 0) = 0;
dose_trunk_mask = mask_trunk .* dose_trunk;
dose_trunk_color = ind2rgb(round(dose_trunk_mask*100), cmap);
% 2: Leg and head dose radius = 14;
mask_leg= radius_dist == 14;
dose_leg = reshape(feval(cos_fit_14, cos_uv/100), [1000, 1000]);
dose_leg(dose_leg < 0) = 0;
dose_leg_mask = mask_leg .* dose_leg;
dose_leg_color = ind2rgb(round(dose_leg_mask*100), cmap);
% 3:arm dose radius = 9;
```

```
mask_arm= radius_dist == 9;
dose_arm = reshape(feval(cos_fit_9, cos_uv/100), [1000, 1000]);
dose_arm(dose_arm < 0) = 0;
dose_arm_mask = mask_arm .* dose_arm;
dose_arm_color = ind2rgb(round(dose_arm_mask*100), cmap);
% 4 hand dose radius = 6;
mask_hand= radius_dist == 6;
dose_hand = reshape(feval(cos_fit_6, cos_uv/100), [1000, 1000]);
dose_hand(dose_hand < 0) = 0;
dose_hand_mask = mask_hand .* dose_hand;
dose_hand_color = ind2rgb(round(dose_hand_mask*100), cmap);
% 5 Foot dose radius = 7;
mask_feet= radius_dist == 7;
dose_feet = reshape(feval(cos_fit_7, cos_uv/100), [1000, 1000]);
dose_feet(dose_feet < 0) = 0;
dose_feet_mask = mask_feet .* dose_feet;
dose_feet_color = ind2rgb(round(dose_feet_mask*100), cmap);
% Accumulate different parts of the body
dose_total = dose_trunk_mask + dose_leg_mask + dose_arm_mask + ...
            dose_hand_mask + dose_feet_mask;
dose_total_color = ind2rgb(round(dose_total*100), cmap);
imshow(dose_total_color)
save(output_file_name, 'dose_total');

% File name: accu_dose2.m (for Stanford technique)
% Author: Tianshun Miao

clc
clear all
close all

ap_uv = load('simu_dose/AP_dose.mat').dose_total;
pa_uv = load('simu_dose/PA_dose.mat').dose_total;
lao_uv = load('simu_dose/LAO_dose.mat').dose_total;
rao_uv = load('simu_dose/RAO_dose.mat').dose_total;
rpo_uv = load('simu_dose/RPO_dose.mat').dose_total;
lpo_uv = load('simu_dose/LPO_dose.mat').dose_total;

im_total = (ap_uv + pa_uv + rao_uv + lao_uv + rpo_uv + lpo_uv);
norm_val = 3.2;
im_norm = im_total / norm_val * 100;

cmap = cmocean('magma', 130);
uv_total = ind2rgb(round(im_norm), cmap);
imshow(uv_total)
c = colorbar;
```

```
colormap(cmap)
caxis([0 130])
c.Ruler.TickLabelFormat='%g%%';
set(gca, 'Fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold');
```

```
% File name: accu_dose_simu.m (for rotational technique)
% Author: Tianshun Miao
clc
clear all
close all

uv_mat = zeros(1000, 1000, 60);
for i = 0:59
    file_name = strcat('simu_dose/Pt13_dose_rot_', num2str(i*6), '.mat');
    uv_mat(:, :, i+1) = load(file_name).dose_total;
end
imtotal = sum(uv_mat, 3)/30 * 100;
% imtotal_norm = imtotal / max(imtotal(:));
% imtotal2 = flipud(imtotal);
cmap = cmocean('magma', 130);
uv_total = ind2rgb(round(imtotal), cmap);
imshow(uv_total)
% plot color map
c = colorbar;
colormap(cmap)
caxis([0 130])
c.Ruler.TickLabelFormat='%g%%';
set(gca, 'Fontweight', 'bold', 'fontsize', 20);
```

A4.5 Statistical analysis of the area dose distribution

The treatment simulation work also conducted the statistical analysis similar to the TSET Cherenkov dosimetry study. The Matlab scripts generated the similar area dose distribution analysis. In addition, the scripts conducted the DAH analysis. The scripts also plotted the comparisons between the different techniques and the different patients.

```
% File name: area_analysis.m
% Author: Tianshun Miao

clc
clear all
close all

% Load and covert the UV map
uv_map = flipud(load('pt13_rot_total_dose.mat').imtotal / 100);
% load the positon of the obj vertex
vert_list = dlmread('AP_triangulated/vert_location_list.txt');

% load the correspondence of faces, with index of four corners in the 3D
% space
f_vert_list = dlmread('AP_triangulated/face_vert_list.txt');

% load the correspondence of faces, with index of the four corners in the
% texture coordinate
f_tex_list = dlmread('AP_triangulated/face_tex_list.txt');

% then load the list of texture point coordinate
tex_list = dlmread('AP_triangulated/tex_location_list.txt');

% Convert c index to  matlab index
f_tex_list = f_tex_list + 1;
f_vert_list = f_vert_list + 1;

% Get the number of faces
f_num = size(f_tex_list, 1);

% Initiate the area vector of all the faces
area_vect = zeros(f_num, 1);
%% First get the area of the triangle
for i = 1:f_num
    % Get the index of the vertices
    ind = f_vert_list(i, :);
```

```
    % Get the 3D location of the vertices
    v_loc = vert_list(ind, :);
    % Get the area
    area_vect(i)= 1/2*norm(cross(v_loc(2, :)-v_loc(1, :), v_loc(3, :)-v_loc(1, :)));
end

%%  Get the dose reading of the three vertices in the UV map
f_reading_vect = zeros(f_num, 1);
f_cent_vect = zeros(f_num, 2);
zero_mat = zeros(3, 2);
for i = 1:f_num-7
    % Get the index of the uv vertices
    ind = f_tex_list(i, :);
    % Get the locations of the uv vertices
    uv_loc = min(round(max(zero_mat, tex_list(ind, :) * 1000))+1, 1000);
    reading1 = uv_map(uv_loc(1, 2), uv_loc(1, 1));
    reading2 = uv_map(uv_loc(2, 2), uv_loc(2, 1));
    reading3 = uv_map(uv_loc(3, 2), uv_loc(3, 1));
    f_reading_vect(i) = mean([reading1 reading2 reading3]);
    f_cent_vect(i, :) = mean(uv_loc, 1);
end

%% plot the coverage map to verify the zero values
zero_ind = f_reading_vect == 0;
figure(1)
imagesc(uv_map)
axis equal
hold on;
% scatter(round(tex_list(:, 1)*1000)+1, round(tex_list(:, 2)*1000)+1);
scatter(f_cent_vect(zero_ind, 1), f_cent_vect(zero_ind, 2));
hold off;
% plot the histogram
figure(2)
% The reference dose is 185
ref_dose = 1;
valid_area = area_vect(~zero_ind);
valid_dose = f_reading_vect(~zero_ind);
[histw, intervals] = histwc(valid_dose, valid_area, 40);
interval_percent = intervals / ref_dose;
area_total = sum(histw);
bar(interval_percent*100, histw/area_total*100, 'barWidth',1)
% ax = axes;
xtickformat('%g%%');
ytickformat('%g%%');
xlabel('Percentage of Prescribed Dose');
ylabel('Skin Area percentage');
```

```
% get the statistics
dose_mean = sum(valid_area .* valid_dose) / sum(valid_area);
dose_std = std(valid_dose, valid_area);
hold on;
mean_line=vline(dose_mean/ref_dose*100, 'r');
set(mean_line,'linewidth',3)
hold off;
axis([0 120 0 30])
% alpha(.5)
set(gca, 'FontSize', 26, 'FontWeight', 'bold', 'LineWidth', 3)

%% Calculate DAH (instead of DVH)
dose_index = interval_percent;
bin_num = size(dose_index, 2);
area_val = histw;
total_area = sum(area_val);
area_percent = area_val/total_area * 100;
accu_area = zeros(bin_num, 1);
accu_area(1) = 100 - area_percent(1);
for i = 2:bin_num
   accu_area(i) = accu_area(i-1) - area_percent(i);
end
figure(3)
bar(interval_percent*100, accu_area, 'barWidth',1)
xtickformat('%g%%');
xlabel('Percentage of Prescribed Dose');
ytickformat('%g%%');
ylabel('Skin Area Percentage');
set(gca, 'FontSize', 18, 'FontWeight', 'bold', 'LineWidth', 2)
title('Plot of DAH')

figure(4)
% bar(interval_percent*100, histw/total_area*100, 'barWidth',1)
plot(interval_percent*100, accu_area, 'Linewidth', 2);
% ax = axes;
xtickformat('%g%%');
xlabel('Percentage of Prescribed Dose');
ytickformat('%g%%');
ylabel('Skin Area percentage');
% get the statistics
dose_mean = sum(valid_area .* valid_dose) / sum(valid_area);
dose_std = std(valid_dose, valid_area);
hold on;
mean_line=vline(dose_mean/ref_dose*100, 'r');
set(mean_line,'linewidth',3)
```

```
hold off;
% alpha(.5)
set(gca, 'FontSize', 18, 'FontWeight', 'bold', 'LineWidth', 2)
save('pt13_rot_dah.mat', 'accu_area', 'interval_percent');

% File name: area_dose_comp.m
% Author: Tianshun Miao
clc
clear all
close all
% Pt 6 rot and stanford
pt_6_rot_interval_percent = load('pt6_rot_dah.mat').interval_percent;
pt_6_stanford_interval_percent = load('pt6_stanford_dah.mat').interval_percent;
pt_6_rot_dah = load('pt6_rot_dah.mat').accu_area;
pt_6_stanford_dah = load('pt6_stanford_dah.mat').accu_area;
% Pt 13 rot and stanford
pt_13_rot_interval_percent = load('pt13_rot_dah.mat').interval_percent;
pt_13_stanford_interval_percent = load('pt13_stanford_dah.mat').interval_percent;
% pt_13_stanford_overrotate_interval_percent =
load('pt13_stanford_overrotate_dah.mat').interval_percent;
pt_13_rot_dah = load('pt13_rot_dah.mat').accu_area;
pt_13_stanford_dah = load('pt13_stanford_dah.mat').accu_area;
% pt_13_stanford_overrotate_dah = load('pt13_stanford_overrotate_dah.mat').accu_area;


plot(pt_6_rot_interval_percent*100, pt_6_rot_dah,...
   pt_13_rot_interval_percent*100, pt_13_rot_dah, ...
   'Linewidth', 4, 'LineStyle', '--');
hold on;
plot(pt_6_stanford_interval_percent*100, pt_6_stanford_dah,...
   'Linewidth', 4, 'Color', '#0072BD');
plot(pt_13_stanford_interval_percent*100, pt_13_stanford_dah,...
   'Linewidth', 4, 'Color', '#D95319');
% plot(pt_13_stanford_overrotate_interval_percent*100,
pt_13_stanford_overrotate_dah,...
%     'Linewidth', 4, 'Color', '#D95319', 'LineStyle', ':');
hold off;
legend('Low Body Mass Rotational', 'High Body Mass Rotational',...
   'Low Body Mass Stanford', 'High Body Mass Stanford');
xtickformat('%g%%');
xlabel('Percentage of Prescribed Dose');
ytickformat('%g%%');
ylabel('Skin Area percentage');
set(gca, 'FontSize', 24, 'FontWeight', 'bold', 'LineWidth', 3)
axis([0 130 0 100]);
```

Reference

1. Khan FM, Gibbons JP. *Khan's the physics of radiation therapy.* Lippincott Williams & Wilkins; 2014.
2. Klein EE, Hanley J, Bayouth J, et al. Task Group 142 report: Quality assurance of medical accelerators. *Med Phys.* 2009;36(9Part1):4197-4212.
3. Almond PR, Biggs PJ, Coursey BM, et al. AAPM's TG-51 protocol for clinical reference dosimetry of high-energy photon and electron beams. *Med Phys.* 1999;26(9):1847-1870.
4. Tailor RC, Hanson WF, Ibbott GS. TG-51: Experience from 150 institutions, common errors, and helpful hints. *J Appl Clin Med Phys.* 2003;4(2):102-111.
5. Arjomandy B, Tailor R, Zhao L, et al. EBT2 film as a depth-dose measurement tool for radiotherapy beams over a wide range of energies and modalities. *Med Phys.* 2012;39(2):912-921.
6. Das IJ, Cheng CW, Watts RJ, et al. Accelerator beam data commissioning equipment and procedures: report of the TG-106 of the Therapy Physics Committee of the AAPM. *Med Phys.* 2008;35(9):4186-4215.
7. Chiu-Tsao ST, Chan MF. Photon beam dosimetry in the superficial buildup region using radiochromic EBT film stack. *Med Phys.* 2009;36(6Part1):2074-2083.
8. Dobler B, Streck N, Klein E, et al. Hybrid plan verification for intensity-modulated radiation therapy (IMRT) using the 2D ionization chamber array I'mRT MatriXX—a feasibility study. *Phys Med Biol.* 2010;55(2):N39-N55.
9. Gonzalez A, Castro I, Martínez J. A procedure to determine the radiation isocenter size in a linear accelerator. *Med Phys.* 2004;31(6):1489-1493.

10. Depuydt T, Penne R, Verellen D, et al. Computer-aided analysis of star shot films for high-accuracy radiation therapy treatment units. *Phys Med Biol.* 2012;57(10):2997.
11. García-Garduño OA, Celis MÁ, Lárraga-Gutiérrez JM, et al. Radiation transmission, leakage and beam penumbra measurements of a micro-multileaf collimator using GafChromic EBT film. *J Appl Clin Med Phys.* 2008;9(3):90-98.
12. Rowshanfarzad P, Sabet M, Barnes MP, et al. EPID-based verification of the MLC performance for dynamic IMRT and VMAT. *Med Phys.* 2012;39(10):6192-6207.
13. Richart J, Pujades M, Perez-Calatayud J, et al. QA of dynamic MLC based on EPID portal dosimetry. *Phys Med.* 2012;28(3):262-268.
14. Woods K, Rong Y. Technical report: TG-142 compliant and comprehensive quality assurance tests for respiratory gating. *Med Phys.* 2015;42(11):6488-6497.
15. Alexander DA, Zhang R, Brůža P, et al. Scintillation imaging as a high-resolution, remote, versatile 2D detection system for MR-linac quality assurance. *Med Phys.* 2020.
16. Girardi M, Heald PW, Wilson LD. The pathogenesis of mycosis fungoides. *N Engl J Med.* 2004;350(19):1978-1988.
17. Diamandidou E, Cohen PR, Kurzrock R. Mycosis fungoides and Sezary syndrome. *Blood.* 1996;88(7):2385-2409.
18. Piotrowski T, Milecki P, Skórska M, et al. Total skin electron irradiation techniques: a review. *Advances in Dermatology and Allergology/Postępy Dermatologii I Alergologii.* 2013;30(1):50.

19. Karzmark C. Total skin electron therapy: technique and dosimetry. *International Journal of Radiation Oncology• Biology• Physics.* 1986;12:84-85.
20. Xie Y, Petroccia H, Maity A, et al. *Cherenkov imaging for Total Skin Electron Therapy (TSET).* Vol 10478: SPIE; 2018.
21. Xie Y, Petroccia H, Maity A, et al. Cherenkov imaging for total skin electron therapy (TSET). *Med Phys.* 2020;47(1):201-212.
22. Petroccia H, Miao T, Maity A, et al. *Analysis of cumulative surface dose based on Cherenkov imaging of Total Skin Electron Therapy (TSET).* Vol 10860: SPIE; 2019.
23. Parida DK, Verma KK, Chander S, et al. Total skin electron irradiation therapy in mycosis fungoides using high-dose rate mode: A preliminary experience. *Int J Dermatol.* 2005;44(10):828-830.
24. Marinello G, Jaffre F, Slosarek K, et al. Total skin electron irradiation. *Reports of Practical Oncology & Radiotherapy.* 1998;3(1):19-22.
25. Evans MD, Hudon C, Podgorsak EB, et al. Institutional experience with a rotational total skin electron irradiation (RTSEI) technique—a three decade review (1981–2012). *Reports of Practical Oncology & Radiotherapy.* 2014;19(2):120-134.
26. Hensley FW, Major G, Edel C, et al. Technical and dosimetric aspects of the total skin electron beam technique implemented at Heidelberg University Hospital. *Rep Pract Oncol Radiother.* 2013;19(2):135-143.
27. Cherenkov PA. Visible emission of clean liquids by action of γ radiation. Paper presented at: Dokl. Akad. Nauk SSSR1934.

28. Zhang R, Colleen JF, Adam KG, et al. Superficial dosimetry imaging of Čerenkov emission in electron beam radiotherapy of phantoms. *Phys Med Biol.* 2013;58(16):5477.

29. Glaser AK, Voigt WHA, Davis SC, et al. Three-dimensional Cerenkov tomography of energy deposition from ionizing radiation beams. *Opt Lett.* 2013;38(5):634-636.

30. Arkani M, Gharib M. Reactor core power measurement using Cherenkov radiation and its application in Tehran Research Reactor. *Annals of Nuclear Energy.* 2009;36(7):896-900.

31. De Vries K, van Den Berg A, Scholten O, et al. Coherent Cherenkov radiation from cosmic-ray-induced air showers. *Phys Rev Lett.* 2011;107(6):061101.

32. Doro M, Conrad J, Emmanoulopoulos D, et al. Dark matter and fundamental physics with the Cherenkov Telescope Array. *Astroparticle Physics.* 2013;43:189-214.

33. Pogue BW, Zhang R, Glaser A, et al. Cherenkov imaging in the potential roles of radiotherapy QA and delivery. *Journal of Physics: Conference Series.* 2017;847(1):012046.

34. Zhang R, Gladstone DJ, Jarvis LA, et al. Real-time in vivo Cherenkoscopy imaging during external beam radiation therapy. 2013.

35. Andreozzi JM, Zhang R, Glaser AK, et al. Camera selection for real-time in vivo radiation treatment verification systems using Cherenkov imaging. *Med Phys.* 2015;42(2):994-1004.

36. Glaser AK, Andreozzi JM, Zhang R, et al. Optical cone beam tomography of Cherenkov-mediated signals for fast 3D dosimetry of x-ray photon beams in water. *Med Phys.* 2015;42(7):4127-4136.

37. Bruza P, Andreozzi JM, Gladstone DJ, et al. Real-time 3D dose imaging in water phantoms: reconstruction from simultaneous EPID-Cherenkov 3D imaging (EC3D). *Journal of Physics: Conference Series.* 2017;847(1):012034.

38. Jarvis LA, Zhang R, Gladstone DJ, et al. Cherenkov Video Imaging Allows for the First Visualization of Radiation Therapy in Real Time. *International Journal of Radiation Oncology*Biology*Physics.* 2014;89(3):615-622.

39. Pogue BW, Hachadorian RL, Garg S, et al. Cherenkov emission from external beam irradiation: proportional to the dose buildup gradient and inversely affected by tissue optical attenuation. Paper presented at: Optics and Ionizing Radiation2020.

40. Jarvis LA, Gladstone DJ, Pogue BW, et al. Cherenkoscopy for Treatment Verification: Correlation of Radiation Dose to Cherenkov Emission Intenisty in Whole Breast Radiation Therapy. *Int J Radiat Oncol Biol Phys.* 2017;99(2):E673.

41. Andreozzi JM, Zhang R, Gladstone DJ, et al. Cherenkov imaging method for rapid optimization of clinical treatment geometry in total skin electron beam therapy. *Med Phys.* 2016;43(2):993-1002.

42. Hachadorian RL, Bruza P, Jermyn M, et al. *Correcting Cherenkov images for large-scale tissue-optical property attenuation using SFDI and patterned light reflectance for quantitative dosimetry.* Vol 10874: SPIE; 2019.

43. Hachadorian R, Bruza P, Jermyn M, et al. Imaging Radiation Dose in Breast Radiotherapy by X-ray CT Calibration of Cherenkov Light. *Nature Communications.* 2020;In Press.
44. Wang LV, Wu H-i. *Biomedical optics: principles and imaging.* John Wiley & Sons; 2012.
45. Meng S, Kaxiras E. Mechanisms for ultrafast nonradiative relaxation in electronically excited eumelanin constituents. *Biophys J.* 2008;95(9):4396-4402.
46. Köhler A, Wilson JS, Friend RH. Fluorescence and phosphorescence in organic materials. *Advanced Materials.* 2002;14(10):701-707.
47. Wang S-Y. *Photochemistry and photobiology of nucleic acids.* Vol 1: Elsevier; 2012.
48. Long DA. Raman spectroscopy. *New York.* 1977:1-12.
49. Wang H-W, Zhu TC, Putt MP, et al. Broadband reflectance measurements of light penetration, blood oxygenation, hemoglobin concentration, and drug concentration in human intraperitoneal tissues before and after photodynamic therapy. *J Biomed Opt.* 2005;10(1):014004.
50. Zijlstra W, Buursma A. Spectrophotometry of hemoglobin: absorption spectra of bovine oxyhemoglobin, deoxyhemoglobin, carboxyhemoglobin, and methemoglobin. *Comparative Biochemistry and Physiology Part B: Biochemistry and Molecular Biology.* 1997;118(4):743-749.
51. Quickenden T, Irvin J. The ultraviolet absorption spectrum of liquid water. *The Journal of Chemical Physics.* 1980;72(8):4416-4428.
52. Srinivasan S, Pogue BW, Jiang S, et al. Interpreting hemoglobin and water concentration, oxygen saturation, and scattering measured in vivo by near-

infrared breast tomography. *Proceedings of the National Academy of Sciences.* 2003;100(21):12349-12354.

53. Meyers DE, Anderson LD, Seifert RP, et al. Noninvasive method for measuring local hemoglobin oxygen saturation in tissue using wide gap second derivative near-infrared spectroscopy. *J Biomed Opt.* 2005;10(3):034017.
54. Axelsson J, Glaser AK, Gladstone DJ, et al. Quantitative Cherenkov emission spectroscopy for tissue oxygenation assessment. *Opt Express.* 2012;20(5):5133-5142.
55. Matcher S, Cope M, Delpy D. In vivo measurements of the wavelength dependence of tissue-scattering coefficients between 760 and 900 nm measured with time-resolved spectroscopy. *Appl Opt.* 1997;36(1):386-396.
56. Hergert W, Wriedt T. *The Mie theory: basics and applications.* Vol 169: Springer; 2012.
57. Bucholtz A. Rayleigh-scattering calculations for the terrestrial atmosphere. *Appl Opt.* 1995;34(15):2765-2773.
58. Patterson MS, Wilson BC, Wyman DR. The propagation of optical radiation in tissue I. Models of radiation transport and their application. *Lasers Med Sci.* 1991;6(2):155-168.
59. Pogue BW, Patterson MS. Frequency-domain optical absorption spectroscopy of finite tissue volumes using diffusion theory. *Phys Med Biol.* 1994;39(7):1157.
60. Pharr M, Jakob W, Humphreys G. *Physically based rendering: From theory to implementation.* Morgan Kaufmann; 2016.

61. Toublanc D. Henyey–Greenstein and Mie phase functions in Monte Carlo radiative transfer computations. *Appl Opt.* 1996;35(18):3270-3274.

62. Miao T, Petroccia H, Xie Y, et al. Computer animation body surface analysis of total skin electron radiation therapy dose homogeneity via Cherenkov imaging. *Journal of Medical Imaging.* 2020;7(3):034002.

63. Glaser AK, Zhang R, Andreozzi JM, et al. Cherenkov radiation fluence estimates in tissue for molecular imaging and therapy applications. *Phys Med Biol.* 2015;60(17):6701.

64. Dsouza A, Lin H, Gunn JR, et al. Cherenkov-excited multi-fluorophore sensing in tissue-simulating phantoms and in vivo from external beam radiotherapy. *Radiat Res.* 2018;189(2):197-204.

65. Miao T, Bruza P, Pogue BW, et al. Cherenkov imaging for linac beam shape analysis as a remote electronic quality assessment verification tool. *Med Phys.* 2019;46(2):811-821.

66. Glaser AK, Kanick SC, Zhang R, et al. A GAMOS plug-in for GEANT4 based Monte Carlo simulation of radiation-induced light transport in biological media. *Biomedical optics express.* 2013;4(5):741-759.

67. Bushberg JT, Boone JM. *The essential physics of medical imaging.* Lippincott Williams & Wilkins; 2011.

68. Kim J, Avants B, Patel S, et al. Structural consequences of diffuse traumatic brain injury: a large deformation tensor-based morphometry study. *Neuroimage.* 2008;39(3):1014-1026.

69. Haripotepornkul NH, Nath SK, Scanderbeg D, et al. Evaluation of intra-and inter-fraction movement of the cervix during intensity modulated radiation therapy. *Radiother Oncol.* 2011;98(3):347-351.

70. Tanguturi SK, Lyatskaya Y, Chen Y, et al. Prospective assessment of deep inspiration breath-hold using 3-dimensional surface tracking for irradiation of left-sided breast cancer. *Pract Radiat Oncol.* 2015;5(6):358-365.

71. Lagendijk JJ, Raaymakers BW, Raaijmakers AJ, et al. MRI/linac integration. *Radiother Oncol.* 2008;86(1):25-29.

72. O'Rourke M. *Principles of three-dimensional computer animation: modeling, rendering, and animating with 3D computer graphics.* WW Norton & Company; 2003.

73. Burtnyk N, Wein M. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communications of the ACM.* 1976;19(10):564-569.

74. Liu P-C, Wu F-C, Ma W-C, et al. Automatic animation skeleton using repulsive force field. Paper presented at: 11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.2003.

75. Wang XC, Phillips C. Multi-weight enveloping: least-squares approximation techniques for skin animation. Paper presented at: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation2002.

76. Heckbert PS. *Fundamentals of texture mapping and image warping.* EECS Department, University of California, Berkeley;1989.

77. Hartley R, Zisserman A. *Multiple view geometry in computer vision.* Cambridge university press; 2003.

78. Zhang Q, Pless R. Extrinsic calibration of a camera and laser range finder (improves camera calibration). Paper presented at: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)2004.

79. Liang Z-m, Gao H-m, Wang Z-j, et al. Sub-pixels corner detection for camera calibration. *TRANSACTIONS-CHINA WELDING INSTITUTION.* 2006;27(2):102.

80. Bradski G, Kaehler A. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc."; 2008.

81. Fetić A, Jurić D, Osmanković D. The procedure of a camera calibration using Camera Calibration Toolbox for MATLAB. Paper presented at: 2012 Proceedings of the 35th International Convention MIPRO2012.

82. Nath R, Biggs PJ, Bova FJ, et al. AAPM code of practice for radiotherapy accelerators: Report of AAPM Radiation Therapy Task Group No. 45. *Med Phys.* 1994;21(7):1093-1121.

83. Kutcher GJ, Coia L, Gillin M, et al. Comprehensive QA for radiation oncology: Report of AAPM Radiation Therapy Committee Task Group 40. *Med Phys.* 1994;21(4):581-618.

84. American Association of Physicists in Medicine. *Basic quality control in diagnostic radiology.* American Association of Physicists in Medicine; 1978.

85. Xu LB. *Commissioning of a GafChromic EBT Film Dosimetry Protocol at Ionizing Radiation Standards Group of National Research Council* [M.S. thesis], McGill University, Montreal; 2009.

86. Bouchard H, Seuntjens J. Ionization chamber-based reference dosimetry of intensity modulated radiation beams. *Med Phys.* 2004;31(9):2454-2465.

87. Sun B, Yaddanapudi S, Goddu SM, et al. A self-sufficient method for calibration of Varian electronic portal imaging device. *Journal of Physics: Conference Series.* 2015;573(1):012041.

88. Létourneau D, Gulam M, Yan D, et al. Evaluation of a 2D diode array for IMRT quality assurance. *Radiother Oncol.* 2004;70(2):199-206.

89. Nilsson B, Rudén B-I, Sorcini B. Characteristics of silicon diodes as patient dosemeters in external radiation therapy. *Radiother Oncol.* 1988;11(3):279-288.

90. Leybovich LB, Sethi A, Dogan N. Comparison of ionization chambers of various volumes for IMRT absolute dose verification. *Med Phys.* 2003;30(2):119-123.

91. Poppe B, Blechschmidt A, Djouguela A, et al. Two-dimensional ionization chamber arrays for IMRT plan verification. *Med Phys.* 2006;33(4):1005-1015.

92. Sotiri S. Ionization chamber array for patient specific VMAT, Tomotherapy and IMRT QA. *Journal of Physics: Conference Series.* 2010;250(1):012029.

93. Van Esch A, Basta K, Evrard M, et al. The Octavius1500 2D ion chamber array and its associated phantoms: Dosimetric characterization of a new prototype. *Med Phys.* 2014;41(9):091708.

94. Snyder C, Pogue BW, Jermyn M, et al. Algorithm development for intrafraction radiotherapy beam edge verification from Cherenkov imaging. 2018.

95. Zhang R, Gladstone DJ, Jarvis LA, et al. Real-time in vivo Cherenkoscopy imaging during external beam radiation therapy. *J Biomed Opt.* 2013;18(11):110504.

96. Andreozzi JM, Mooney KE, Brůža P, et al. Remote Cherenkov imaging-based quality assurance of a magnetic resonance image-guided radiotherapy system. *Med Phys.* 2018;45(6):2647-2659.

97. Black PJ, Velten C, Wang YF, et al. An Investigation of Clinical Treatment Field Delivery Verification Using Cherenkov Imaging: IMRT Positioning Shifts and Field Matching. *Med Phys.* 2018.

98. Harris C, Stephens M. A combined corner and edge detector. Paper presented at: Alvey Vision Conference1988.

99. Seibert JA, Boone JM, Lindfors KK. Flat-field correction technique for digital detectors. Paper presented at: Medical Imaging 1998: Physics of Medical Imaging1998.

100. González A, Castro I, Martínez JA. A procedure to determine the radiation isocenter size in a linear accelerator. *Med Phys.* 2004;31(6):1489-1493.

101. Depuydt T, Penne R, Verellen D, et al. Computer-aided analysis of star shot films for high-accuracy radiation therapy treatment units. *Phys Med Biol.* 2012;57(10):2997.

102. Karzmark C, Loevinger R, Steele R, et al. A technique for large-field, superficial electron therapy. *Radiology.* 1960;74(4):633-644.

103. Jones GW, Kacinski BM, Wilson LD, et al. Total skin electron radiation in the management of mycosis fungoides: consensus of the European Organization for Research and Treatment of Cancer (EORTC) Cutaneous Lymphoma Project Group. *J Am Acad Dermatol.* 2002;47(3):364-370.

104. Harrison C, Young J, Navi D, et al. Revisiting low-dose total skin electron beam therapy in mycosis fungoides. *International Journal of Radiation Oncology• Biology• Physics.* 2011;81(4):e651-e657.

105. El-Khatib E, Hussein S, Nikolic M, et al. Variation of electron beam uniformity with beam angulation and scatterer position for total skin irradiation with the Stanford technique. *International Journal of Radiation Oncology• Biology• Physics.* 1995;33(2):469-474.

106. Navi D, Riaz N, Levin YS, et al. The Stanford University experience with conventional-dose, total skin electron-beam therapy in the treatment of generalized patch or plaque (T2) and tumor (T3) mycosis fungoides. *Arch Dermatol.* 2011;147(5):561-567.

107. Hensley FW, Major G, Edel C, et al. Technical and dosimetric aspects of the total skin electron beam technique implemented at Heidelberg University Hospital. *Reports of Practical Oncology & Radiotherapy.* 2014;19(2):135-143.

108. Guidi G, Gottardi G, Ceroni P, et al. Review of the results of the in vivo dosimetry during total skin electron beam therapy. *Reports of Practical Oncology & Radiotherapy.* 2014;19(2):144-150.

109. Alderliesten T, Sonke J-J, Betgen A, et al. Accuracy Evaluation of a 3-Dimensional Surface Imaging System for Guidance in Deep-Inspiration Breath-Hold Radiation Therapy. *International Journal of Radiation Oncology*Biology*Physics.* 2013;85(2):536-542.

110. Dey D, Gobbi DG, Slomka PJ, et al. Automatic fusion of freehand endoscopic brain images to three-dimensional surfaces: creating stereoscopic panoramas. *IEEE Trans Med Imaging.* 2002;21(1):23-30.

111. Aspert N, Santa-Cruz D, Ebrahimi T. MESH: measuring errors between surfaces using the Hausdorff distance. Paper presented at: Proceedings. IEEE International Conference on Multimedia and Expo; 26-29 Aug. 2002, 2002.

112. Heckbert PS. Survey of Texture Mapping. *IEEE Computer Graphics and Applications.* 1986;6(11):56-67.

113. Karzmack C, Anderson J, Fessenden P, et al. AAPM report No. 23, total skin electron therapy: technique and dosimetry. *Report of group.* 1987;30.

114. Archambault L, Arsenault J, Gingras L, et al. Plastic scintillation dosimetry: Optimal selection of scintillating fibers and scintillators. *Med Phys.* 2005;32(7Part1):2271-2278.

115. Alexander DA, Tendler II, Bruza P, et al. Assessment of imaging Cherenkov and scintillation signals in head and neck radiotherapy. *Phys Med Biol.* 2019;64(14):145021.

116. Chetty IJ, Curran B, Cygler JE, et al. Report of the AAPM Task Group No. 105: Issues associated with clinical implementation of Monte Carlo-based photon and electron external beam treatment planning. *Med Phys.* 2007;34(12):4818-4853.

117. Arce P, Rato P, Canadas M, et al. GAMOS: A Geant4-based easy and flexible framework for nuclear medicine applications. Paper presented at: IEEE Nuclear Science Symposium Conference Record2008.

118. Ye SJ, Pareek PN, Spencer S, et al. Monte Carlo techniques for scattering foil design and dosimetry in total skin electron irradiations. *Med Phys.* 2005;32(6):1460-1468.

119. Bjärngard BE, Chen GT, Piontek RW, et al. Analysis of dose distributions in whole body superficial electron therapy. *International Journal of Radiation Oncology* Biology* Physics.* 1977;2(3-4):319-324.

120. Graham MV, Purdy JA, Emami B, et al. Clinical dose–volume histogram analysis for pneumonitis after 3D treatment for non-small cell lung cancer (NSCLC). *International Journal of Radiation Oncology* Biology* Physics.* 1999;45(2):323-329.

121. Gerbi BJ, Antolak JA, Deibel FC, et al. Recommendations for clinical electron beam dosimetry: Supplement to the recommendations of Task Group 25. *Med Phys.* 2009;36(7):3239-3279.

122. Laaksomaa M, Sarudis S, Rossi M, et al. AlignRT® and Catalyst™ in whole-breast radiotherapy with DIBH: Is IGRT still needed? *J Appl Clin Med Phys.* 2019;20(3):97-104.

123. Miao T, Petroccia H, Xie Y, et al. Computer animation body surface analysis of total skin electron radiation therapy dose homogeneity via Cherenkov imaging. *J MedImaging.* 2020;7(3):034002.

124. Thyng KM, Greene CA, Hetland RD, et al. True colors of oceanography: Guidelines for effective and accurate colormap selection. *Oceanography.* 2016;29(3):9-13.