Simulation Procedure for Marker and Camera Placement

iThemba LABS &

University of Stellenbosch

Andre van der Merwe

December 10, 2003

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Engineering Science
at
Stellenbosch University.

Promoter: Doctor N. Muller

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Medical Radiation department at iThemba LABS[1] provides proton beam therapy facilities for irradiation of intracranial, head and neck lesions. Proton radiation treatment offers a number of advantages over alternative radiation therapy modalities. The most significant advantage is the ability to localize the dose to the lesion or target volume [16]. Lesions are located by means of medical imaging processes, such as Computer Tomography (CT) or Magnetic Resonance Imaging (MRI) scans. Patient treatment commences at the existing treatment facility of iThemba LABS. The patient positioning system that is currently in use at this facility was designed for only one horizontal beam delivery system and a limited number of treatment positions. The possibility of acquiring an additional beam delivery system and improving the utilization of the system resulted in plans to expand the current proton therapy capabilities. These plans resulted in the development of a new treatment vault, complete with a new patient positioning system. The new vault will cater for two beam delivery systems and expand current treatment positions. For a description of the current proton treatment facilities, refer to [2] and [24].

The patient positioning system (PPS) consists of a patient positioner, a patient alignment system and a digital radiograph system. The patient positioner is a

[1]iThemba LABS, P.O.Box 722, Somerset West 7129, South Africa.

commercial robot manipulator, fitted with either a chair or a couch as a patient support system. This robot will be responsible for mechanically moving the patient to the required treatment positions. The patient alignment system will be used to derive the movement path required by the patient positioner. The digital radiograph system will be used to determine the small corrections that might be needed to compensate for possible misalignment of the patient. In the first stage of patient treatment, the computer tomography (CT) scanner produces slices of x-ray images through a region of the patient. These x-ray images are combined to create a three-dimensional (3D) model of the patient's anatomy in that region. This model is used to construct a treatment plan for the treatment of the patient. In the second stage of patient treatment, this information is passed on to a computerized treatment planning system. This system uses this treatment plan to compute the optimal arrangement of the proton beam through the patient. Such an optimal arrangement corresponds to a high and homogeneous dose distribution in the target volume, while sparing normal tissue as much as possible. The optimal beam configuration is passed to the patient positioner which positions the patient relative to the beam delivery system according to these specifications. The content of this thesis relates to the patient alignment system which will be discussed further. A detailed discussion of the components of the PPS is provided in *The conceptual design of a robot-based patient positioning system* by Evan de Kock [7].

The patient alignment system (PAS) is a vital component of the PPS because it determines the movements of the patient positioner required to position the patient correctly. The two main components of the PAS are the *CT scanner stereophotogrammetric (SPG) system* and the *Treatment SPG system.* Both these SPG systems are multi-camera computer systems that capture sets of video images of markers positioned on the patient's mask. They use SPG techniques to determine the 3D coordinates of the center of the markers from the video images. The CT scanner SPG system calculates the position of the markers while the patient is located in the reference position used in the CT study. Since the relationship between the SPG system and the CT scanner is known, this establishes the position of the lesion relative to the markers. The 3D coordinates of the markers are transfered to the Treatment SPG system. This system uses this data to determine the relationship that describes the position of the patient in the treatment vault relative to the reference position in the CT scanner room. This relationship, or coordinate transformation, is used to determine the movement of the patient that is required to position the patient in the prescribed treatment position as issued by the treatment planning system.

The CT scanner SPG system and the Treatment SPG system are also responsible for monitoring patient motion during a CT study and treatment sessions respectively. The monitoring process involves determining the position of the patient's mask from video images captured at small time intervals. The position of the patient's mask relative to the respective reference positions describe the movement of the patient. The CT scanner SPG system uses this information to detect patient motion and this information can be used to correct small errors in the CT study which may result from the patient's movement. The Treatment SPG system uses the information to inform the safety system in the treatment room of large[2] patient movements.

The relation between the position of the cameras and the position of the markers on the patient's mask is critical to the success of both SPG systems. The minimum requirement (also referred to as the *marker placement criteria*) of these two systems requires that at least three markers whose position on the mask is known, are visible on images captured by at least two different cameras, as this is needed to reconstruct the unique 3D position of the patient's mask. A minimum of three markers are needed because, one marker are required to determine the position of the mask, but two additional markers are needed to determine the unique orientation of the mask with respect to the treatment vault coordinate system. The position of the cameras in both SPG systems should therefore ensure that a suffi-

[2]A large movement indicates that the patient moved out of the allowed treatment region for time period that is longer than a specified cutoff time (see [7]).

ciently large area of the patient's mask is visible to at least one camera pair. This thesis investigates this relationship between the camera positions and the marker positions on the patient's mask for the CT scanner SPG system and the Treatment SPG system.

Two individual simulation routines were developed. These simulations were implemented in the C++ programming language and used the OpenGL graphics library [34] to produce a graphical representation of the setup in each of the two vaults. The OpenGL library was selected not just because it is one of the most familiar and widely used software interfaces, but because it also provides various commands well suited for interactive 3D applications. These simulation routines provided an ideal environment in which to test and compare the many possible combinations of camera positions and marker positions. Although neither simulations nor computer graphics are exact sciences, errors in simulations are largely due to inadequate modeling rather than graphics problems. A brief overview of some of the main concepts involved in these computer graphics representations are provided in Chapter 2.

The possible position of the cameras in the CT scanner SPG room were fixed before the simulation routines. Their positions were determined by the geometry of the CT scanner and manufacturing constraints. The success of the CT scanner SPG system depends on whether or not enough markers on the patient's mask are visible to the cameras in these fixed positions. A sufficient number of markers are required to successfully reconstruct the position of the patient's mask. The position of the patient's mask is used to establish the relationship between the position of the treatment lesion and the mask, needed by the Treatment SPG system to position the patient in the treatment position and to monitor patient movement during the study. It is therefore critical to investigate the effect of the fixed camera positions on the position of the markers on the patient's mask. This investigation is the topic of Chapter 3.

The position of the markers of the patient's mask are determined in the CT scanner room because of the constraints imposed on the position of those cameras. The success of the Treatment SPG system also depends on the number of markers visible to the cameras in the treatment vault. The two most commonly used treatment masks, a face mask and a body mask, were used in the Treatment vault simulations. This simulation routine aims to determine the camera positions in the treatment vault that will maximize the number of visible markers on the patient's masks. Since neither mask has completely specified marker positions, the simulation routine maximizes the total visible area of each mask. This is a reasonable assumption because a greater number of markers can be placed in a larger visible area of the mask and remain visible to the cameras. In other words, we assume that the number of visible markers increase with an increase in the size of the visible mask areas. Maximizing the number of visible markers minimizes the error in reconstructing the position of the masks. An accurate patient position is necessary for positioning the patient in the required treatment position and determining whether or not the patient moves in or out of position during this treatment. The optimal position of the cameras in the treatment vault is determined in Chapter 4.

Both simulation routines were implemented on the Red Hat Linux 7.3 operating system with a i686-optimized kernel. Three different computers were used. Two were equipped with Pentium III, 1 GHz dual processors with 512 Mb of available RAM. The third computer was equipped with Pentium IV 2 GHz dual processors with 512 Mb of available RAM. The Pentium IV processors also support hyper threading.

A compact disc accompanies this thesis. This disc contains a digital copy of this document (./thesis), the complete set of results from both the marker and camera position simulations (./SimResults), source code of both simulations (./CTScannerRoomSimulations and ./TreatmentRoomSimulations) and digital copies of some of the reference material (./Articles).

We discuss our conclusion in Chapter 5.

Chapter 2

Computer Graphics Concepts

2.1 Introduction

"*The point of computer graphics is to convert a mathematical or geometrical description of an object – a computer graphics model – into a visualization – a two-dimensional (2D) projection – that simulates the appearance of a real object*"– Alan Watt, *3D Computer Graphics* [28]. Computer graphics models are created in application software which provides information to the graphics hardware. In turn, the graphics hardware is responsible for displaying the created objects on a computer screen. The **OpenGL** graphics library is an interface between hardware and software applications. Its libraries provide a set of commands which allows a user to create complex objects from primitives like points, lines and polygons. In addition, these commands also provide facilities to manipulate the objects and the way they are displayed on a computer screen. This kind of functionality provides an ideal environment in which the behavior of objects can be simulated.

This chapter discuss some of the main concepts used in the implementation of the marker placement and the camera placement simulations. The marker placement simulations in Chapter 3 use computer graphics primitives to model the setup in the CT scanner room while the camera placement simulations in Chapter 4 use the primitives to model the setup in the new treatment vault. Some of the more basic OpenGL operations are covered in Section 2.2. These operations include the various phases or steps involved in the rendering process, different techniques used to represent a graphical object, blending and texture mapping. Some more complicated operations like projecting objects to a plane and collision detection are discussed in Section 2.3.

2.2 Basic operation in OpenGL

2.2.1 The graphics pipeline

The graphics pipeline refers to the series of steps that are followed when displaying a created object on the computer screen. Figure 2.1 shows a graphical representation of a typical graphics pipeline. Not all graphic engines follow the exact same order of steps, but most include the steps shown in Figure 2.1. At each step an unique coordinate system provides a framework for a set of distinct and relevant operations. Operations include: modeling transformations, view transformations and rendering processes like hidden surface removal and rasterization. A definition of the different coordinate systems and transformations between them follow.

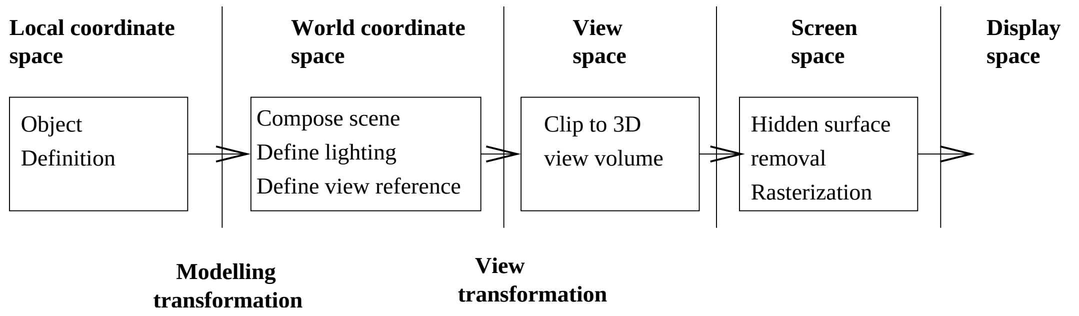


Figure 2.1: The graphics pipeline. The image was taken from *3D Computer Graphics* [28] (page 142).

Basic transformations

The most basic and elementary unit in computer graphics is a vertex point ($\mathbf{v}$). Three transformations, *translation* ($\mathbf{T}$), *rotation* ($\mathbf{R}$) and *scaling* ($\mathbf{S}$) transform such a vertex point in R^3as:

$$\begin{aligned} \mathbf{v} &= \mathbf{v} + \mathbf{T} \\ \mathbf{v} &= \mathbf{R}\mathbf{v} \\ \mathbf{v} &= \mathbf{S}\mathbf{v} \qquad (2.1) \end{aligned}$$

Equation 2.1 expresses these transformations in matrix notation. Only translation is not specified as a matrix multiplication, but as a matrix addition. Specifying these transformations in homogeneous coordinates allow translation to be represented as a matrix multiplication rather than an addition. The result is an unified scheme for linear transformation, all represented by a single matrix multiplication. The homogeneous representation of a vertex point is:

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (2.2)$$

The corresponding single matrix multiplication that represents each transformation is:

$$\mathbf{v}\prime = \mathbf{T}\mathbf{v} \qquad (2.3)$$

$$\begin{bmatrix} x\prime \\ y\prime \\ z\prime \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (2.4)$$

$$(2.5)$$

$$\mathbf{v}\prime = \mathbf{R}\mathbf{v} \qquad (2.6)$$

$$\begin{bmatrix} x\prime \\ y\prime \\ z\prime \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{Rot} & \underline{0} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

$$\tag{2.8}$$

$$\mathbf{v}\prime = \mathbf{Sv} \tag{2.9}$$

$$\begin{bmatrix} x\prime \\ y\prime \\ z\prime \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.10}$$

with **Rot** the 3×3 rotation matrix around a particular axis. The OpenGL commands, *glTranslate(T_x,T_y,T_z)*, *glRotate(angle,R_x,R_y,R_z)* and *glScale(S_x,S_y,S_z)* are responsible for translation, rotation and scaling respectively. The rotation angle (in degrees) is specified with the *angle* parameter in *glRotate(angle,R_x,R_y,R_z)* while the axis of rotation is specified with R_x, R_y and R_z.

Coordinate systems in the graphics pipeline

A different coordinate system is associated with each step in the graphics pipeline shown in 2.1. These coordinate systems provide frameworks for sets of related operations. For example, the **view coordinate system** provides the framework in which the 3D scene is clipped to the view volume. The view volume represents that part of the 3D scene that projects onto the computer screen, while clipping is the process of eliminating those objects, or parts there of, that fall outside this view volume. This step of the graphics pipeline involves transformations and operations related to the way in which an object is viewed. The coordinate systems associated with the steps shown in 2.1 are:

Local coordinate system

The local coordinate system (**L**) is located at a point in or close to the created object. The quadruplet $(x^l, y^l, z^l, 1)$ specifies the coordinates of a vertex point in this coordinate system.

World coordinate system

The homogeneous world coordinate system (**W**) is the global coordinate system of the 3D scene. This coordinate system consists of modeled objects and their respective local coordinate systems. The quadruplet $(x^w, y^w, z^w, 1)$ specifies the coordinates of a vertex point in this coordinate system. All the objects are transformed into this common space to define their relative spatial relationships.

View coordinate system

The homogeneous view coordinate system (**V**) is fixed to the observer (or camera). This coordinate system defines the viewing parameters (viewpoint, view direction) and the view volume. The origin of this coordinate system is the viewpoint while the quadruplet $(x^v, y^v, z^v, 1)$ indicates the coordinates of a vertex point.

Screen coordinate system

The screen coordinate system (**D**) represents the 2D screen. A vertex point in this coordinate system is specified by (x^s, y^s). The origin of this coordinate system is taken to be the lower left corner of the computer screen.

Light coordinate system

The homogeneous light coordinate system (**G**) is fixed to the light source. The origin of this coordinate system is the light source while the quadruplet $(x^g, y^g, z^g, 1)$ specifies the coordinates of a vertex point in this coordinate system.

Texture coordinate system

The texture coordinates system (**K**) is a 2D coordinate system that provides an index into a texture image. A vertex point in this coordinate system is specified by (x^t, y^t).

A graphical representation of these coordinate systems is shown in Figure 2.2. The transformation matrices that are responsible for transforming a vertex point between these coordinate systems are:

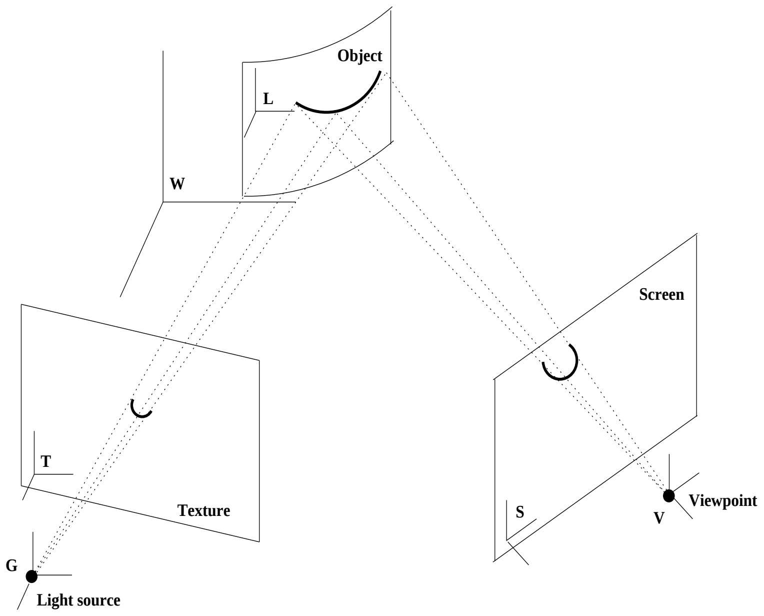


Figure 2.2: A graphical representation of the various coordinate systems present in the graphics pipeline.

Transformation matrices

The matrix representation of the homogeneous coordinates of the vertex point $\mathbf{v}$ in $\mathbf{W}$ is:

$$\mathbf{v} = \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{2.11}$$

The position and orientation of each object is specified relative to its local coordinate system. The *modeling* transformation matrix $\mathbf{M_m}$ is a combination of matrices $\mathbf{T_m}$, $\mathbf{R_m}$ and $\mathbf{S_m}$ and transforms a vertex point in $\mathbf{W}$ to a vertex point in **L**. This transformation is described by:

$$\mathbf{M_m} = \mathbf{T_m R_m S_m} \tag{2.12}$$

$$\begin{bmatrix} x^l \\ y^l \\ z^l \\ 1 \end{bmatrix} = \mathbf{M_m} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{2.13}$$

The *viewing* transformation matrix $\mathbf{M_v}$ is a combination of $\mathbf{T_v}$ and $\mathbf{R_v}$ (assuming unit scaling) and specifies from where and in what direction the objects will be viewed. The transformation of a vertex point **v** from **W** to **V** is described by:

$$\mathbf{M_v} = \mathbf{T_v R_v} \tag{2.14}$$

$$\begin{bmatrix} x^v \\ y^v \\ z^v \\ 1 \end{bmatrix} = \mathbf{M_v} \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{2.15}$$

OpenGL's default viewing parameters are: view point= $(0, 0, 0)$, view direction= $(0, 0, -1)$ and up= $(0, 1, 0)$. These parameters translate to a view point at the origin, the view direction along the negative z-axis and the positive y-axis as straight up. OpenGL command *glLookat(*x_1,y_1,z_1,x_2,y_2,z_2,x_3,y_3,z_3*)* is responsible for changing the default viewing parameters. x_1, y_1 and z_1 specify the x, y and z coordinates of the view point which are transformed by the T_x, T_z and T_z components of the $\mathbf{T_v}$. x_2, y_2 and z_2 specify the view direction and x_3, y_3 and z_3 specifies up. The rotation axis of $\mathbf{R_v}$ is given by the cross product [1] of the normalized vectors $(0, 0, -1)$ and (x_2, y_2, z_2) which are the default and new view directions respectively. The cosine of the angle of rotation is given by the dot product of the same normalized vectors. The matrix $\mathbf{M_g}$ is similar to $\mathbf{M_v}$ and transforms a vertex **v** from **W** to **G**. The *modeling* matrix and the *viewing* matrix are combined to produce the *modelview* matrix $\mathbf{M_{mv}}$. The OpenGL command *glMatrixMode(GL_MODELVIEW)* initializes $\mathbf{M_{mv}}$

[1] vector product

to the identity matrix **I**. Any calls to the OpenGL commands *glTranslate(T_x,T_y,T_z)*, *glRotate(angle,R_x,R_y,R_z)* or *glScale(S_z,S_y,S_z)* after this initialization result in a transformation of $\mathbf{M_{mv}}$.

The transformation from **V** to **D** is via the 3D **screen space**. This transformation determines how the objects are projected onto the 2D screen. Objects can be projected either perspectively or orthographically onto the 2D screen. A perspective transformation makes objects that are farther away appear smaller, which matches how we see things in daily life. An orthographic projection maps objects directly onto the screen without affecting their relative size. The matrix responsible for the perspective transformation, $\mathbf{M_p}$, will be discussed further. The OpenGL command *glFrustum(left,right,bottom,top,near,far)* controls this projection. The frustum's view volume is defined by the parameters *(left,bottom,-near)* and *(right,top,-near)* which specify the (x, y, z) coordinates of the lower-left and upper-right corners of the near clipping plane. Parameters *near* and far give the distances from the viewpoint to the near and the far clipping planes. The parameters defining the view frustum are shown in Figure 2.3. The *glFrustum(left,right,bottom,top,near,far)* command generates the matrix $\mathbf{M_p}$ which perspectively transforms the coordinates of **v** in **V** to a coordinate in the 3D **screen space**.

$$\mathbf{v\prime} = \mathbf{M_p v} \tag{2.16}$$

$$\begin{bmatrix} x\prime \\ y\prime \\ z\prime \end{bmatrix} = \begin{bmatrix} \frac{2\,near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2\,near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{2\,far\,near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^v \\ y^v \\ z^v \\ 1 \end{bmatrix} \tag{2.17}$$

(2.18)

The matrix $\mathbf{M_p}$ is defined as long as $left \neq right$, $top \neq bottom$ and $far \neq near$.

The *viewport* transformation matrix $\mathbf{M_{vp}}$ transforms the coordinates of a vertex $\mathbf{v\prime}$ in 3D **screen space** to a vertex in **D**. The viewport is the rectangular re-

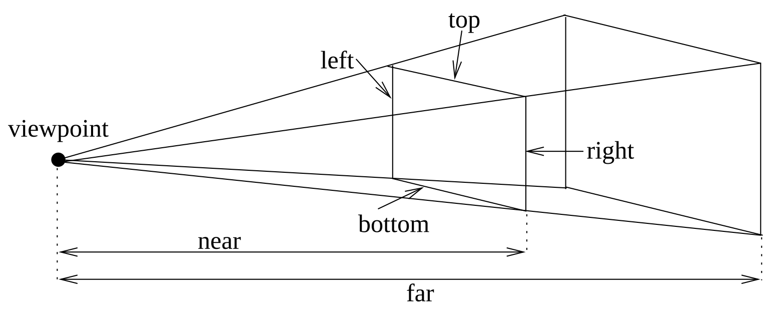


Figure 2.3: The frustum's viewing volume

gion of the window where the image is drawn and reflects the relative position of pixels on the screen relative to the lower-left corner of the window. The *glViewport(x,y,width,height)* command controls this transformation. The x and y parameters specify the position of a vertex on the screen while the *width* and *height* parameters specify the width and height of the screen. The transformation between **G** and **K** will be described in Section 2.2.4.

2.2.2 Representation of 3D objects

A number of techniques exist in which to create objects in computer graphics. These techniques involve polygonal representations, constructive solid geometry (CSG) and implicit representations. An implicit representation describes an object in terms of an implicit function. For example, a sphere can be described in terms of the function

$$x^2 + y^2 + z^2 = r^2$$

where r is the radius of the sphere. CSG is the technique of representing objects with 'combinations'of elementary shapes or geometric primitives. Representing a block with a hole in it as the result of a 3D subtraction of a cylinder from a rectangular solid, is an example of CSG. In another example, the treatment beam nozzle, used in the computer simulations of Chapter 4, is represented as a combination of a cylinder and a cone. This representation of the beam nozzle is shown in Figure 2.4.

Polygonal representation is the technique that is most commonly used to represent

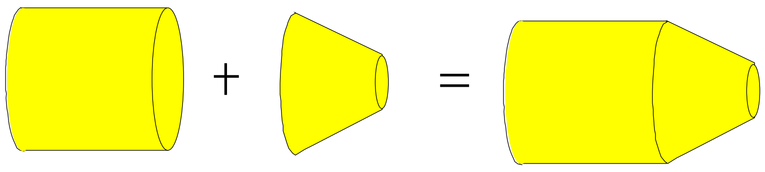


Figure 2.4: CSG representation of the treatment beam nozzle.

objects in computer graphics. With this technique, objects are approximated with a mesh of polygon facets. Each polygon facet is defined by three or more vertices. The position of these vertices can be obtained from scanning the object with either a laser ranger or a 3D-digitizer. Scans like these produce sets of 3D-coordinates which provide the position of the vertices. A common strategy for ensuring an adequate representation is to draw a net over the surface of the object. The position of the vertices are then defined to be the intersection of the curved net lines. A number of algorithms exist that take these vertex coordinates and produce polygon facets. H. Hoppe *et al* developed an algorithm that creates a surface (consisting of polygon facets) from unorganized data points [13] [14] [15]. Their method uses triangulation and mesh optimization routines to reconstruct the surface of the object from the set of vertex coordinates. In addition, a smoothing function is used to smooth sharp edges by increasing the number of polygons in the final representation of the object.

The simulation routine implemented in Chapter 4 uses a polygonal representation of a face mask to determine the optimal position of the cameras in the treatment vault. In this application a patient mask was scanned to produce a set of equally spaced vertices, 0.5 mm apart. This particular face mask produced 2542 vertices at an accuracy of approximately 0.1 mm. The algorithm developed by H. Hoppe was used to construct smooth polygon facets from the vertices in the data set. The different stages in the polygonal representation of the face mask are shown in Figure 2.5. This technique allowed for a fairly accurately representations of a complicated object such as a face mask.

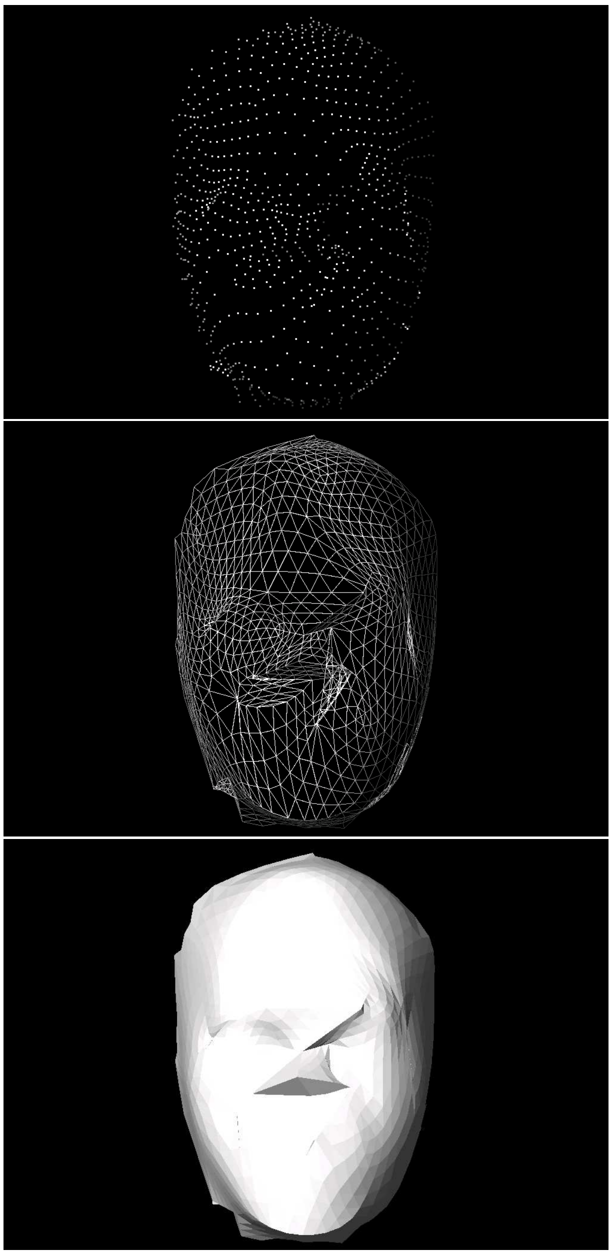


Figure 2.5: Three stages in the polygon representation of a face mask.

2.2.3 Blending

The computer hardware causes each pixel on the computer screen to emit different amounts of red (R), green (G) and blue (B) light [28]. For each object drawn on the screen, one of these RGB (combination of colours red (R), green (G) and blue (B)) values is stored for each pixel. These values are assigned using the *glColor4f(R,G,B,A)* command in OpenGL. One additional value is stored with each RGB value. This value is the alpha value ($A \in [0,1]$) which controls the amount of blending between the colours of different objects. For example, when one object is drawn in front of another, the alpha value controls how much the colour of the object that was drawn first (object furthest from the observer) should be combined with the colour of the object that is drawn second (closest to the observer). Blending and alpha values enables us to recreate objects that are transparent, opaque or semi-transparent. A lower alpha value normally results in a more transparent object.
Blending is performed in a two-stage process. First, you specify how to compute the blending factors for the source (object drawn second and closest to the observer) and destination (object drawn first and furthest from the observer) objects. These blending factors are RGBA quadruplets that are multiplied by the colour quadruplets of the source and destination objects, respectively. Finally, the corresponding components in the two sets of RGBA quadruplets are added. For example, if the colour components of the source and destination objects are specified by the quadruplets *(R_s,G_s,B_s,A_s)* and *(R_d,G_d,B_d,A_d)* respectively and their blending factors are specified with *(S_r,S_g,S_b,S_a)* and *(D_r,D_g,D_b,D_a)* respectively, the final blended RGBA values are:

$$\begin{aligned} R &= R_sS_r + R_dD_r \\ G &= G_sS_g + G_dD_g \\ B &= B_sS_b + B_dD_b \\ A &= A_sS_a + A_dD_A \qquad (2.19) \end{aligned}$$

The colour quadruplets of the source and the destination objects are specified with the OpenGL commands *glColor4f*(R_s,G_s,B_s,A_s) and *glColor4f*(R_d,G_d,B_d,A_d) respectively. The blending factors of both the objects are specified with the OpenGL command *glBlendFunc(source_factor,destination_factor)*. The values of the *source_factor* and *destination_factor* parameters specify how to compute the source and destination blending factors with (R_s,G_s,B_s,A_s) and (R_d,G_d,B_d,A_d). For example, substituting the values of *source_factor* $= (0, 0, 0, 0)$ (black) and *destination_factor* $= (1, 1, 1, 1)$ (white) in Equation 2.19 results in a final colour of (R_d,G_d,B_d,A_d). These blending factors result in replacing the colour of the source object with the colour of the destination object. Blending is enabled and disabled with the OpenGL commands, *glEnable(GL_BLEND)* and *glDisable(GL_BLEND)*.

A blending example

This blending example involves mapping two individual texture images onto the same object. Texture mapping is the topic of Section 2.2.4 and will not be discussed here. Without blending, the second texture will be mapped over the first texture thereby completely obscuring it. One way of preventing the second texture from obscuring the first is to specify a transparent background for the second texture image. This way, the images in both the first and the second texture will be visible. In the first step, the first texture image is mapped onto the object (oval model in this case). Secondly, the blending function is set to *glBlendFunc(GL_ONE_MINUS_SRC_COLOR,GL_SRC_COLOR)*. This blending function specifies that the colour of the source object (object drawn second) should be one minus the colour of the destination object; and the destination object's colour should remain unchanged. In the following example, both the source and the destination objects are texture images featuring images of a black marker on a white background. A black colour for the source object and a white colour for the destination object results in a black colour for the current screen pixel. This result is obtained from substituting values

$$\begin{aligned}(R_s, G_s, B_s, A_s) &= (0,0,0,0)\\(R_d, G_d, B_d, A_d) &= (1,1,1,1)\\(S_r, S_g, S_b, S_a) &= (1,1,1,1)\\(D_r, D_g, D_b, D_a) &= (0,0,0,0)\end{aligned} \tag{2.20}$$

in Equation 2.19. Similarly, if the destinations object's colour is also black, the values

$$\begin{aligned}(R_s, G_s, B_s, A_s) &= (0,0,0,0)\\(R_d, G_d, B_d, A_d) &= (0,0,0,0)\\(S_r, S_g, S_b, S_a) &= (1,1,1,1)\\(D_r, D_g, D_b, D_a) &= (0,0,0,0)\end{aligned} \tag{2.21}$$

result in a black colour for the current screen pixel. The only situation whereby the resulting colour will be white is when both the source and the destination object's colours are white. This resulting colour is obtained from the following values:

$$\begin{aligned}(R_s, G_s, B_s, A_s) &= (1,1,1,1)\\(R_d, G_d, B_d, A_d) &= (1,1,1,1)\\(S_r, S_g, S_b, S_a) &= (0,0,0,0)\\(D_r, D_g, D_b, D_a) &= (1,1,1,1)\end{aligned} \tag{2.22}$$

This result is illustrated in Figure 2.6. The image on the left shows the position of the oval model with respect to the texture images. This effect was obtained by enabling lighting in the scene. In the second image this lighting is disabled again. The image on the right is an example of the images used in the simulations in Chapter 3.

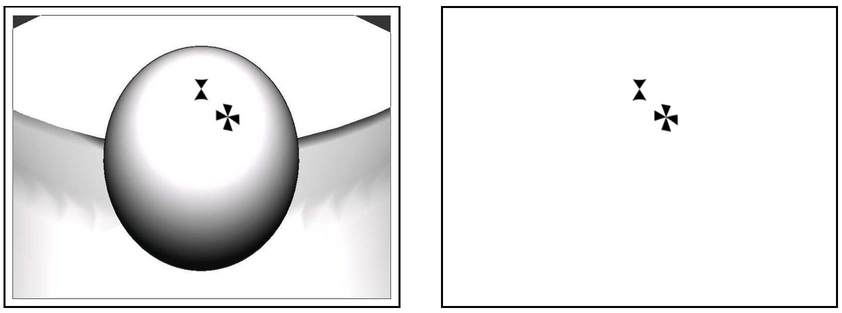


Figure 2.6: A blending example that shows the effects of blending when specifying transparent objects.

2.2.4 Texture Mapping

Textures are rectangular arrays of data - for example - colour data. Texture mapping is the process of applying these textures to an object in a 3D scene. The simplest form of texture mapping involves applying a 2D texture image to a rectangular polygon facet which is specified by four vertex points. The polygon facet is interpolated to produce texture coordinates which are used as an index into the texture image. The colour values of the texture image either replace the object's colour value or blend with it as described in Section 2.2.3. The OpenGL command *glTexImage2D('Texture_name')* specifies the name of the texture while *glTexCoord(x,y,z)* specifies its coordinates. In the case of applying a texture image to the rectangular polygon facet, the texture coordinates and coordinates of the polygon facet are specified by:

$$\begin{aligned} glTexCoord(-1,1,0); &\qquad glVertex(-1,1,0);\\ glTexCoord(1,1,0); &\qquad glVertex(1,1,0);\\ glTexCoord(1,-1,0); &\qquad glVertex(1,-1,0);\\ glTexCoord(-1,-1,0); &\qquad glVertex(-1,-1,0); \end{aligned}$$

Texture mapping becomes a bit more tricky when applying a rectangular texture image to more general shaped objects. The problem lies with the non-linear interpolation of the objects. Standard techniques, such as environment mapping, have been developed for mapping a texture on quadratic objects like spheres or cylinders. Mapping textures on non-quadratics object, like an oval model, requires a technique called projective texture mapping.

Projective texture mapping

Projective texture mapping is the method of texture mapping that allows the texture image to be projected onto a object as if by a slide projector [23] [3] [6]. It refers both to the way texture coordinates are assigned to the vertices, and the way they are computed during rasterization[2] of the objects. The key to projective texture mapping is the contents of the texture transform matrix ($\mathbf{M_{tg}}$). This matrix is a concatenation of the *modelview* matrix, the projective matrix and a scaling matrix. The modelview matrix orientates the projection in the scene using the OpenGL command *glLookat()* (see Section 2.2.1). The projective matrix is responsible for a perspective correct mapping using the OpenGL command *glFrustum()* and the scaling matrix maps the texture coordinates to the near clipping plane. The texture transformation matrix is given by:

$$\mathbf{M_{tg}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{M_p M_v M_m} \qquad (2.23)$$

where $\mathbf{M_m}$ is the modeling matrix, $\mathbf{M_v}$ is the viewing matrix, $\mathbf{M_p}$ is the projective matrix and the final matrix renormalizes the texture coordinates to the range $[0, 1]$. We describe the process of projective texture mapping in the framework shown in Figure 2.2. The *modelview* matrix transforms the coordinates of the light source to the origin (default view point) and the texture coordinates to the projection centered along the negative z-axis. In this case the viewer can be thought of as a light source and the near clipping plane of the projection as the location of the texture image, which can be thought of as printed on a transparent film. The projective matrix converts these coordinates from **WC** to normalized device coordinate (**NDC**) space. In the **NDC** space the coordinates x, y and z range from -1 to 1. The scaling matrix then renormalizes these coordinates to texture coordinates ranging from 0 to 1. This transformation to **NDC** space ensures that the desired portion of the image is centered and covers the entire near plane defined by the projection. It also ensures a correct projection on various different hardware platforms and graphics devices.

[2]Rasterization or scan conversion is the process of determining the actual pixels of an object and assigning it an intensity value.

All that remains is to specify the coordinates of the primitives on which the texture will mapped. This can be done by enabling OpenGL's texture generation facility *glTexGen(GL_EYE_LINEAR)*. This facility simply generates texture coordinates from the vertex attributes in **WC**. These texture coordinates are transformed by the texture transform matrix. This matrix performs both a *modelview* and projection transformation which orientates and projects the primitive's texture coordinates to NDC space. Lastly, these coordinates are normalized to $[0,1]$. Any additional filtering operations are performed and each pixel on the primitive is assigned the intensity value of the corresponding pixel in the texture image.
Projecting a non-repeating texture onto an untextured surface can be done by setting the *GL_MODULATE* environment variable and the texture repeat mode to *GL_CLAMP*. If the texture border is white, the surface outside the projected texture will be modulated with white. A texture repeat mode set to *GL_REPEAT* will have the opposite effect and repeat the image over the primitive. The effect of using these different texture repeat modes is illustrated in Figure 2.7. Note how the texture images of the marker look as if they have been painted on the oval model.

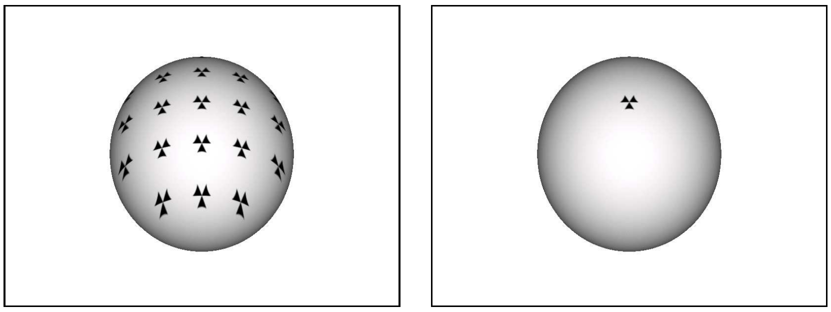


Figure 2.7: An example that shows the effects of *GL_REPEAT* and *GL_CLAMP* parameters when projecting a texture onto an object.

2.3 Ray Based Algorithms

Algorithms related to computer graphics include: radiosity, rendering, rasterization, global illumination, mapping and collision detection, to name but a few. Many of these algorithms are based on the concept of rays. Rays are mathematical representations of directed line segments. One common use of rays is ray tracing algorithms where the path of light through the environment is simulated by rays. At each step, these rays are tested against intersections between the objects in the environment. This technique is used to determine the intensity of the light at each particular point on an object.
The definition of a ray is given in Section 2.3.1. This section also describes one way of computing the intersection between a ray and one of three standard primitives namely a cylinder, a plane and a cone [11]. In Chapter 4 we use this concept to determine which part of the face mask gets projected onto the CCD of the cameras in the treatment vault. In the same chapter, a ray is also used to detect whether a collision occurred between the beam nozzle and the treatment couch. Both these applications are discussed in Sections 2.3.2 and 2.3.3 respectively.

2.3.1 The ray

The 3D coordinates of a vertex is defined as the vector:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.24}$$

A ray is defined by an origin (or eye point), $E = (x_E, y_E, y_E)$, and an offset vector $D = (x_D, y_D, z_D)$. The equation of a ray is:

$$P(t) = E + tD, \quad t \geq 0 \tag{2.25}$$

Finding the intersection point between a ray and an object requires finding the lowest non-negative value of t in Equation 2.25. If D is a unit vector, the value of t indicates the distance to the point of collision.

Intersection between a ray and a plane

A plane can be defined by a normal vector, N, and a point on the plane, Q. A point, P, is on the plane if:

$$N \cdot (P - Q) = 0 \tag{2.26}$$

To find the ray/plane intersection, we substitute the equation of a ray (2.25) in the equation of the plane (2.26), which gives

$$N \cdot (E + tD - Q) = 0$$

thus

$$t = \frac{N \cdot (Q - E)}{N \cdot D} \tag{2.27}$$

If $t \leq 0$ then the plane is behind the eye point and there is no intersection. If $t \geq 0$ then the intersection point is $E + tD$. If $N \cdot D = 0$ then the plane is parallel to the plane, and there is no intersection point.

Intersection between a ray and a cylinder

The finite cylinder aligned along the z-axis is defined as:

$$x^2 + y^2 = 1, \quad z_{min} < z < z_{max} \tag{2.28}$$

To intersect a ray with a cylinder, substitute the ray equation in the equation of the cylinder (Equation 2.28).

$$\begin{aligned} (x_E + tx_D)^2 + (y_E + ty_D)^2 &= 1 \\ t^2(x_D^2 + y_D^2) + t(2x_E x_D + 2y_E y_D) + (x_E^2 + y_E^2 - 1) &= 0 \\ at^2 + bt + c &= 0 \\ t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \end{aligned} \tag{2.29}$$

where

$$\begin{aligned} a &= x_D^2 + y_D^2 \\ b &= 2x_E x_D + 2y_E y_D \\ c &= x_E^2 + y_E^2 - 1 \end{aligned} \tag{2.30}$$

Equation 2.29 produces at most two values for t. The value of t that satisfies $z_{min} < z < z_{max}$, or the smallest t value if both satisfy this condition, represents the closest intersection point between the ray and the cylinder. A further test determines the intersection point between the ray and the end cap of the cylinder. This test is similar to the intersection test between a ray and a plane with two additional criteria. The end caps have the formulas:

$$\begin{aligned} z = z_{min}, &\qquad x^2 + y^2 \leq 1 \\ z = z_{max}, &\qquad x^2 + y^2 \leq 1 \end{aligned} \tag{2.31}$$

Intersection between a ray and a cone

The finite cone, aligned along the z-axis, is defined as:

$$x^2 + y^2 = z^2, \quad z_{min} < z < z_{max} \tag{2.32}$$

To intersect a ray with a cone, substitute the ray equation in the equation of the cone (Equation 2.32).

$$\begin{aligned} (x_E + tx_D)^2 + (y_E + ty_D)^2 &= (z_E + z_D)^2 \\ t^2(x_D^2 + y_D^2 - z_D^2) + t(2x_E x_D + 2y_E y_D - z_E z_D) + (x_E^2 + y_E^2 - z_E^2) &= 0 \\ at^2 + bt + c &= 0 \\ t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \end{aligned} \tag{2.33}$$

where

$$\begin{aligned} a &= x_D^2 + y_D^2 - z_D^2 \\ b &= 2x_E x_D + 2y_E y_D - 2z_E z_D \\ c &= x_E^2 + y_E^2 - z_E^2 \end{aligned} \tag{2.34}$$

If z_{min} and z_{max} are both positive or both negative you get a single cone with its top truncated.

Transforming the primitives

The ray/primitive intersections described above detect the point of intersection between a ray and primitives that are centered at the origin. In order to ray trace these primitives in an arbitrary location, we use geometric transformations to scale (**S**), rotate (**R**) and translate (**T**) them to the desired locations. These transform the primitive from the standard position (centered at the origin) to the desired location. To perform the intersection, we take the inverse transform of the ray, intersect this with the primitive on the standard position, and then transform the resulting intersection point to its correct location. For example, to intersect ray, $E+tD$, with B we transform the point, E and the displacement, D as follows:

$$\begin{aligned}\hat{E} &= S^{-1}R^{-1}T^{-1}E \\ \hat{D} &= S^{-1}R^{-1}D \qquad (2.35)\end{aligned}$$

Note, the displacement is not translated, because it is not affected by translation. Now we intersect this new ray, $\hat{E}+t\hat{D}$, with the object in its standard position, $\hat{B}$. The value of t can then be substituted in Equation 2.25 to give the correct intersection point, if it exists.

2.3.2 Projecting a face model onto the CCD of the camera

Rays can be used to determine which part of the face model projects onto the CCD of the camera. In this example, each vertex point of each triangle of the face mask model has to be tested to determine whether it projects onto the CCD of the camera or not. The start of the rays are defined by the coordinates of the vertex points of the triangles and their directions are determined by subtracting each ray's start from the position of the camera's lens. Each of these rays are then tested for an intersection between itself and the CCD of the camera (a plane). If all the triangle points fall within the boundaries of camera's CCD, that triangle projects successfully onto the camera's CCD and is therefore visible to that camera. Conversely, if any triangle point fall outside the camera's CCD, that triangle does not project onto the camera's CCD and it is not visible to that camera. The triangle fractions that are not considered as a result of this last criteria, are assumed to be negligibly small. The effect of projecting a triangle and a complete face mask to a plane are shown in Figures 2.8 and 2.9 respectively. Note how in each case the projected images are upside down on both the plane and the CCD of the camera.

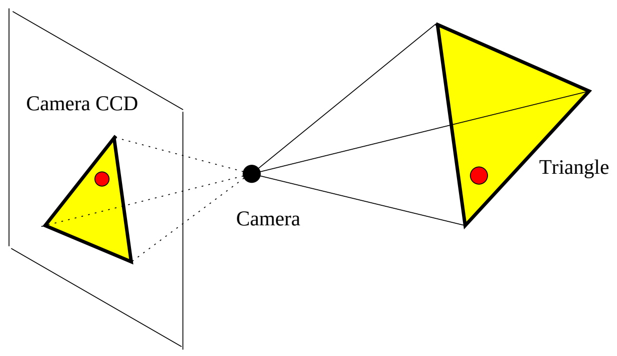


Figure 2.8: Projecting a triangle onto the CCD of a pinhole camera.

There is an additional complication that must be considered. The problem is that some triangles might project to the same area of the camera's CCD. It these cases, both rays had the the same direction, but different starting points. This problem is easily dealt with by comparing the t values of each ray. These t values are an indication of the distance between this triangle and the CCD of the camera and thus the smaller of the two values will be associated with the visible triangle.
To determine if two triangles project onto the same area, we test each projected triangle against every vertex of every other projected triangle using Algorithm 2.3.1 [18]. If a vertex point lies within the tested triangle, the two triangles project to the same space on the CCD of the camera. The algorithm proceeds by imagining an observer "walking" from one point of the triangle to the next, each time determining

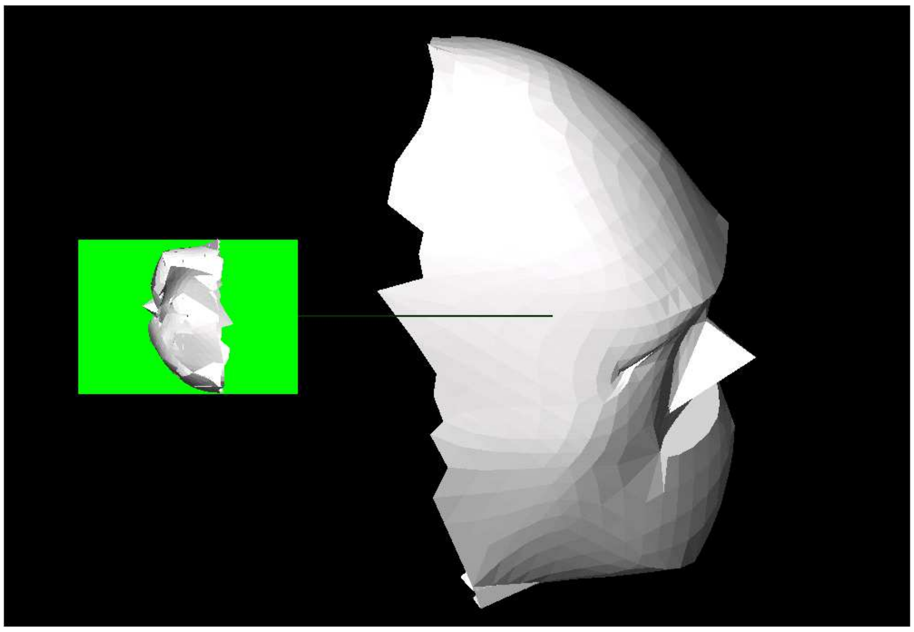


Figure 2.9: The projection of the face model onto the CCD of the camera.

whether the vertex from another triangle lies to his/her left or right. After the third "walk" between two triangle points, the algorithm stops. If the vertex point from the other triangle stayed on the same side of the observer (either left or right), the point lies within the triangle. If not, the point does not lie in the triangle. We do not need to consider partially obscured triangles as we assume a solid object and where this is not true, the errors are small. We use A, B and C to indicate the triangle points with coordinates x_j and y_j with $j = 1, 2, 3$. The coordinates of the point being tested are given by x and y.

Algorithm 2.3.1

```
1 if $(f_{AB}() \times f_{BC} > 0)$ & $(f_{BC}() \times f_{CA} > 0)$
2   then $return\ Inside$
3    else $return\ Outside$
4   endif
5   $f_{AB}() : return\ (y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1)$
6   $f_{CA}() : return\ (y - y_3)(x_1 - x_3) - (x - x_3)(y_1 - y_3)$
```

```
7  $f_{BC}() : return\ (y - y_2)(x_3 - x_2) - (x - x_2)(y_3 - y_2)$
```

2.3.3 A collision detection algorithm

A number of different collision detection algorithms exist. The one discussed here is based on the intersection between a ray and either a cylinder or a cone. The treatment beam nozzles are represented by the combination of a cylinder and a cone (see Figure 2.4). In the treatment room simulation, a face model is positioned in close proximity to the cone end of one of these beam nozzles. Certain orientations of this face model will result in a collision with the beam nozzles. A collision detection test is required. If a collision occurs, the front end (cone) of the relevant beam nozzle is contracted approximately 100 mm from the point of the face model that is closest to the beam nozzle. The algorithm is listed in Algorithm 2.3.2.

Algorithm 2.3.2

```
for each triangle in face model do
    ray_start = point on triangle
    ray_direction = movement direction
    test intersection between ray and cylinder
    text intersection between ray and cone
    if (intersection occured)
      then $t$ = the distance to intersection
    endif
endfor
$t_{min} = min\{t\}$
if ($t_{min} < 100\,mm$)
  then contract the beam nozzle
endif
```

Chapter 3

Marker Simulations

3.1 Introduction

The CT scanner SPG system is implemented in the CT scanner room and consists of six CCD cameras and a computer with stereophotogrammetric software. The aim of this system is to determine the 3D coordinates of the center of the retro-reflective markers placed on the patient's mask. This is achieved by analyzing images of the markers, which were captured by the cameras in the CT scanner room, and using stereo techniques to determine their positions. The CT scanner SPG system requires a minimum of three markers to be visible to at least one camera pair in the CT scanner room. These markers need to be known in the sense that we should know their position on the mask. The minimum requirement of the CT scanner SPG system regarding the markers is referred to as the *marker placement criteria.* If the marker positions on the patient's mask satisfy this criteria, then the CT scanner SPG system will be able to extract enough information to determine the unique 3D position of the mask.

The retro-reflective markers on the patient's mask are classified as either identifiable or standard markers. Identifiable markers are unique and thus can be identified purely from their appearance. From the set of identifiable markers, only one of each will be allowed on the mask. Standard markers are all identical and can only be identified based on their position relative to other markers. Their purpose is to increase the accuracy of determining the position of the patient's mask by increasing the number of usable features in a set of stereo images. The *marker placement criteria* applies to the identifiable markers.

The position of the six cameras in the CT scanner room is fixed. Their positions were determined by the geometry of the CT scanner and manufacturing constraints [19]. These fixed camera positions affect the visibility of the patient's mask and therefore also the usable positions of the markers on it. A simulation routine was developed to simulate different marker positions on the patient's mask. This routine resulted in a set of images that represent the different camera views of the patient's mask. These images were fed to a preliminary *marker identification* algorithm [27] which attempted to identify the different markers in each image. The algorithm resulted in either a *hit* or a *miss*, depending on whether the marker was successfully identified or not. These *hits* and *misses* determine a grid on the patient's mask which indicate marker positions that are sufficiently visible (visible enough to be identified) to at least one camera pair.

The *marker placement criteria* should at least be satisfied for the mask positioned at the start and end of the scan. Favorable marker positions at these mask positions require a less dense distribution of markers on the mask. Reducing the number of markers on the mask simplifies the marker detection process. It also enhances patient comfort because less of the mask is covered with markers.

The setup of the CT scanner room is described in Section 3.2 while a description of the simulation routine is given in Section 3.3. Note that this simulation routine only considers an approximation to the face mask. The body mask imposes no significant additional criteria and can therefore be ignored for this simulation. This routine also uses a preliminary marker identification algorithm, briefly described in Section 3.4, because the final marker identification algorithm is still being developed. The proposed marker positions that resulted from the simulation are shown in Section 3.5. Section 3.6 concludes this chapter.

3.2 The CT scanner setup

CT scanners are designed to perform *studies* of a patient [7]. A study results in a set of CT images where each image represents a thin slice through a section of the patient. The pixels in these images represent some physical property of the patient's anatomy. Consecutive images are normally (with conventional step-by-step CT scanners) small distances, typically 5 mm, apart and parallel to each other. These consecutive images from a CT study are used to build a 3D model of the patient's anatomy in a region of interest. Such models are used to determine the exact size and position of a lesion within the patient's body [1].

The CT scanner room consists of a conventional step-by-step CT scanner and the six cameras used by the CT scanner SPG system. The dimensions of the CT scanner, as provided by iThemba labs, are shown in Figure 3.1. The CT scanner room coordinate system is defined relative to the CT scanner, and described in Section 3.2.1. Any movement of the patient is described in terms of this coordinate system. Since the relationship between the SPG coordinate system and the CT scanner coordinate system is known, any movement of the patient can also be described in terms of the SPG coordiante system. The particular movement of a patient required when conducting a CT study is specified relative to three reference patient positions described in Section 3.2.2.

[1]CT scanner data is often used with MRI data to improve accuracy, etc.

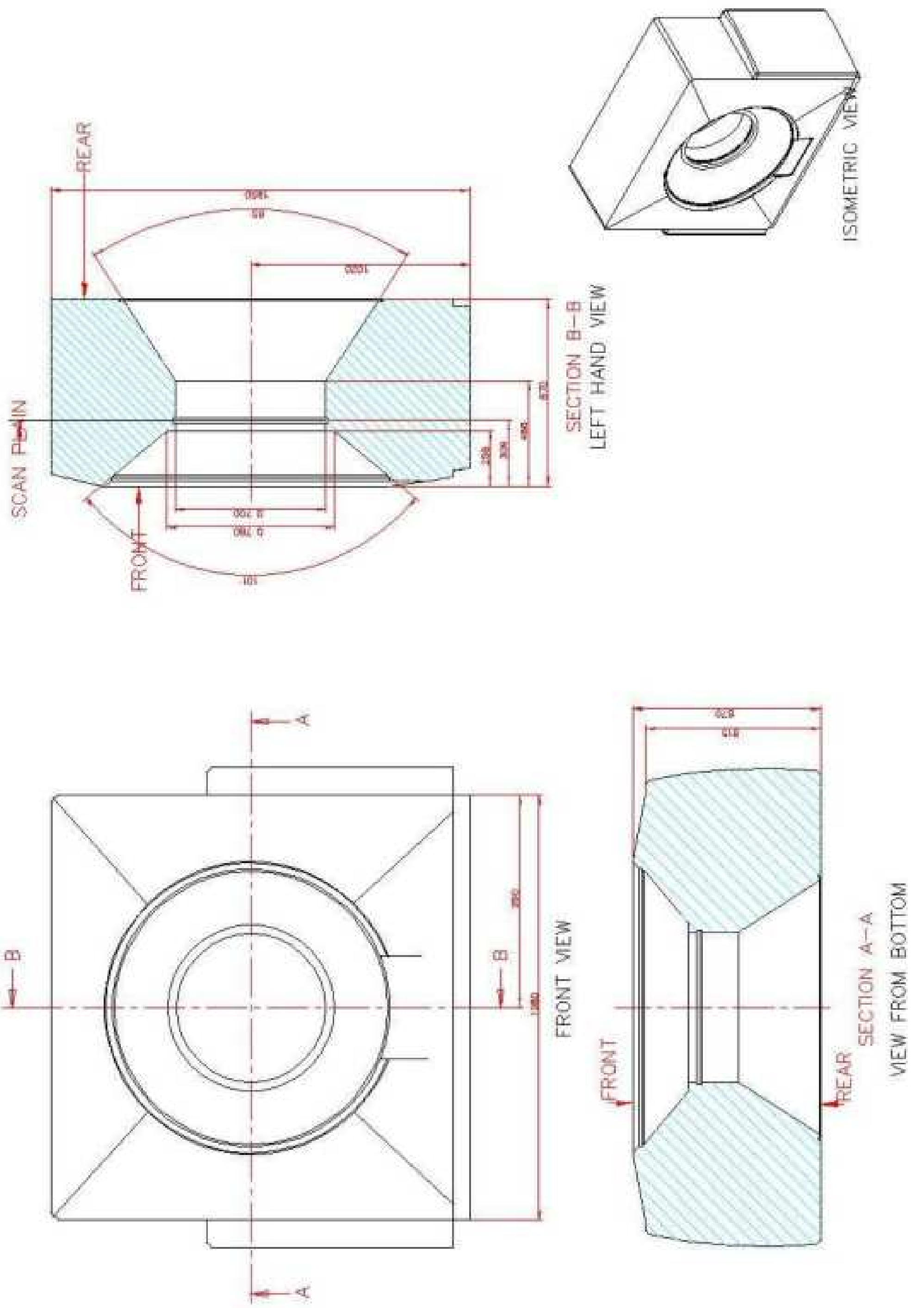


Figure 3.1: The specification of the CT-scanner at iThemba labs.

3.2.1 CT scanner model

CT images are taken at the *scan plane* indicated in Figure 3.1. The *scan point* is, by definition, the point in the middle of this *scan plane*. This point also represents the origin of the CT scanner room coordinate system. In this coordinate system, the positive z-axis points to the front of the CT scanner, the positive x-axis points to the top of the CT scanner and the positive y-axis points to the right of the CT scanner when facing the front of the CT scanner. This coordinate system is shown in Figure 3.2. The CT scanner coordinate system portrays reflection symmetry of the CT scanner with respect to the z-direction.

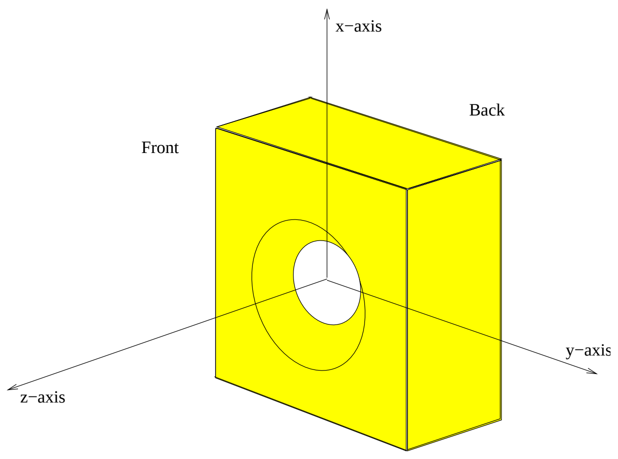


Figure 3.2: The CT scanner room coordinate system relative to the CT scanner.

3.2.2 CT study

The CT study of a patient is conducted in three stages. These three stages are characterized by the direction a patient is translated along the z-axis and two reference patient positions that bounds this translation. The reference positions are:

CT-study reference position
This position is also referred to as the default or initial patient position. Before starting the CT study, the patient is positioned in such a manner that the center of the study volume (region of interest) coincides with the *scan point*. The position of the study volume relative to the *scan plane* is shown in Figure 3.3.

Start of scan
At this position, the study volume is positioned completely in front (positive z-direction) of the *scan plane* and has one endpoint positioned close to the *scan point*. This position bounds the translation of the patient in the positive z-direction. The position of the study volume relative to the *scan plane* is also shown in Figure 3.3.

End of scan
At this position, the study volume is positioned completely behind the *scan plane* and has one endpoint positioned more or less at the *scan point*. This position bounds the translation of the patient in the negative z-direction. The position of the study volume relative to the *scan plane* for each position is also shown in Figure 3.3.

The three stages in the CT study are:

Stage 1
This stage starts with the study volume positioned at the *CT-study reference*

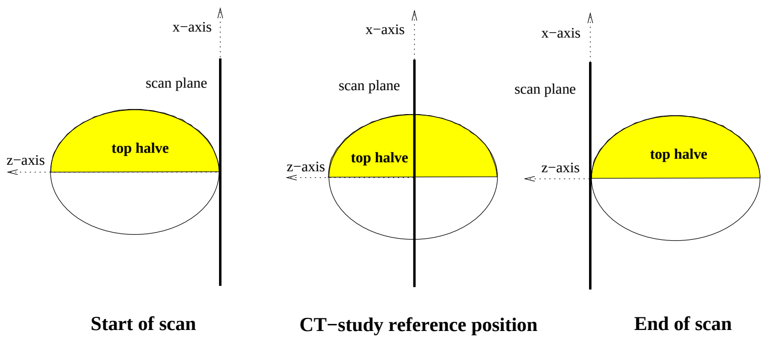


Figure 3.3: The patient reference positions relative to the *scan plane*. The top halves of the spheres indicate the regions in which the markers are placed.

position and ends with the study volume positioned at the *start of scan*. The patient is translated in small incremental steps (5 mm) from the start to the end of this stage. No CT images are taken during this stage in the scan.

Stage 2

This stage starts with the study volume positioned at the *start of scan* and ends with the study volume positioned at the *end of scan*. Again the patient is translated in small incremental steps (5 mm) from the start to the end of this stage, but in the opposite direction to that in stage 1. A CT image of the patient gets taken at each translation step during this stage.

The initial position of the study volume at the *CT-study reference position* allows alignment of the mask coordinate system (a coordinate system that is rigidly attached to the mask of the patient) with the CT scanner coordinate system. The purpose of stage 1 in the CT study is to position the study volume at the *start of scan*. The extent of movement of the study volume through the latter two stages in the CT study is enough to cover the entire region of interest.

3.2.3 Cameras positions in the CT scanner room

Six charged-coupled device (CCD) cameras are mounted in the CT scanner room. Their positions are symmetrical with respect to the z-axis. Three of these cameras are mounted to the roof in front of the CT scanner while the remaining three are mounted to a frame on the floor behind the CT scanner. All six of these cameras are focused on the *scan point*. Their fields of view are large enough to allow a clear view of that section of a patient that has to be included in the CT study. A horizontal view angle of $\nu_h = 12.18°$, a vertical view angle of $\nu_v = 9.15°$ and a focal length of $f = 30\,\text{mm}$ ensures this, at least for when the study volume is positioned at the *CT-study reference position* [26]. The coordinates of the camera positions are summarized in Table 3.1. A top view of these positions is shown in Figure 3.4.

camera	**x-coordinate**	**y-coordinate**	**z-coordinate**
Front middle	1265 mm	0 mm	1109 mm
Front offset	1265 mm	$\pm$600 mm	1109 mm
Back middle	680 mm	0 mm	-1441 mm
Back offset	680 mm	$\pm$300 mm	-1441 mm

Table 3.1: The coordinates of the camera positions in the CT scanner room

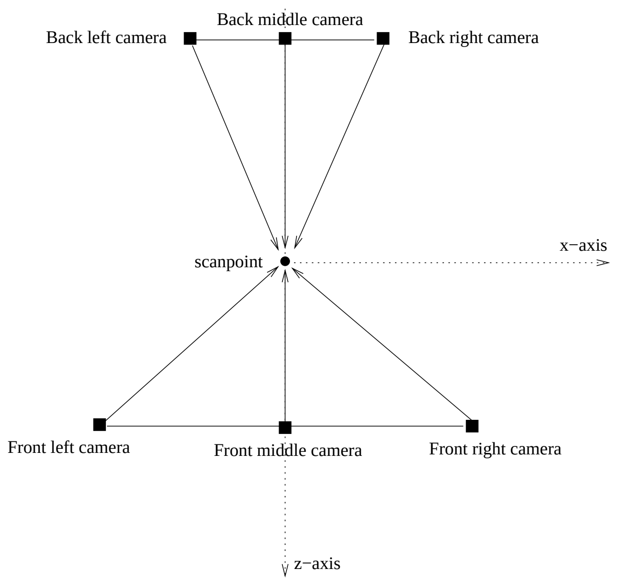


Figure 3.4: A top view of the camera positions in the CT scanner room.

3.3 Simulation procedure

The simulation procedure determines which areas on the mask are suitable for marker placement. Suitable mask areas are visible from at least two cameras simultaneously. Altough these areas satisfy the minimum requirements of the CT scanner SPG system, they do not always result in a correct reconstruction of the marker position. For example, although a marker is visible from two cameras simultaneously, the marker might be at such an acute angle that a correct identification of that marker is impossible. An incorrect identification of a marker will undoubtedly lead to an incorrect reconstruction of the marker position. The simulation uses a preliminary marker identification algorithm when determining positions on the mask that is suitable for marker placement. The final system will have a number of consistency checks in place to handle misidentification robustly, but these are not included in the simulations.

Dirk Wagener developed a pattern registration algorithm (simply referred to as the preliminary marker identification algorithm) used to identify the markers at different positions on the patient's mask [27]. This algorithm identifies candidate patterns (or markers in this case) by detecting the corners in these patterns and is described in more detail in Section 3.3.1. This identification technique is sensitive due to line identification problems like corners and will therefore underestimate the areas on the mask that are suitable for marker placement. The results from this algorithm provides information regarding the robustness of the marker positions under a worst case scenario. The final marker identification algorithm should be able to identify markers positioned in at least these mask areas.

The marker identification algorithm takes as input a stream of video images of the different marker positions on the oval model. These images are created by the simulation procedure described in Section 3.4. Three different sets of images are created. Each one represents the different marker positions with the mask in one of the three reference mask positions.

3.3.1 Marker identification algorithm

The feature based pattern classifier uses corner detection algorithms to identify different markers, invariant to scale and rotation (both in and out of plane rotation). This algorithm uses the set of six distinct markers (marker set) shown in Figure 3.5. The size of each marker is approximately 15 mm×15 mm. The algorithm takes as input a stream of video images and identifies the detected markers in each image. The algorithm uses a so-called model-based approach where each detected marker is compared to the markers in the marker set and then assigned a marker number based on the number of straight edges present in each marker. A *hit* indicates a correct identification of a marker and a *miss* indicates an incorrect identification of a marker. The marker tracking algorithm tracks features in correctly identified

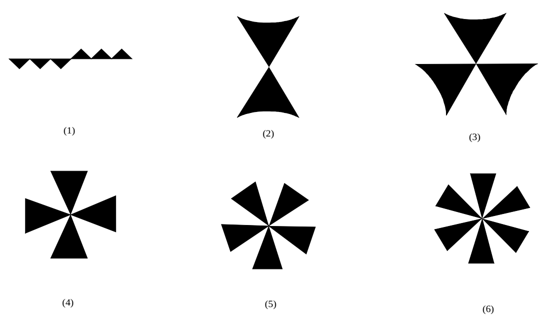


Figure 3.5: The six identifiable markers in the marker set.

markers. The success of the tracking algorithm is therefore directly dependent on the robustness of the marker identification algorithm. In this simulation procedure, we will only simulate the marker identification and not marker tracking.

3.4 Simulation routine

The simulation routine is implemented as a C++ program that uses OpenGL libraries and primitives for modeling and viewing. This routine results in a computer graphics representation of the setup in the CT scanner room. Several simplifications were made. For example, the patient's mask was approximated using an oval shaped model. This approximation was necessary because of the complications involved in mapping textures (2D images of the markers) on a non-quadratic object (see Section 2.2.4). The size of this oval model is 280 mm in the oblong direction (z direction) and 200 mm in the flat direction (x and y directions). This model is considered a good representation of an average sized head mask. Also, only the top or "face" half of the oval model will be considered for marker placement. The face half of the oval model corresponds to the part of the oval that is above the *zy-plane* when the oval is positioned at the *CT-study reference position*. The CT scanner is modeled according to the CT scanner specifications given in Section 3.2.1. The six cameras in the CT scanner room are modeled as perspective pinhole camera models based on the specifications in Section 3.2.3. The view of the oval model, in the *CT-study reference position*, from different cameras are shown in Figures 3.6 and 3.7. Only two of the markers in Figure 3.5 are selected for the

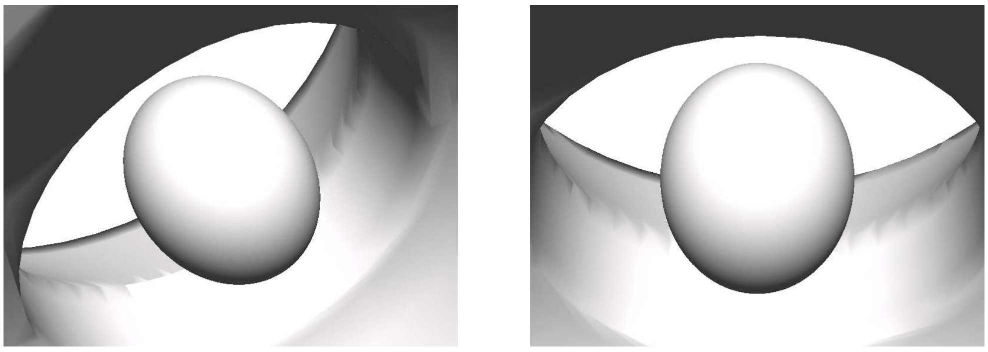


Figure 3.6: The view of the oval model from the front offset (left) and the front middle (right) cameras.

simulations. These are markers 2 and 3. Selecting two different markers ensures some independence with regard to the markers used in the simulation routine. The simulation routine captures a stream of images, where each image shows one of these two markers at a different position on the oval model. Since this is not the final marker identification routine, it is doubtful whether any significant information can be gained by using more marker types. Running the marker identification

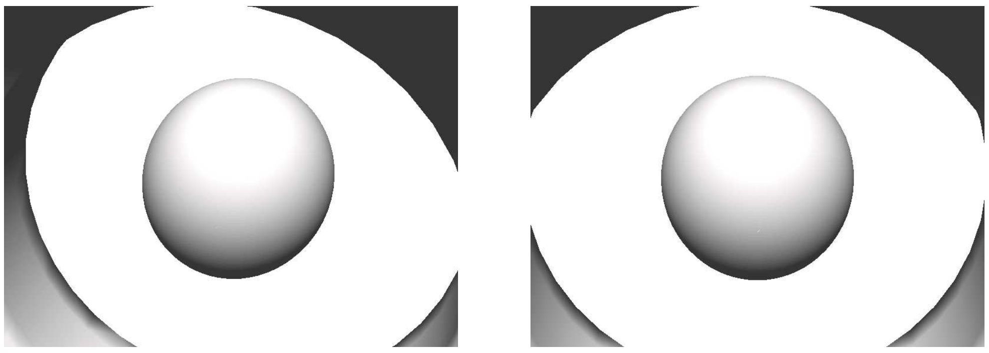


Figure 3.7: The view of the oval model from the back offset (left) and the back middle (right) cameras.

algorithm on such a set of images results in either a *hit* or a *miss*, depending on whether the marker in each specific position was identified or not. This process is repeated for the oval model in each of the reference mask positions.

3.4.1 Markers positions on the oval model

The size of the oval model is 280 mm and 200 mm in the elongated and flat directions respectively. A grid of dimensions approximately $10\,mm \times 10\,mm\,(max)$ is defined over the face (top) half of the oval model. One set of grid lines are parallel to the y-axis (10 mm apart) and the other set connects the two end points of the oval with the largest displacement from the *scan point*. These grid lines are shown in Figures 3.8 and 3.9. Only one half of the grid lines on the face half of the oval model is shown in these figures. The symmetrical nature of the oval model and camera positions allow us to consider only one half of the total number of marker positions. The results from the markers in the other marker positions will be symmetrical to the results from the considered positions. The specified grid size results in $28 \times 14 = 392$ grid points, each one representing a different marker position. The position of a marker with respect to one of these grid points is shown in Figures 3.8 and 3.9.

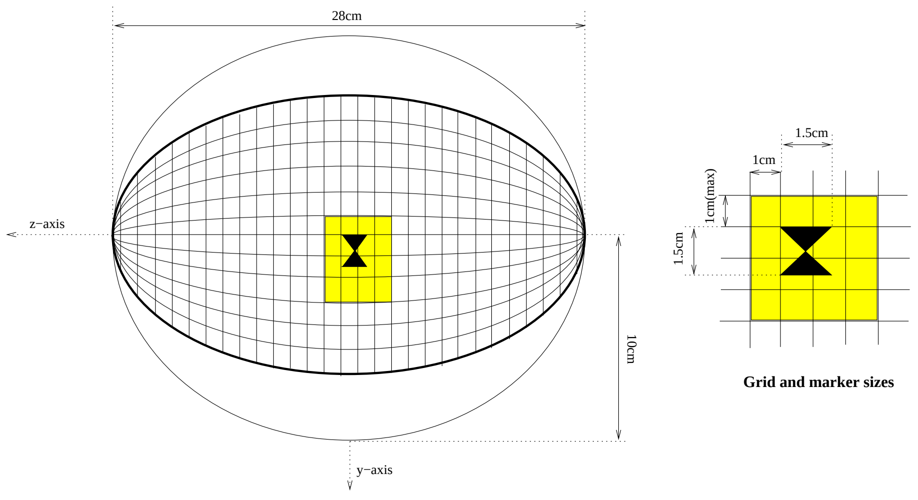


Figure 3.8: Side view of the grid on the oval model.

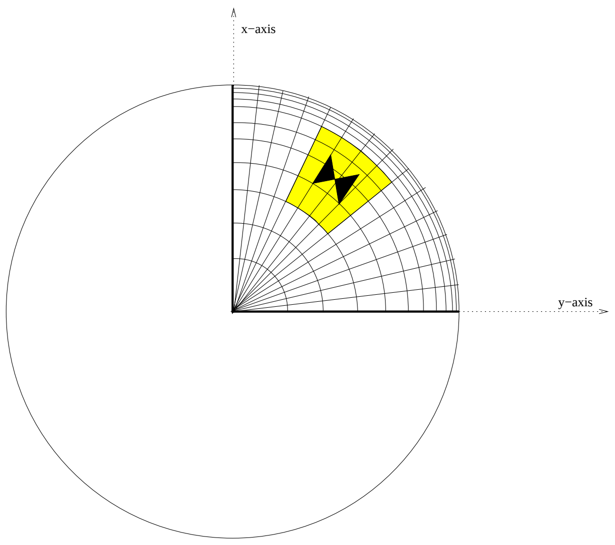


Figure 3.9: Front view of the grid on the oval model.

3.5 Simulation results

The results from the marker identification algorithm are represented on a grid of 28×14 squares. The results from the marker identification algorithm at each of these marker positions are indicated as either a black (*hit*) or white (*miss*) square. Images of these grids are produced for each of the two markers and each camera. An additional image gives a crude representation of the area on the oval model that corresponds to *hits* in these images. This process was repeated with the oval model in all three reference positions. The full set of results is provided in the *./SimResults* directory on the accompanied compact disc.

The *marker placement criteria* of the CT-scanner SPG system requires a marker to be visible from at least two cameras simultaneously. These areas are obtained by combining the marker position that resulted in hits. A marker position that results in a *hit* from more than one camera satisfies the placement criteria.

The total areas of the mask that satisfy the *marker placement criteria* for an oval model positioned at the *CT-study reference position*, at the *start of scan* and at the *end of scan* are shown in Figure 3.10. These areas include areas on both sides of the symmetrical axis defined in Section 3.4.1.

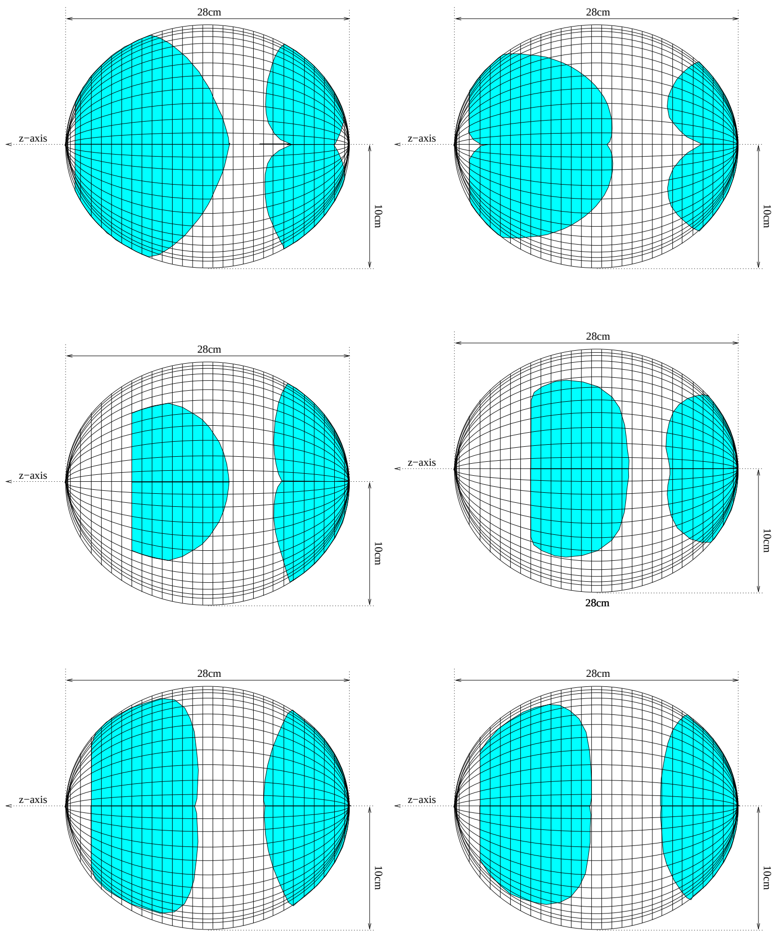


Figure 3.10: The marker positions that satisfy the marker placement criteria when the oval model is positioned at the *CT-study reference position* (top), the *start of scan* (middle) and the *end of scan* (bottom). The images on the left represents the results of using marker 2 and the images on the right represents the results of using marker 3.

3.6 Conclusion

The aim of this chapter was to determine if the *marker placement criteria* of the CT scanner SPG system can be satisfied given the fixed camera positions in the CT scanner room. This criteria requires at least three identifiable markers to be visible to at least one camera pair in the CT scanner room. The results from the simulations indicate the areas on the oval model that are visible to at least one camera pair. The shaded regions in Figure 3.10 show the visible areas on the oval model positioned when it is positioned in each of the three reference mask positions. The regions on the left represent those areas that are visible to the front cameras while the areas on the right represent those areas visible to the back cameras. The overlap of these areas is shown in Figure 3.11. The *marker placement criteria* will be satisfied when placing three identifiable markers in these areas of the mask. This is true for the oval model in all three the reference mask positions.

Limiting the markers to the shaded regions in Figure 3.11 over constrains the

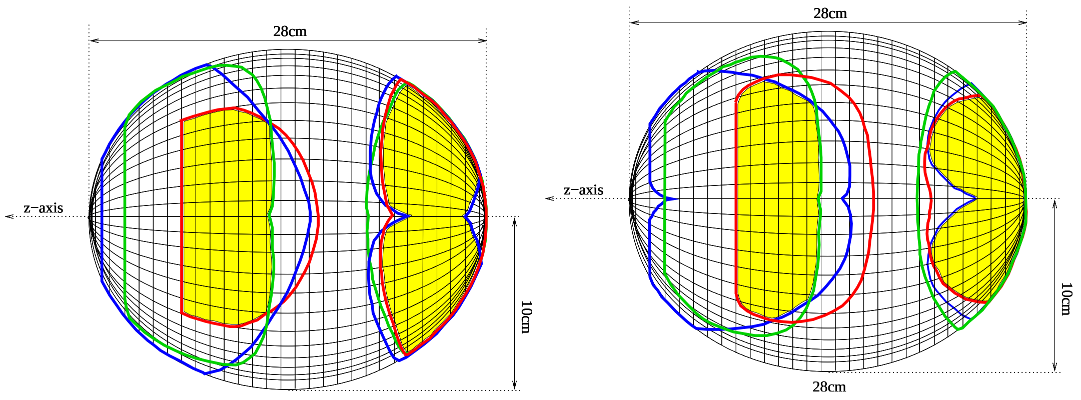


Figure 3.11: The area visible to both the front and the back cameras for the oval model positioned in each of the three reference mask positions. The left image indicates the results for marker 2 and the right image indicates the results for marker 3.

marker placement problem. Consider the following setup: The three cameras that are positioned in front of the CT scanner monitor the patient when the patient's mask is positioned between the *start of scan* and the *CT study reference position* (first part of a scan). Similarly, the three cameras that are positioned behind the CT scanner monitor the patient when the patient's mask is positioned between the *CT study reference position* and the *end of scan* (second part of a scan). In such a setup, the usable areas are shown in the shaded regions in Figure 3.12.

Markers positioned in the areas indicated in Figures 3.11 and 3.12 satisfy the

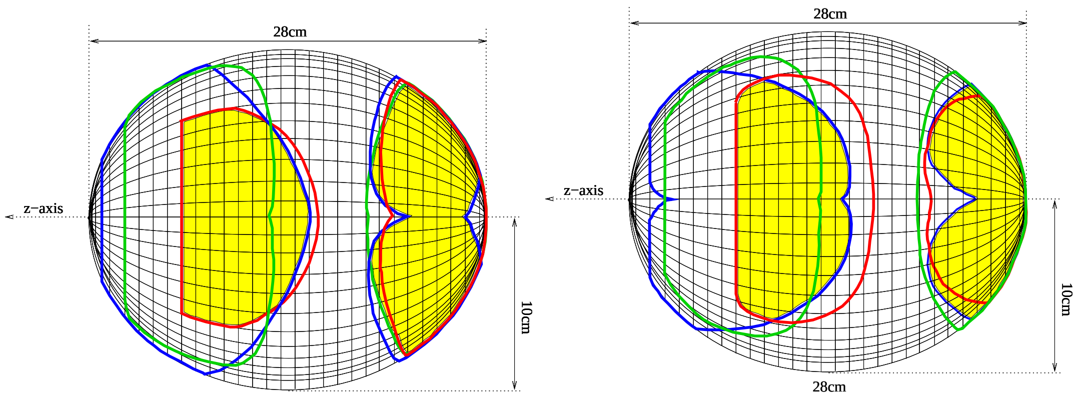


Figure 3.12: The expanded area visible to both the front and the back cameras in a configuration where the front cameras observe the oval model in the first part of the CT study and the back cameras observe the oval model in the latter part of the CT study. The left image indicates the results for marker 2 and the right image indicates the results for marker 3.

minimum requirements of the CT scanner SPG system. Replacing the preliminary marker identification algorithm with a more robust marker identification algorithm is expected to increase the usable area.

Chapter 4

Camera placement in the Treatment vault

4.1 Introduction

The Treatment SPG system is implemented in the treatment room and consists of nine CCD cameras and a computer with stereophotogrammetric software. This system uses the relationship between the position of the patient's mask and the position of the target lesion (this relationship was established using the CT scanner SPG system) to determine the position of the patient in the prescribed treatment position. The Treatment SPG system also monitors patient motion during treatment. It instructs the safety system to block the treatment beam when the patient moves outside the allowed treatment area.

The Treatment SPG system also requires a minimum of three identifiable markers to be visible to at least one camera pair in order to determine the unique 3D position of the patient. The position of the markers on the mask is such that it satisfies the constraints of the CT scanner SPG system. On the other hand, the cameras from the Treatment SPG system can be placed fairly freely. Ideally, they will be placed to ensure maximum visibility of the patient's mask and therefore also maximize the number of visible markers. Finding these optimal camera positions is an optimization problem. Each different camera placement configuration should be evaluated over the entire search space (all combinations of the variables that define the problem) and then compared. The camera placement configuration with the highest value will then give the optimal camera positions.

A simulation routine was developed to compare the different camera placement configurations in the treatment vault. The merits of these camera positions were evaluated using a suitable cost function, defined in Section 4.1.1. This cost function considered both the visibility of the patient's mask and the error in determining the position of the markers as a result of the relative camera positions. The camera positions were constrained by the two treatment beam nozzles, the movements of the portal x-ray image acquisition system and resolution restrictions. The extent to which these constraints affect the camera positions is discussed in Section 4.2. The problem's multidimensional search space is defined by the evaluation parameters and the camera positions parameters. The evaluation parameters are used to calculate the cost function and include: the type of patient mask (face or body mask) and the camera combinations used. We optimize the camera position parameters. They are: the distance between each camera and the patient's mask, the height of each individual camera rig and the position of each camera on the camera rigs. These variables are discussed in Section 4.3.

The simulation routine does an exhaustive search when optimizing the position of the cameras. This technique is preferred over standard optimization methods such as the conjugate gradient method, because we want to ensure coverage of the complete search space. Also, since this is a high dimensional problem, local minima are likely to be a problem with conventional optimization techniques. Other external factors are also important in the final decision, therefore we need to have information not just of the optimal camera position but also of other possibly "good enough" camera configurations, as well as information about the sensitivity of these camera positions to errors due to manufacturing issues, etc. We discuss

the simulation routine in Section 4.4. The implementation detail and correctness testing of this routine are discussed in Sections 4.5 and 4.6 respectively.
The simulation resulted in a number of possible solutions to the camera positioning problem. As the constraints used, differ between the simulations, these possible solutions differ and need to be evaluated on criteria such as flexibility and ease of manufacturing. Some results are given in Section 4.7 and the full set is provided in the *./SimResults* directory on the accompanied compact disc. We summarize the results in Sections 4.8, discuss the other criteria in Sections 4.9 and conclude in 4.10.

4.1.1 Cost function

There are two types of treatment mask, namely the face mask and the body mask. Both masks are covered with a number of markers. For any given mask, each camera can potentially view an area on the mask containing a subset of the markers. Reconstruction of the 3D coordinates of a marker on the mask requires that the marker be visible to a camera pair. When the coordinates of a sufficient number of markers can be reconstructed, a grid of the markers can be defined. This grid will then be used to represent the mask.
Maximizing the area of the mask that is visible to a camera combination that is used in the reconstruction of the 3D coordinates, increases the accuracy of the grid by enhancing the number of grid points. The relative position between the cameras in such a camera combination also affects the accuracy of the reconstruction. The weight function (ω) defined below is a measure of the reconstruction error from different relative camera positions. A tradeoff exists between maximizing the visible area of the mask and minimizing the error in the reconstruction process. The cost function attempts to trade these contradictory goals off against each other. Therefore, maximizing the cost function will attempt to maximize the visible area of the mask while simultaneously minimizing the error in the reconstruction.
A minimum of three cameras will be used to monitor a patient during treatment in

the new treatment vault. For each camera triplet, three different camera pairs can be used to reconstruct the marker coordinates. Consider the three cameras i, j and k, that belong to any camera triplet, i.e. where $i, j, k = 1, ..., 9$ and $i \neq j \neq k$. The object area visible to each camera is denoted by A_x, with $x = i, j, k$. Therefore, the overlap of the areas visible to pairs of these cameras is

$$A_{xy} = A_x \bigcap A_y, \quad x = i, j, k, \;\; y = i, j, k \text{ and } i \neq j \neq k \tag{4.1}$$

Likewise, the overlap of the areas visible to all three cameras is given by

$$B_{ijk} = A_i \bigcap A_j \bigcap A_k \tag{4.2}$$

For such a camera triplet, the cost function is defined as

$$\begin{aligned} F_{ijk} &= \omega_{ij}\left(A_{ij} - B_{ijk}\right) + \omega_{ik}\left(A_{ik} - B_{ijk}\right) + \omega_{jk}\left(A_{jk} - B_{ijk}\right) \\ &+ \frac{1}{3}\left(\omega_{ij} + \omega_{ik} + \omega_{jk}\right) B_{ijk}, \end{aligned} \tag{4.3}$$

where ω is a weight function for each camera pair. This weight function is discussed below. The areas represented by each term in the cost function is illustrated in Figure 4.1.

The appropriateness of this cost function can easily be illustrated by considering the following two extreme cases. When all three cameras view the same area of the mask, the first three terms will be zero and F_{ijk} will reduce to $\frac{1}{3}\left(\omega_{ij} + \omega_{ik} + \omega_{jk}\right) B_{ijk}$. The factor $\frac{1}{3}\left(\omega_{ij} + \omega_{ik} + \omega_{jk}\right)$ ensures that the area B_{ijk} is never added more than once in the cost function, even with all three weight function values at a maximum of one. In the other extreme case there is no common area of the mask visible to the camera triplet. In this case all the B_{ijk} terms will be zero and only the A_{ij}, A_{jk} and A_{ik} terms will contribute to the value of the cost function.
In some instances the use of a camera quadruplet will also be required. We can define the cost function for a camera quadruplet in a similar way. For this formulation consider the four cameras i, j, k and l where $i, j, k, l = 1, ..., 9$ respectively.

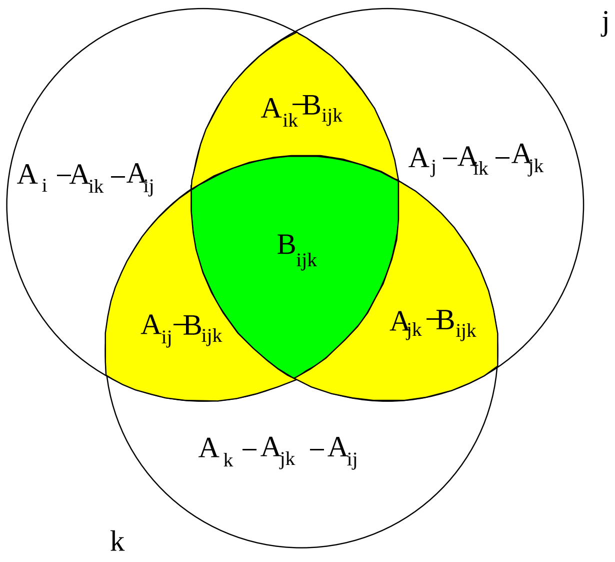


Figure 4.1: Graphical illustration of overlapping areas of a camera triplet. The overlapping areas are shaded and each region represents a term in the cost function.

Again the object area visible to each camera is denoted by A_x, with $x = i, j, k, l$. The overlap of areas visible to camera pairs and triplets is given by equations 4.1 and 4.2, respectively. The overlap of the areas visible to all four cameras is

$$C_{ijkl} = A_i \bigcap A_j \bigcap A_k \bigcap A_l. \tag{4.4}$$

The definition of the cost function for a camera quadruplet is

$$\begin{aligned}
C_{ijkl} &= \omega_{ij}\left(A_{ij} - B_{ijk} - B_{ijl} + C_{ijkl}\right) \\
&+ \omega_{ik}\left(A_{ik} - B_{ikl} - B_{ijk} + C_{ijkl}\right) \\
&+ \omega_{il}\left(A_{il} - B_{ilk} - B_{ijl} + C_{ijkl}\right) \\
&+ \omega_{jk}\left(A_{jk} - B_{ijk} - B_{jkl} + C_{ijkl}\right) \\
&+ \omega_{kl}\left(A_{kl} - B_{ikl} - B_{jkl} + C_{ijkl}\right) \\
&+ \omega_{jl}\left(A_{jl} - B_{jkl} - B_{ijl} + C_{ijkl}\right) \\
&+ \frac{1}{3}\left(\omega_{ij} + \omega_{ik} + \omega_{jk}\right) B_{ijk} \\
&+ \frac{1}{3}\left(\omega_{ij} + \omega_{il} + \omega_{jl}\right) B_{ijl} \\
&+ \frac{1}{3}\left(\omega_{ik} + \omega_{il} + \omega_{kl}\right) B_{ikl}
\end{aligned}$$

$$
\begin{aligned}
&+ \frac{1}{3}\left(\omega_{jk} + \omega_{jl} + \omega_{kl}\right) B_{jkl} \\
&+ \frac{1}{6}\left(\omega_{ij} + \omega_{ik} + \omega_{il} + \omega_{jk} + \omega_{jl} + \omega_{kl}\right) C_{ijkl}.
\end{aligned} \tag{4.5}
$$

Figure 4.2 shows the areas of overlap for a specific configuration of four cameras. Each area represents one of the terms in the cost function. The appropriateness of this cost function can again be illustrated by considering the two extreme cases. When all four cameras view the same area, only the last term will determine the area. In this case all camera pairs, camera triplets and camera quadruplets will view the same area (C_{ijkl}). The multiplication factor in front of this area serves the same purpose as the multiplication factor before the B_{ijk} term of the cost function for the camera triplet above. In the other extreme case there is no common area of the mask visible to any camera triplet or camera quadruplet. In this case all the B_{xyz} and C_{ijkl} terms will be zero and only the A_x terms will contribute to the value of the cost function.

The cost function defined here not only gives us information on the total area of the mask that is visible to a particular combination of cameras, but also on the areas of the mask that is visible to each pair of cameras that belong to this combination.

A cost function defined in such a manner has a natural tendency to draw cameras closer to each other. This is a result of trying to maximize the areas of overlap between the different camera combinations. The cost function will be at a maximum when all the cameras are at the same position and view the same area of the mask. This is clearly not a good camera configuration. We introduce a weight function (ω) to rectify this behavior of the cost function.

Weight function

One of the biggest challenges in stereo reconstruction is determining an image-point correspondence [17] [21]. This process requires both depth recovery and feature matching. Depth recovery involves determining the distance between an object

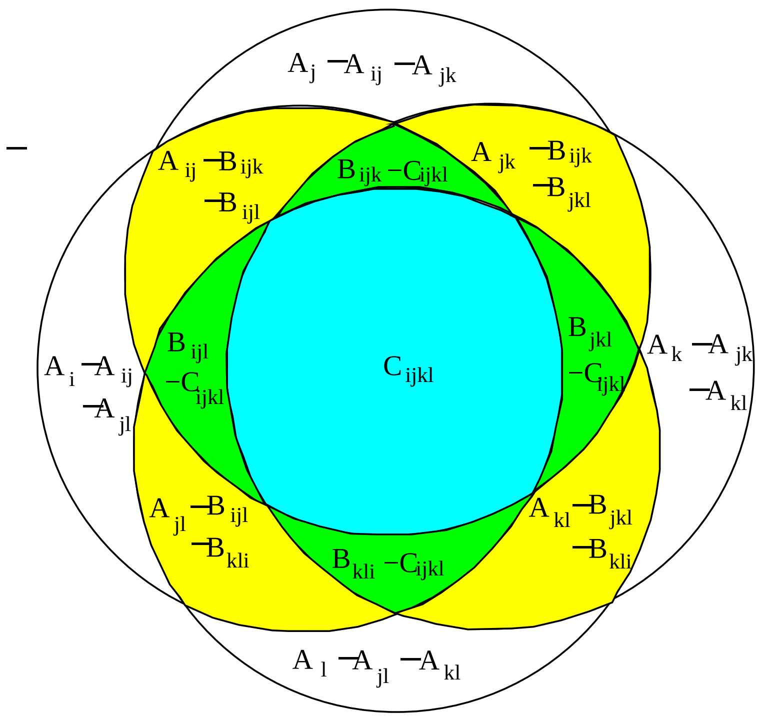


Figure 4.2: Graphical illustration of overlapping areas of a camera quadruplet. The overlapping areas are shaded and each region represents a term in the cost function.

and the cameras by means of triangulation. Feature matching involves finding a point-to-point correspondence in a pair of images captured from stereo cameras, i.e., finding a correspondence between features from two images that corresponds to the same physical point. Unfortunately, a trade-off exists between these two processes. A short baseline (narrow separation angle ϕ_c) implies that the estimated distance between the cameras and the object will be less precise due to narrow triangulation. A wider baseline improves this distance estimation, but enlarges the areas on both images that must be searched to find the same object point. As a result, the matching of points (or features) is more difficult, thus the possibility of a false match is greater.

Borghese and Ferrigno [4] used the relation between the accuracy of depth recovery and the ease of feature matching to determine the optimal angle between two cameras in a stereo rig. Such an optimal angle results in the best accuracy during

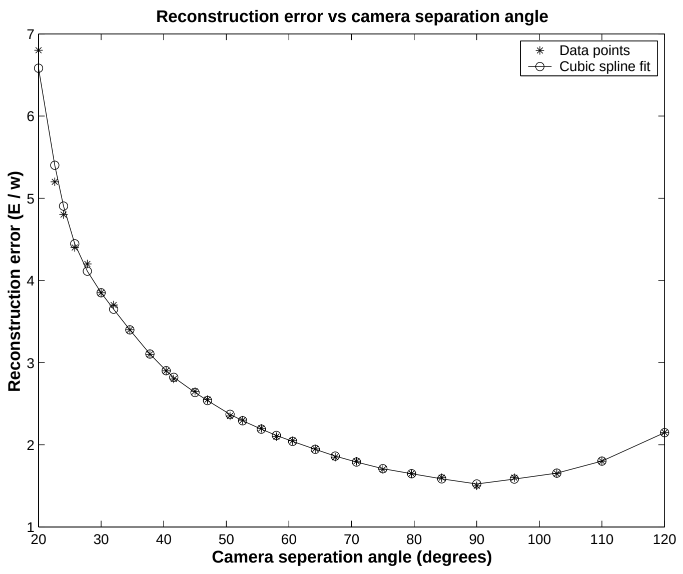


Figure 4.3: Normalized error in reconstruction of 3D coordinates at different camera separation angles (ψ).

3D reconstruction. A set of data points, similar to those obtained by Borghese and Ferrigno, is plotted in Figure 4.3. This graph indicates the error (E) in reconstructing the 3D coordinates of a marker normalized by the quantization error (w) and plotted as a function of the camera separation angle (ψ_j). Each data point therefore indicates the maximum distance (in mm) between the reconstructed position of the marker and its actual position after being normalized by $w = 0.5$ (equivalent to a spatial resolution of two pixels per millimeter on a camera). These error values indicate the maximum error for a given camera separation angle. This maximum error is a minimum of 1.5238 mm at a camera separation angle of 90° [4]. The corresponding cubic spline interpolant (Γ_j) for the set of data points $\psi_0 < \psi_1 < \cdots < \psi_n$ is defined in terms of the following cubic polynomial

$$\begin{aligned} \Gamma_j(\psi) &= a_j + b_j(\psi - \psi_j) + c_j(\psi - \psi_j)^2 + d_j(\psi - \psi_j)^3 \\ & \quad \text{for each } j = 0, 1, ..., n-1 \end{aligned} \tag{4.6}$$

where a_j, b_j, c_j and d_j are the coefficients of the function evaluated at the data points. The cubic spline resulted in a good fit to the data set and can easily be implemented with a lookup table. This interpolant is a minimum at the optimal camera separation angle of $\psi = 90°$. Any other camera separation angles result in an exponential increase in the error of reconstructing 3D coordinates. The rapid increase in reconstruction error motivates the need to include this error factor in the cost function formulation.

The weight function (ω) is defined in terms of Γ_j as

$$\omega(\psi) \quad = \quad 1 - \frac{\Gamma_j(\psi)}{max\{\Gamma_j(\psi)\}}. \tag{4.7}$$

for $j = 0, 1, ..., n-1$ and is shown in Figure 4.4.

Incorporating this weight function in the cost function has two advantages. Firstly,

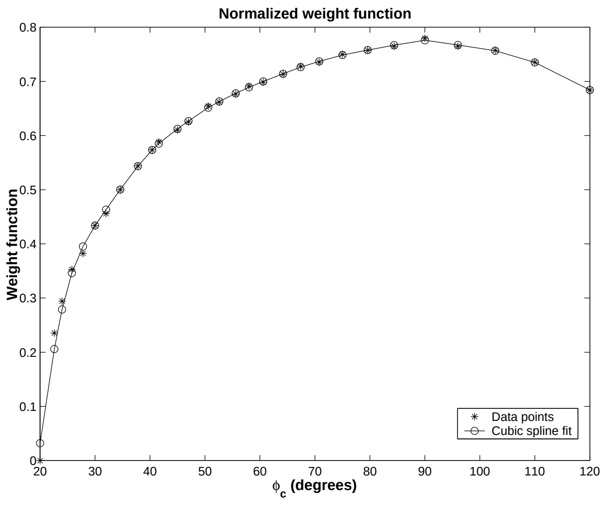


Figure 4.4: Weight function (ω) vs angle (ψ).

such a cost function incorporates the error of reconstructing 3D coordinates for a stereo camera rig. Secondly, without the weight function the optimal cost function value would be obtained from cameras that are positioned at the same point. Maximizing both the cost function and the weight function simultaneously result in an optimal camera separation angle of $0° < \psi < 90°$.

4.2 Treatment vault setup

The layout of the treatment vault defines the boundaries to the search space of the camera positioning problem. A well defined search space simplifies the problem. The different structures that place boundaries on the search space will be discussed in this section.

4.2.1 Treatment vault coordinate system

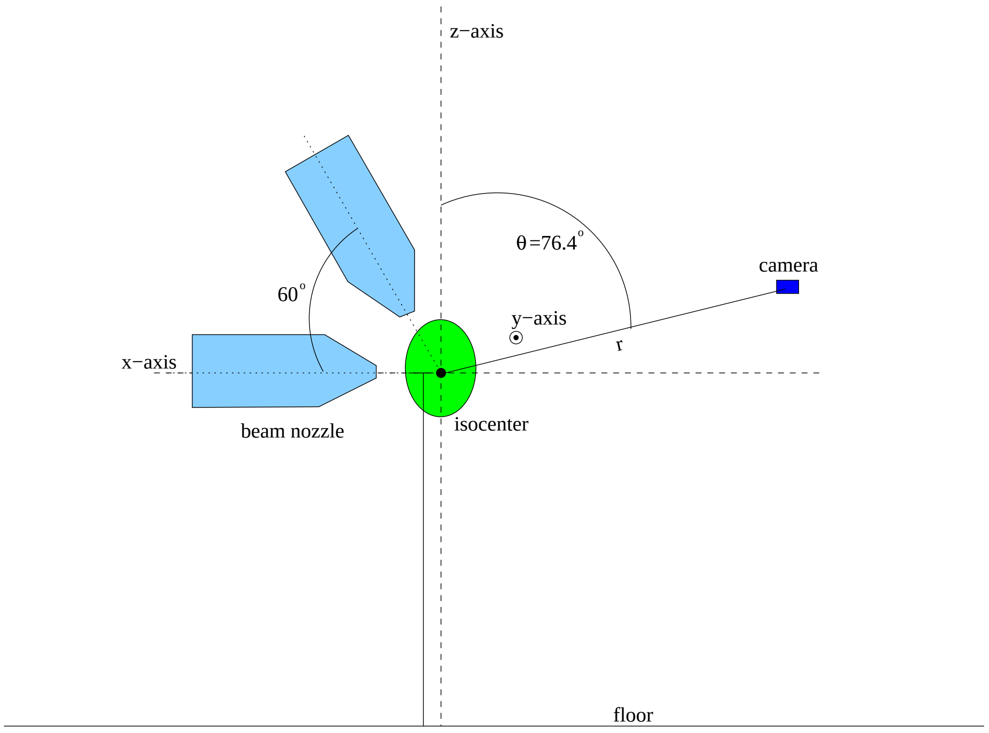


Figure 4.5: Lateral view of the treatment vault layout around the isocenter.

The layout of the treatment vault can conveniently be described in terms of the spherical coordinate system displayed in Figure 4.5. The position of a point in the treatment vault is identified by a triple (r, ϕ, θ), where r is the length of the position vector, the polar angle $(\theta \in [0, \pi])$ is the angle between r and the *z-axis*, and the azimuth angle $(\phi \in [0, 2\pi])$ is the angle between the projection of r onto the *xy-plane* and the *x-axis*. Only r and θ are indicated in Figure 4.5. Also indicated is the origin of this coordinate system, the isocenter.

4.2.2 Beam nozzle setup

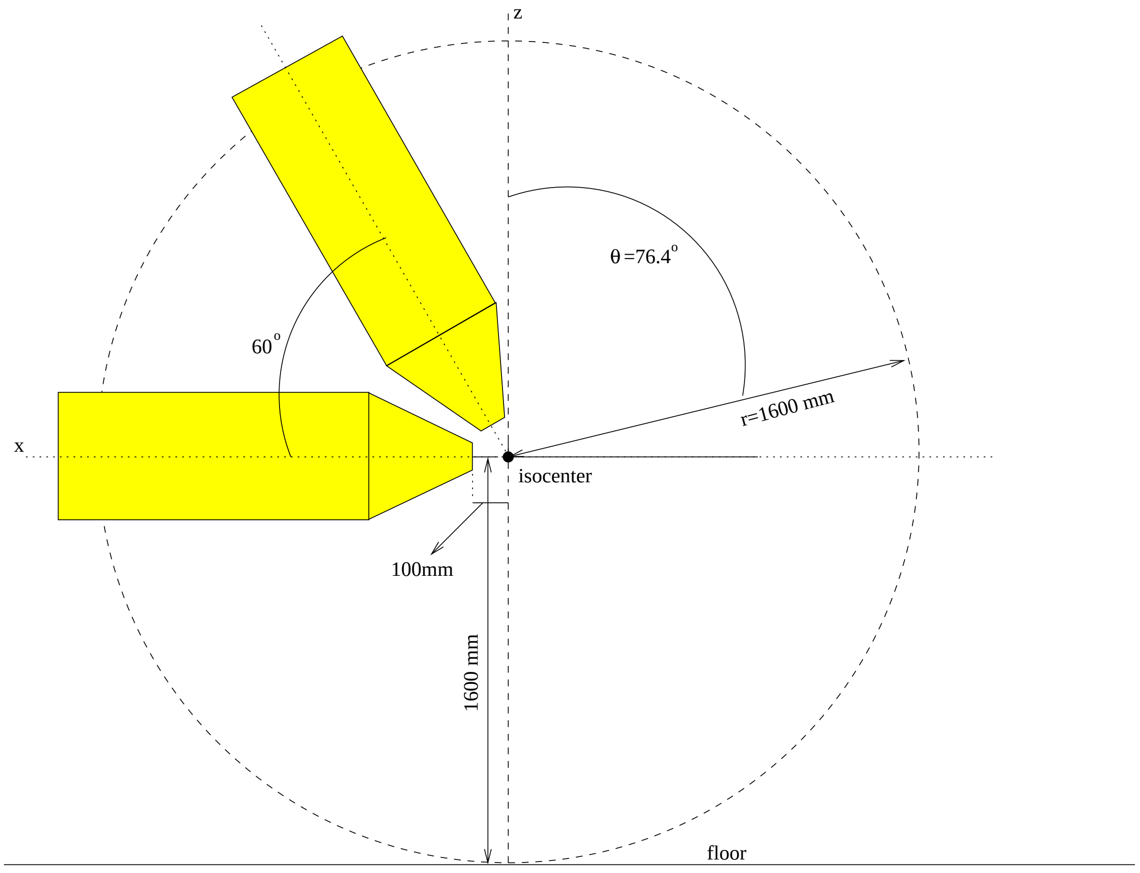


Figure 4.6: The position of the beam nozzles relative to the isocenter.

The height of the isocenter in the treatment vault is approximately 1600 mm. The beam axis of the horizontal beam-line coincides with the *x-axis*. A second, skew beam-line is positioned in the *xz-plane* at a 60° angle off the horizontal. Both beam-lines share a common treatment isocenter, which is defined as the point where the central axes of the two beam-lines intersect. The last section of each nozzle ends in a telescopic snout which allows the distance between the collimator and patient to be adjusted. The collimator is fitted at the end of the telescopic snout and determines the shape of the proton beam. When the telescopic snout is fully extended, the gap between the treatment collimator and the isocenter is approximately 100 mm.

4.2.3 Portal x-ray

A portal x-ray image acquisition system will be used to take images of the patient in the treatment position. These images are used to confirm that the patient is in the correct position. This system is lowered down from the roof of the treatment vault. The positioning of the cameras must allow for unrestricted movement of the image acquisition system down to a point behind the patient. The portal x-ray image acquisition system's dimensions are shown in Figure 4.7.

A top view of the position of the image acquisition system relative to the beam

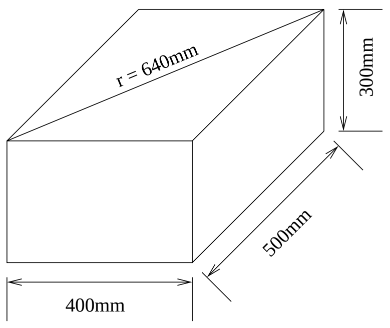


Figure 4.7: The dimensions of the portal x-ray image acquisition system.

nozzles and the isocenter is shown in Figure 4.8. A limit of 838.2 mm is placed on the minimum radius allowed for the camera rigs. A camera rig radius larger than this minimum radius will prevent collision between the camera rigs and the image acquisition system.

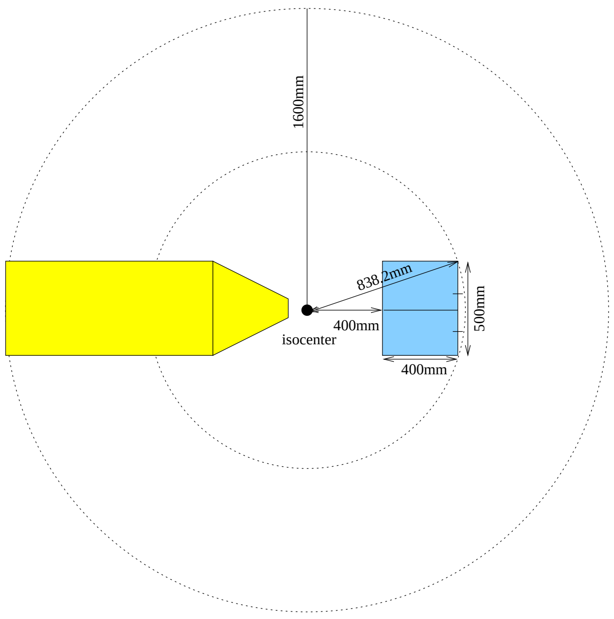


Figure 4.8: A top view of the position of the portal x-ray image acquisition system relative to the beam nozzles and the isocenter.

4.2.4 The camera rigs setup

In theory, the cameras can be placed anywhere in the treatment vault. Practical and manufacturing constraints limit this search space by restricting the camera positions to, at most, two camera rigs. Both these camera rigs will be permanently mounted in the treatment vault. Their respective sizes as well as the distances at which they are mounted from the floor of the treatment vault might be different. For the purpose of the simulation, we assume circularly shaped camera rigs. This allow us to specify radii for both camera rigs which will be used to determine the minimum and maximum heights allowed for each camera rig. The radius (R) for a camera rig is given by

$$R = 1600 \times \sin(\theta) \tag{4.8}$$

where the height of the rig is indicated in terms of the polar angle θ. The portal x-ray image acquisition system restricts the minimum radius of a camera rig to 838.2 mm (see Section 4.2.3). This minimum radius results in a polar angle of 31.6°

which corresponds to a maximum height of 1362.78 mm above the isocenter. The minimum height allowed for a camera rig is 533.6 mm above the isocenter. Cameras positioned below this height will impose restrictions on the movement of the personnel attending to the patient in the vault. This height value translates to a polar angle of 70.5° and a maximum camera rig radius of 1508.4 mm. The height angles allowed for each camera rig are therefore: $\theta_1 \in [31.59°, 70.52°]$ for camera rig one and $\theta_2 \in [31.59°, 70.52°]$ for camera rig two.

Two different camera rig setups will be considered in the simulations. In the first setup, the cameras mounted on both camera rigs will be kept at a fixed distance (1600 mm) from the isocenter. This forces the radii of the camera rigs to vary when they are positioned at different heights above the isocenter. This camera rig setup is illustrated in the top image of Figure 4.9 where θ_1 and θ_2 indicate the height of camera rig one and two respectively. The second camera rig setup that will be considered in the simulations requires the radii of both camera rigs to be equal. This setup forces the cameras on the top rig to be further away from the isocenter than 1600 mm. This setup is illustrated in the bottom image of Figure 4.9. The outer half circle indicates the maximum distance of 1950 mm allowed between a camera and the isocenter (see Section 4.3.2).

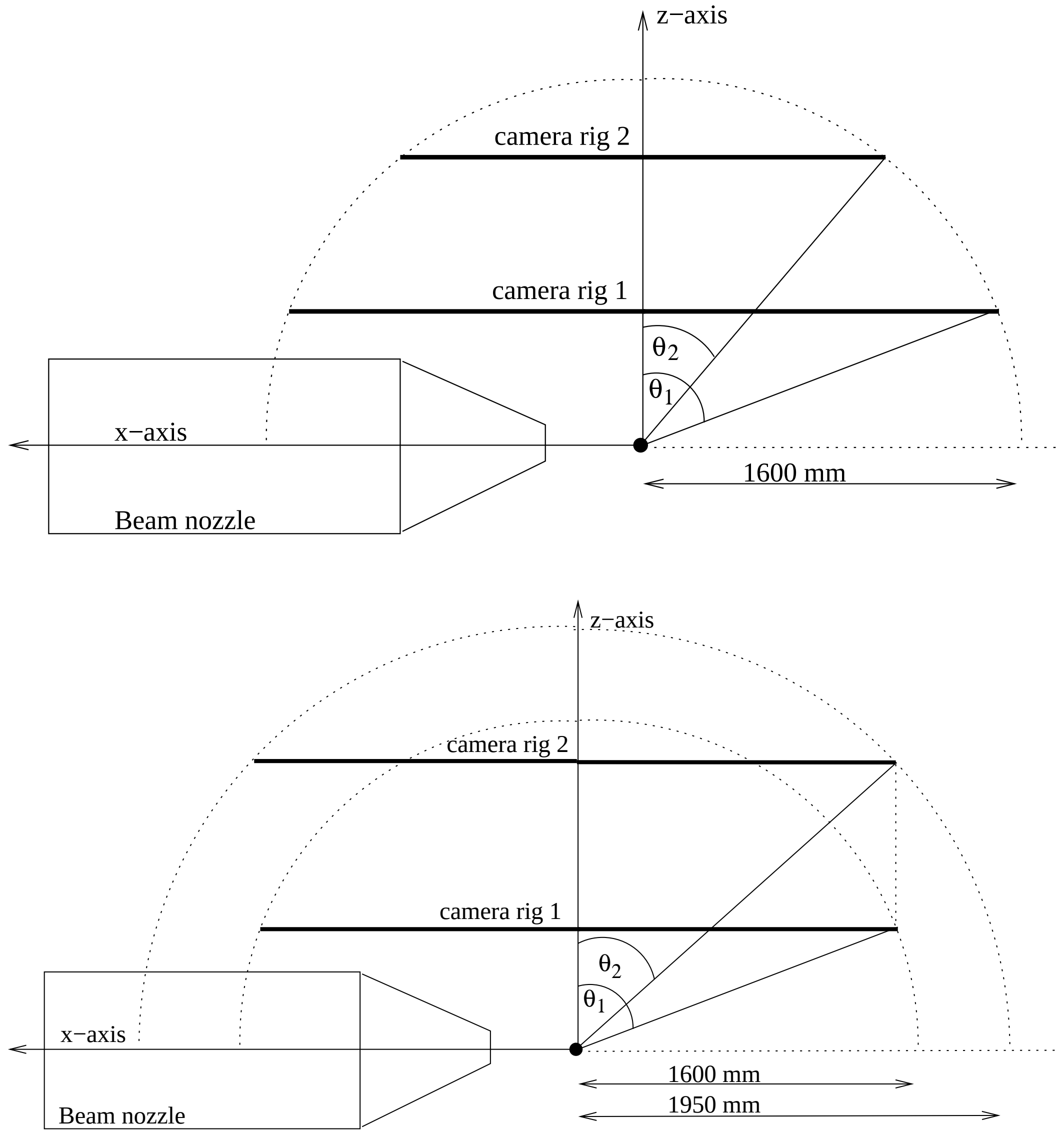


Figure 4.9: The two different camera rig setup that with be considered in the simulations. The image at the top shows all the cameras at fixed distances from the isocenter while the image at the bottom shows the camera rigs with equal radii.

4.2.5 The camera setup

A total of eight or nine cameras, depending on whether one or two beam delivery systems are considered, will be mounted on the two camera rigs. All these cameras will be focused on the isocenter. The optical axis of each camera connects the pinhole of the camera with the isocenter. The distance between the isocenter and the camera pinhole is restricted to a minimum distance of 1600 mm and a maximum distance of 1950 mm. Each camera has a focal length of $f = 30$ mm which results in a horizontal view angle of $\nu_h = 12.18°$ and a vertical view angle of $\nu_v = 9.15°$. The area of an object that is projected onto the CCD of a camera is the area of the object that is visible to that particular camera.
A diode ring is fitted on each camera. These diodes are responsible for emitting directed light onto the retro-reflective markers on the mask. The nature of the retro-reflective material is such that it reflects directed light along the same path with very little scattering. The light reflected from the retro-reflective markers enhances their visibility to the cameras. By using dichromatic filters and diodes with the same frequency range as the filters, we can eliminate a significant portion of the background noise in the images. This simplifies marker detection.
A symmetrical placement configuration is selected for both the eight and the nine camera setups. These symmetrical setups result in equal separation angles between consecutive cameras, the advantage of which becomes apparent when considering any other camera configuration. For example, consider changing the position of one camera from the symmetrical camera configuration. This move results in an increase of one camera separation angle and a decrease of another. The camera pair separated by the smaller of these two angles will result in a larger or smaller cost function value, depending on whether large or small camera separation angles result in the smallest reconstruction error. Conversely, the camera pair separated by the larger angle will have the opposite effect on the value of the cost function. As a result, it will always be possible to find a mask orientation that results in a lower cost function value when the cameras are positioned asymmetrically.

In both the eight and the nine camera setups, consecutive cameras are positioned on different camera rigs to ensure a camera placement that is symmetrical with respect to the x-axis. In the nine camera setup, the positions of cameras 1 and 9 are indicated by the azimuth angle ϕ_b while the positions of the remaining cameras are indicated by the azimuth angle ϕ_c. The value of ϕ_c is derived from the value of ϕ_b with Equation 4.9. In the eight camera setup, the first camera is positioned directly above the horizontal beam nozzle on the x-axis. The positions of the remaining cameras are specified by $\phi_c = 45°$. The camera positions in these two setups are shown in Figure 4.10. A side view of the nine camera setup is shown in Figure 4.11.

$$\phi_c = \frac{360 - 2\phi_b}{8} \tag{4.9}$$

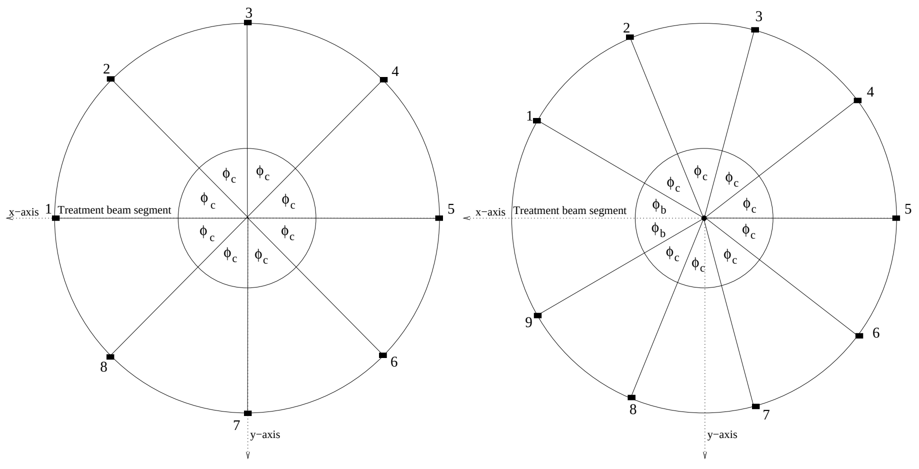


Figure 4.10: A top view of the eight (left) and nine (right) cameras in a symmetrical camera placement configuration on both camera rigs.

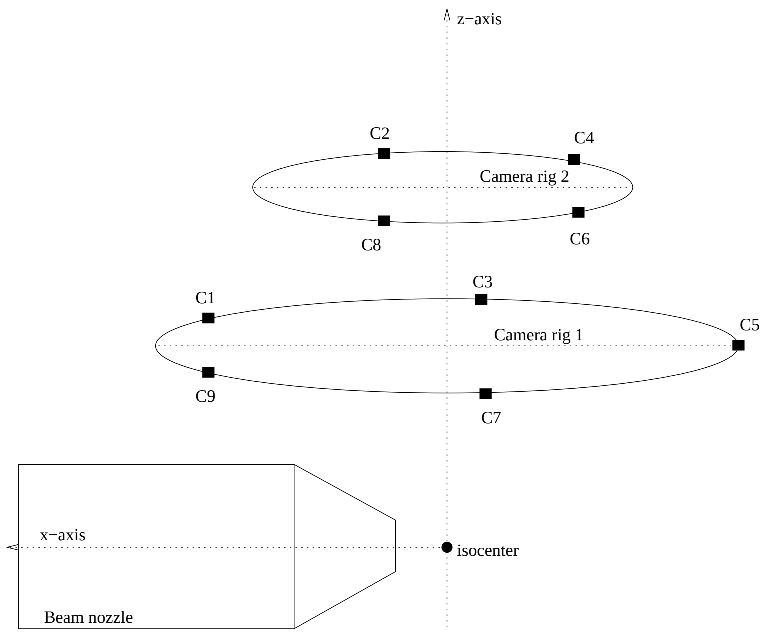


Figure 4.11: A top view and a side view of the camera positions on the camera rigs. C1 to C9 indicate camera positions 1 to 9.

4.3 Restrictions and assumptions

The layout of the treatment vault defined the search space for the optimal camera positions. We also defined a cost function which will be used to compare different camera positions. The remainder of this section describes the conditions under which the cost function will be evaluated. These conditions are governed by restrictions on the movement of the treatment chair and couch as well as restrictions to the placement of cameras. These restrictions are described in Sections 4.3.1 and 4.3.2 respectively.

4.3.1 Restrictions to the patient's pose

The patient's mask is attached to either the treatment chair or the treatment couch. Either the treatment chair or the treatment couch positions the patient in the default treatment position which places the target volume (i.e. a brain tumor) at the isocenter of the treatment vault. As a result, the patient mask is also positioned in close proximity to this reference point. For this reason we can ignore translation and consider only rotation of the mask during treatment. Movement of a treatment mask is restricted by the movement capabilities of the chair or the couch. In turn, while the patient is in the default treatment position, the chair and couch are restricted to the range of rotations allowed around the three Cartesian axes. These restrictions are described next.

Restrictions to the movements of the treatment chair

The restrictions to the movement of the treatment chair are described in terms of the conventional right hand coordinate system shown in Figure 4.5. The directions of rotations around each axis are shown in Figure 4.12. In the default treatment position, the treatment chair is facing the treatment beams (in the direction of the positive x-axis) with its top pointing in the direction of the positive z-axis.

The movement restrictions of the treatment chair are described relative to this default position. The rotation of the chair around the y-axis is restricted to $\theta_y^F \in [-10°, 100°]$. These boundaries result from the movement capabilities of the chair as well as the comfort of the patient during treatment. The same factors restrict its rotation around the x-axis to $\theta_x^F \in [-15°, 15°]$. Rotation around the z-axis depends on the current value of θ_y^F. Small values of θ_y^F indicate that the backrest of the chair is in an upright position. There is no restriction on its rotation around the z-axis ($\theta_z^F \in [0°, 360°]$) when the chair is in such a configuration. Large values of θ_y^F indicate that the backrest of the chair is in a "flatter" position and therefore more similar to the treatment couch. These chair configurations result in a collision between the chair and treatment beam nozzles for some rotation angles around the z-axis (see Figure 4.13). The restrictions to the treatment chair in such a configuration is the same as the restrictions to the treatment couch in the next section.

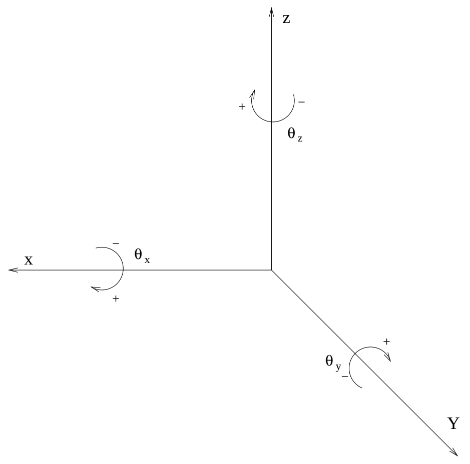


Figure 4.12: The direction of rotation considered during specification of the rotation boundaries.

Restrictions to the movements of the treatment couch

The movements of the treatment couch are more restricted than the movement of the treatment chair. In the default treatment position, the treatment couch is in a horizontal position with the feet end of the couch pointing to the treatment beams (in the direction of the positive x-axis). The movement restrictions of the treatment couch are described relative to this default position. The movement capabilities of the couch as well as the comfort of the patient during treatment restrict its rotations around the x-axis to $\theta_x^B \in [-15°, 15°]$ and the rotations around the y-axis to $\theta_y^B \in [-15°, 15°]$. Z-axis rotations in the region $\theta_z^B \in [-36.56°, 36.56°]$ will result in a collision between the couch and the beam nozzles. The rotation around the z-axis is therefore restricted to $\theta_z^B \in [36.56°, 323.44°]$. This is the same range of rotations allowed around the z-axis for treatments using the treatment chair with large values of θ_y^F (see Section 4.3.1). These boundaries are shown in Figure 4.13.

4.3.2 Restrictions to camera positions

The position of the camera rigs and the cameras mounted on them were discussed in Sections 4.2.4 and 4.2.5 respectively. The restriction imposed on the maximum radius of these camera rigs is a direct consequence of the distance allowed between a camera and the isocenter. These distances directly determine the quality of the images captured by the cameras. Image quality depends on the optimal working distance of the diode rings and the spatial resolution of the cameras. The optimal working distance of a diode ring is obtained when the camera is place at a distance of 1600 mm from the isocenter [25] [26]. This distance also produces an acceptable spatial resolution of two pixels per millimeter. Although this is the optimal working distance for the diode rings, distances up to 1950 mm still yield usable results. This distance is considered the maximum allowed distance between a camera and

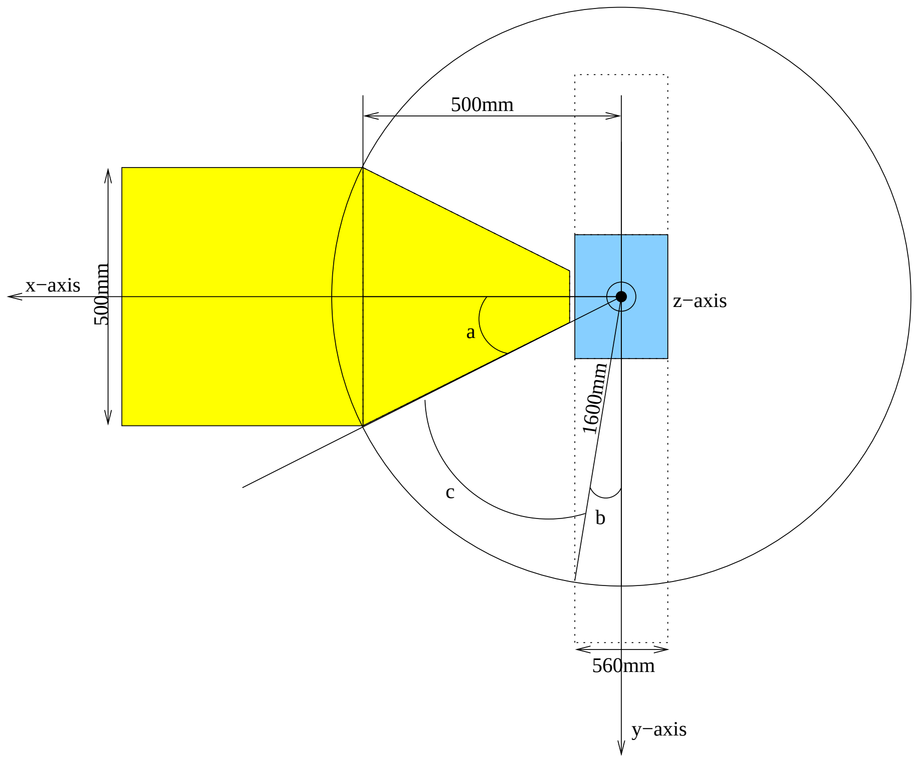


Figure 4.13: Top view of the treatment beam, showing the restrictions on the rotation of the couch around the z-axis.

the isocenter.

Images captured by a camera pair with a camera separation angle of $\phi_c < 15°$ is also undesirable. The diode rings of cameras separated by such small angles introduce hot spots in the illumination which affects the images captured by each camera. To avoid this phenomena the camera separation angles are restricted to angles greater than 15°.

The size and position of the beam nozzles place a restriction on the position of the two cameras closest to it. The closer these two cameras are positioned to the beam nozzles, the greater the portion of their view that is obscured. We need to find the best angle at with these two cameras have to be placed to either side of the beam nozzles. For this reason, the simulations will consider a number of positions for these two cameras in region close to the beam nozzles. The region selected for these

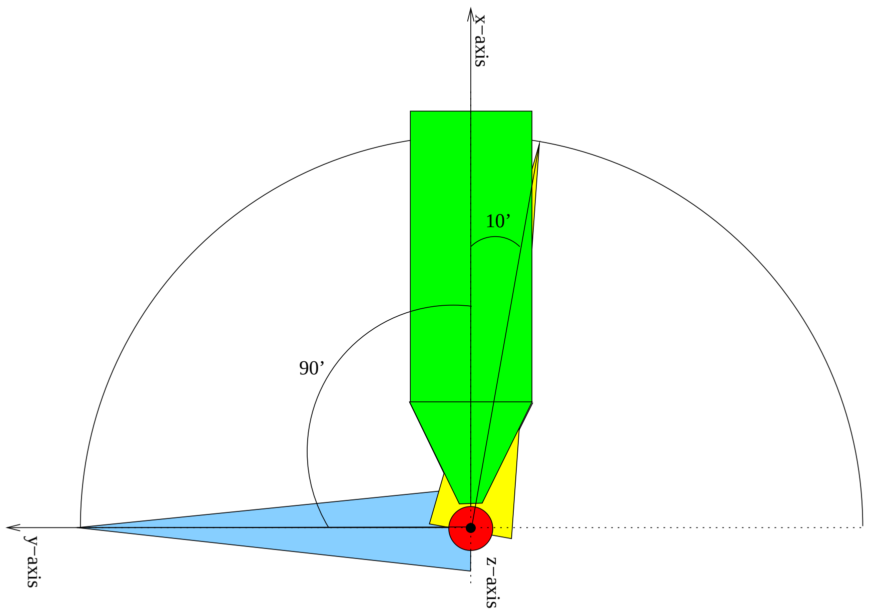


Figure 4.14: A top view of the beam nozzles around the isocenter, showing the restrictions on the camera placement.

camera positions is shown in Figure 4.14. The azimuth angle $\phi_b \in [10°, 90°]$ indicates the position of these two cameras to either side of the beam nozzles. The view of the cameras positioned at $\phi_b = 10°$ will have their views completely obscured by beam nozzles. The opposite is true for the cameras positioned at $\phi_B = 90°$. This ensures that the optimal positions of these two cameras are indeed included.

4.4 Simulation procedure

The aim of the simulations is to determine the optimal position of the cameras in the treatment vault. Such optimal camera positions depend on the number of treatment beams considered during the simulations. One set of simulations will consider both treatment beams while another set will consider only the horizontal treatment beam. Both these simulations will determine the optimal position of the cameras for both treatment masks. Throughout the rest of this document explicit reference will be made to specify the difference between each set of simulations.
The camera combinations that results from the camera setups are discussed in Section 4.4.1. Each camera setup allows the position of all the cameras to be specified by a single angle (ϕ_b). This angle is then used to calculate the separation angle between consecutive cameras in Section 4.4.2. The discrete values associated with the interval of each variable will be specified in Section 4.4.3.

4.4.1 Camera combinations

The camera combinations that will be used in the simulations are specified with respect to the camera positions in Figure 4.10. ${}^{8}C_{3} = 56$ different combinations of camera triplets are possible from a set of eight cameras. The cost function will be evaluated for each of these camera combinations in the simulations considering only the horizontal treatment beam. From the nine cameras used in the second set of simulations a total of ${}^{9}C_{3} = 84$ camera triplets as well as a single camera quadruplet will be considered. The camera quadruplet consists of the cameras at positions 1, 2, 8 and 9. This camera combination is included to deal with face mask orientations in the default position (see Section 4.3.1). At this mask orientation the beam nozzles affect the view of the mask the most.
Another advantage of considering symmetrical camera positions is that only half of the rotations around the z-axis need to be considered. This reduces the rotations around the z-axis of the face mask and body mask around to $\theta_z^F \in [0°, 180°]$ and

$\theta_z^B \in [40°, 180°]$ respectively.

4.4.2 Camera separation angles

Three angles describe the position of the cameras relative to each other. Angles ϕ_b and ϕ_c were described in terms of the azimuth angle ϕ in Section 4.2.5. The third angle, ψ, is the angle between the optical axes of a camera pair. This last angle changes as a function of the height of each camera rig (θ_1 and θ_2) above the isocenter. Although the relative position of the cameras is specified by ϕ_b and ϕ_c, the weight function that is used in the cost function calculations uses ψ as input. A formula is needed that expresses ψ as a function of the azimuth angle. The relationship between ϕ, ψ, θ_1 and θ_2 follows from Figure 4.15.

Figure 4.15 indicates camera M on the first camera rig at a height θ_1, camera P'

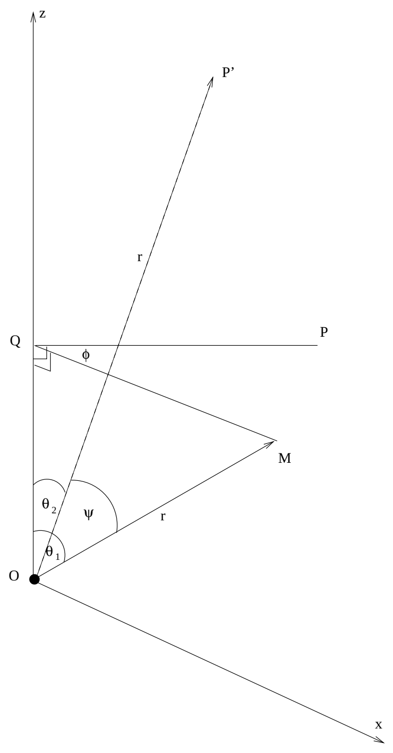


Figure 4.15: The relationship between angles ϕ, ψ, θ_1 and θ_2.

on the second camera rig at a height θ_2, the azimuth angle ϕ and the separation angle ψ between cameras M and P'. The coordinate system is chosen with QM parallel to the x-axis. Lines $\mathbf{OP}$ and $\mathbf{OP'}$ lie in the same plane. This setup ensures that $\psi = \phi$ when $\theta_1 = \theta_2$. The following triplets define the position vector of cameras M and P' as

$$\begin{aligned} \mathbf{M} &= (r\sin\theta_1, 0, r\cos\theta_1) \\ \mathbf{P'} &= (r\sin\theta_2\cos\phi, r\sin\theta_2\sin\phi, r\cos\theta_2) \end{aligned} \tag{4.10}$$

with

$$|\mathbf{M}| = |\mathbf{P'}| = r. \tag{4.11}$$

The scalar product of these two vectors expresses ψ in terms of θ_1, θ_2 and ϕ as:

$$\begin{aligned} \mathbf{M}\cdot\mathbf{P'} &= r^2\sin\theta_1\sin\theta_2\cos\phi + r^2\cos\theta_1\cos\theta_2 \\ &= |\mathbf{M}||\mathbf{P'}|\cos\psi \\ &= r^2\cos\psi \\ \cos\psi &= \sin\theta_1\sin\theta_2\cos\phi + \cos\theta_1\cos\theta_2 \end{aligned} \tag{4.12}$$

Equation 4.12 is used to calculate ψ in terms of the azimuth angle and vice versa. Since this equation is independent of the radius of the camera rigs, r, it can be used to calculate the camera separation angles for any of the specified simulation setups.

4.4.3 Discrete values of the variables

The different variables as well as the boundary values that define their allowed intervals were described in Sections 4.2, 4.3 and 4.4. It is impossible to consider all possible values, so we define points in these respective intervals where the cost function should be evaluated. The step size used in the discrete search space will determine the accuracy of the simulation results. Naturally there exists a trade-off between the accuracy of the results and the execution time of the simulation.

Greater accuracy requires longer simulation execution times. We reduce the computational cost of this simulation by adopting a two-phase iteration technique. In the first phase of this technique, relatively large step sizes define the discrete search space. As a result, this phase produces less accurate results but completes relatively quickly. The second phase repeats the simulations using a finer step size in a small region around the optimal values obtained in the first phase. This smaller step size results in an improvement to the accuracy of the first phase. Note that this approach can potentially exclude minima between the points chosen. However, if initial step size is small enough, minima excluded are uninteresting as they are overly sensitive to changes in the parameters.

The step sizes that define the discrete search space of the allowed mask orientations do not influence the accuracy of the camera positions. For this reason, these step sizes are kept constant for these phases. These step sizes are assumed to be sufficiently small to allow small enough changes between two consecutive mask orientations. The step sizes of the respective orientation variables are summarized in Table 4.1.

Variable	**Interval**	**Step size**
θ_z^F	$[0°, 180°]$	$20°$
θ_x^F	$[-10°, 100°]$	$20°$
θ_y^F	$[-15°, 15°]$	$15°$
θ_z^B	$[40°, 180°]$	$20°$
θ_x^B	$[-15°, 15°]$	$15°$
θ_y^B	$[-15°, 15°]$	$15°$

Table 4.1: Step sizes associated with the orientation variables of each mask during the first iteration phase.

The camera positions are specified by the three variables ϕ_b, θ_1 and θ_2. The first variable is used to derive the relative position of the cameras (specified by ϕ_c) on the two camera rigs while the last two specify the height of each respective camera rig. Manufacturing constraints limit the practically attainable accuracy, which limits the extent of realistic optimization. Also, a relatively small displacement in the values of either of these variables should not affect the area of the mask that is visible to a camera or the relative angle between two cameras in a major way. If neither of the components of the cost function is greatly affected by relatively small displacements, the cost function itself would also not be greatly affected. The remainder of this section determines to what extent the variables ϕ_b, θ_1 and θ_2 are affected by small displacements from the specified positions.

The accuracy of the values from each variable are dependent on the step size used in their respective intervals of allowed values. The accuracy is normally $\frac{\text{step size}}{2}$ to either side of the measured value. The relation between the ϕ_c and ϕ_b is shown in Equation 4.13. The following set of equations derive the accuracy of $\delta\phi_c$ which depends on the value of $\phi_b \in [10°, 90°]$ using a step size $\delta\phi_b$.

$$\begin{aligned} \phi_c \pm \delta\phi_c &= \frac{360° - 2(\phi_b \pm \delta\phi_b)}{8} \\ &= 45° - 0.25\phi_b \pm \frac{\delta\phi_b}{4} \\ &= \phi_c \pm \frac{\delta\phi_b}{4} \\ \delta\phi_c &= \pm\frac{\delta\phi_b}{4} \end{aligned} \tag{4.13}$$

The accuracy with which the camera heights are specified depends on the actual values of variables θ_1 and θ_2. The following equation shows the relation between the accuracy of a particular camera's height (δh) and the value of $\theta \in [\theta_1, \theta_2]$ using a step size $\delta\theta$.

$$\begin{aligned} \delta h &= d\cos(\theta) \pm d\cos(\delta\theta) \\ &= d\{\cos(\theta) \pm \cos(\delta\theta)\} \end{aligned} \tag{4.14}$$

where d is the shortest distance between the camera and the isocenter. Some calculated δh values are shown in Table 4.2. The values in this table indicate the accuracies at both the maximum and minimum allowed heights for the camera placement configurations where all the cameras are kept at a constant distance from the isocenter.

Step size	$\theta = 70.5°$	$\theta = 31.7°$
7.8°	199.0 mm	126.1 mm
3.9°	101.0 mm	60.1 mm

Table 4.2: The accuracy in the height of the camera positions as a result of two different step size for variables θ_1 or θ_2.

4.5 Implementation

The simulations are implemented and compiled as C++ programs that use OpenGL libraries and associated primitives for modeling. Elements of the vault modeled include: two beam nozzles, two camera rigs, nine cameras and both the face mask and the body mask. The beam nozzles are modeled using two standard primitives. Their telescopic snouts are modeled with cones while their back ends are modeled with cylinders (see Section 2.2.2). The circular shapes of the two camera rigs are modeled with a circle while the nine cameras are modeled using OpenGL's standard perspective camera models. This camera model provides features used to set both the intrinsic and the extrinsic camera parameters, resulting in very realistic camera views.

Both the face mask and the body mask are approximated using triangulated polygon meshes [13] [14] [15]. The triangulation routine uses the data points obtained from accurately scanning one face mask and one body mask currently used for treatments. Such data sets ensure that the representation of the masks used in the simulations are good approximations to real treatment masks. The information associated with the triangles that define each mask include: the 3D coordinates of each vertex point, the normal vector of the triangle and the area of each triangle.

There are three main algorithms involved in the calculation of the optimal camera positions. These are:

- *FaceMaskAlgorithm()*
- *BodyMaskAlgorithm()*
- *CombinationAlgorithm()*

The *FaceMaskAlgorithm()* and the *BodyMaskAlgorithm()* calculate the optimal camera positions for simulations involving the face mask and the body mask respectively. The only difference between these two algorithms is the variables used to describe the orientation of each mask. The basic structure of these two algorithms is described in Section 4.5.1.

The *CombinationAlgorithm()* combines the optimal camera positions from the *FaceMaskAlgorithm()* and the *BodyMaskAlgorithm()* and calculates the optimal camera positions associated with both masks. This algorithm is described in Section 4.5.2.

The *CostFunctionAlgorithm()* forms an integral part of the calculations of both the *FaceMaskAlgorithm()* and the *FaceMaskAlgorithm()*. This algorithm is responsible for calculating the cost function value for a particular camera combination and is described in Section 4.5.3.

4.5.1 The face mask and body mask algorithms

The *FaceMaskAlgorithm()* and the *BodyMaskAlgorithm()* iterate through every combination of the variables ϕ_b, θ_1, θ_2, θ_x, θ_y and θ_z. The last three of these variables specify the orientation of the relevant mask and are therefore different in each of these algorithms. The different combinations of the variables are implemented as nested loops where each loop iterates through all the values in the interval of a particular variable.

At every combination of the variables ϕ_b, θ_1, θ_2, θ_x, θ_y and θ_z an additional loop iterates through all the camera combinations while evaluating the cost function values associated with each. The maximum of these cost function values is assigned to the current mask orientation. This reflects the ability of the system to select the camera combination with the best view of the mask. The mask orientation with the lowest cost function value represents the worst mask orientation for the particular combination of ϕ_b, θ_1 and θ_2. This lowest cost function value is then assigned to the current combination of these three variables. Maximizing the resulting cost function values will therefore optimize the worst case scenario. The final combination of ϕ_b, θ_1 and θ_2 with the highest of these cost function values, will indicate the optimal camera positions for the particular mask.

The basic structure of the *FaceMaskAlgorithm()* and the *BodyMaskAlgorithm()* is shown in Algorithm 4.5.1. In each of these algorithms, the variables θ_x, θ_y and θ_z are substituted with the appropriate mask orientation variables. A standard collision detection algorithm detects collisions between the mask and the beam nozzles. If a collision occurs, the snout of the relevant beam nozzle is contracted. Conversely, the relevant beam nozzle's snout is extended to ensure a maximum distance of 100 mm between the mask and the beam nozzles. *F()* represents the function call to the *CostFunctionAlgorithm()*. The *orientation[]* variable holds the cost function value for every mask orientation and the *heightmap[]* variable holds the worst mask orientation associated with each camera position.

Algorithm 4.5.1

```
for all combinations of $\phi_b, \theta_1$ and $\theta_2$ do
    for all combinations of $\theta_y, \theta_x$ and $\theta_z$ do
        Test for collision between the mask and the beam nozzles.
        if a collision occured
          then Set length of relevant beam nozzle snout.
        endif
        for all camera combinations do
            cost [camera combination] = F (camera combination)
        endfor
        orientation $[\theta_y, \theta_x, \theta_z]$= max{cost [all camera combinations]}
    endfor
    heightmap $[\phi_b, \theta_1, \theta_2] = \min\{\text{orientation}[\theta_y, \theta_x, \theta_z]\}$
endfor
```

4.5.2 The combination algorithm

The *CombinationAlgorithm()* determines the optimal camera positions for treatments using both the face mask and the body mask. This algorithm achieves this by comparing the cost function values from the different face mask orientations with the cost function values from the different body mask orientations at every camera position. In the first step of this algorithm the maximum cost function value from the two optimal camera positions, one associated with each mask, is selected for the current value of ϕ_b. The second step determines the difference between the values of *heightmap[]*, from each mask, and the maximum cost function value at each different configuration of the camera rigs. The cost function associated with a maximum difference is assigned to the current configuration of the camera rigs. These two steps are repeated for every value of ϕ_b. As a result of repeating these two steps, the minimum of the cost function values associated with each mask is selected at every camera position. The optimal camera positions for treatments using both mask correspond to the combination of ϕ_b, θ_1 and θ_2 with the maximum value associated with it.

The basic structure of the *CombinationAlgorithm()* is shown in Algorithm 4.5.2. Variable *F_max[]*, on line 4, holds the maximum cost function value of the optimal camera positions for each ϕ_b. The resulting cost function values associated with each camera position and each ϕ_b value are kept in *heightmap[]*. The optimal camera positions are associated with the minimum value of *optimal_value[]*.

Algorithm 4.5.2

```
1  for every value of $\phi_b$ do
2      max_face = max{heightmap from the face mask}
3      max_body = max{heightmap from the body mask}
4      F_max[$\phi_b$] = max{max_face, max_body}
5      for all combinations of $\theta_1$ and $\theta_2$ do
6          diff_face = F_max − heightmap from face mask algorithm
7          diff_body = F_max − heightmap from body mask algorithm
8          heightmap [$\phi_b, \theta_1, \theta_2$] = max{ diff_face, diff_body }
9      endfor
10     optimal_values[$\phi_b$] = min{heightmap [$\phi_b, \theta_1, \theta_2$]}
```

```
11 endfor
```

4.5.3 The cost function algorithm

The *CostFunctionAlgorithm()* takes as input a particular camera combination and returns the associated cost function value. Since algorithms that evaluate the cost function for the camera triplets and the camera quadruplet are similar only the algorithm associated with the camera triplet will be discussed in this section.
The angles (ψ) between the optical axes of the different cameras in a specific combination are calculated in the first step of this algorithm. These calculations are based on Equation 4.12. The second step uses ψ and calculates the value of the weight function for each camera pair. The third step calculates the area of the mask visible by each camera combination. This step involves function calls to *CalcVisibleTriangles()* and *CalcCommonTriangles()*. The last two steps provide both the weight function values and the areas associated with each camera combination which are needed to evaluate the cost function (Equation 4.3). The basic structure of the *CostFunctionAlgorithm()* is shown in Algorithm 4.5.3.

Algorithm 4.5.3

```
1 Calculate separation angles between the cameras.
2 Calculate weight function values for each camera pair.
3 Calculate area of mask visible from each camera pair and the camera triplet.
4 Evaluate the cost function from Equation 4.3.
```

Calculating the visible areas of a mask

Two algorithms are responsible for calculating the area of the mask that is visible to each camera combination. These algorithms are:

- *CalcVisibleTriangles()*

- *CalcCommonTriangles()*

The *CalcVisibleTriangles()* algorithm is responsible for calculating the area of the mask that is visible to an individual camera. Algorithm *CalcCommonTriangles()* calculates the area of the mask that is visible to a camera pair, camera triplets and the camera quadruplet from the results of the *CalcVisibleTriangles()* algorithm. These two algorithms are described next. The principles involved in calculating the area of the mask that is visible to a camera triplet and the camera quadruplet follow naturally from the same calculations for a camera pair and will therefore be omitted. The *CalcVisibleTriangles()* algorithm is described first.

Algorithm 4.5.4

```
1 for each triangle in the model of the mask do
2     if triangle is not obscured by the beam nozzles
3        then if triangle projects onto the CCD of the camera
4                 then triangle is visible
5             endif
6     endif
7 endfor
8 return all the visible triangles.
```

The algorithm traverse through each triangle in the model (or mask) and determine if that triangle is visible to the particular camera. Triangles are only visible to a camera if the camera view of the triangle is not obscured by any beam nozzle and the triangle projects onto the CCD of that camera. The first of these tests involves testing for a collision between the beam nozzles and a ray (line) whose end points are defined by a vertex point of the triangle and the pinhole of the camera. The second test involves projecting each triangle onto the CCD of the camera and determining which triangles project to within the boundaries of the camera's CCD. This test also makes provision for triangles that project to the same area on the CCD of the camera with the use of a *PointInTriangle()* (2.3.1) algorithm. If two or more triangles project to the same area, only the triangle with original position closest to the camera is considered. The *CalcVisibleTriangles()* algorithm terminates by returning all the triangles visible by the camera.

Algorithm 4.5.5

```
1 Calculate triangles visible to each camera in a camera pair.
2 Flag triangles which are visible to both cameras.
3 return the sum of the areas from all the flaged triangles.
```

The first step in the *CalcCommonTriangles()* algorithm calculates the triangles visible to each individual camera by using the *CalcVisibleTriangles()* algorithm. The triangles that are visible to both cameras are flagged. This algorithm terminates by returning the sum of the areas from all the flagged triangles.

4.6 Software testing

A number of tests were conducted to test the accuracy of the simulation routines. Most of these tests involved analyzing the results from the simulation when using a sphere in the place of a mask. Under such a condition the results from the simulation should reflect the symmetrical properties associated with the sphere. For example, the area of the sphere that is visible from camera pairs with equal ϕ_c angles should be the same. We used a sphere because it is analytically tractable.

The first set of tests compared the area of the sphere visible from camera pairs and camera triplets that have similar ϕ_c angles. These tests also include a comparison between the cost function values associated such camera triplets. The results from these tests are described in Section 4.6.1.

The second test compares the areas of the sphere that are calculated by the simulation routines with a manual calculation of the area. The resulting areas for a camera pair are compared with different ϕ_c angles between the cameras. These results are covered in Section 4.6.2.

The third test determined the areas of the sphere visible from the camera pair that is positioned closest to the beam nozzles. These areas are calculated for different ϕ_b angles which shows the change in the areas visible from each individual camera with the change in area visible from both cameras simultaneously. These results are discussed in Section 4.6.3.

The last test determined what the optimal angle between the cameras in a camera pair is for simulations using a sphere. This optimal angle was obtained by evaluating the cost function for a single camera pair at different ϕ_b angles. These results are given in Section 4.6.4.

4.6.1 Symmetrical tests

The camera setup configurations shown in Figure 4.10 have a number of similar camera combinations. Similar camera pairs consist of two cameras separated by equal ϕ_c angles and similar camera triplets consist of three cameras with both ϕ_c angles equal. The visible areas of a sphere calculated by the simulation routines for these similar camera combinations should be equal when ignoring view occlusion. The divergence of the areas visible to each similar camera combination from their mean is indicated using a percentage residual error $(\delta^{(\%)})$ and the standard deviation of this residual error $(\sigma^{(\%)})$. These errors are defined as:

$$\begin{aligned}\delta^{(\%)} &= \frac{A_i - \overline{A}}{\overline{A}} \times 100 \\ \sigma^{(\%)} &= \sqrt{\frac{\sum_{i=1}^{N} \delta^{(\%)2}}{N-1}} \qquad (4.15)\end{aligned}$$

were A_i represents either the area visible from a camera combination or the cost function value associated with a camera triplet. $\overline{A}$ represents the average of all A_i's and N is the number of camera combinations.

The first test compared the area of the sphere that is visible from individual cameras. Each of these cameras has an unoccluded view and should therefore view equal sized areas of the sphere. The areas values obtained from the simulation calculations are shown in Figure 4.16. The graph in this figure shows the residual error $(\delta^{(\%)})$ for each camera which results in a $\sigma^{(\%)} = 0.027\%$.

The second test compared the area of the sphere that is visible from similar camera pairs. The results of the simulation calculations for camera pairs with $\phi_c = 40°$ and $\phi_c = 80°$ are shown in the top and bottom graphs of Figure 4.17. Values of $\sigma^{(\%)} = 0.013\%$ and $\sigma^{(\%)} = 0.020\%$ were obtained from the respective results. The results for camera pairs with $\phi_c = 120°$ and $\phi_c = 160°$ are shown in the top and bottom graphs of Figure 4.18. For this case, we obtained values of $\sigma^{(\%)} = 0.066\%$ and $\sigma^{(\%)} = 0.078\%$ respectively.

The third test compared the cost function values calculated for a set of similar camera triplets. The results from the simulation calculations are shown in Figure 4.19. The residual errors in the areas and the cost functions are shown in the top and bottom graphs respectively. We observed $\sigma^{(\%)} = 0.019\%$ and $\sigma^{(\%)} = 0.026\%$ values for the calculated areas and cost functions.

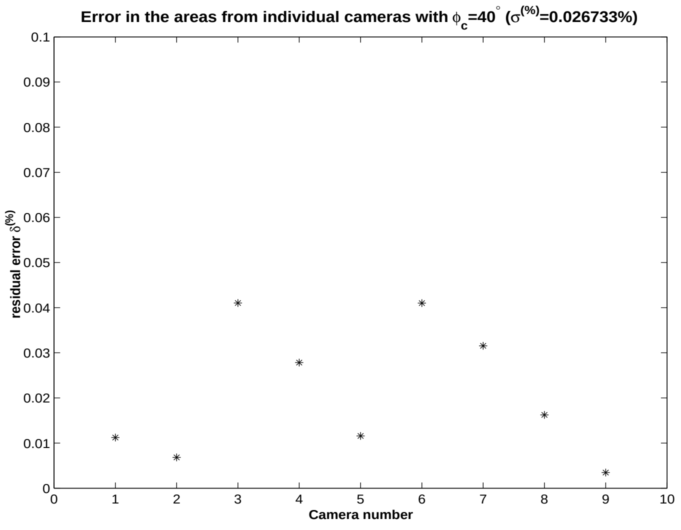


Figure 4.16: The area of a sphere visible from individual cameras.

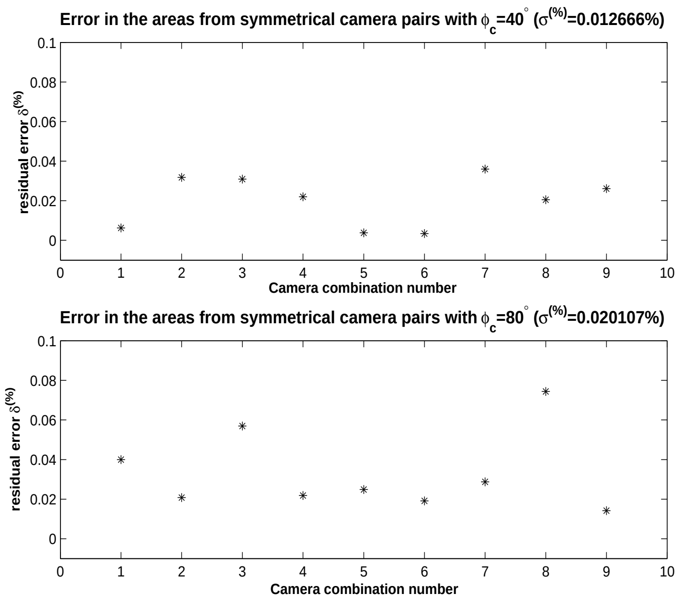


Figure 4.17: The area of a sphere that is visible from similar camera pairs with camera separation angles of 40° (top) and 80° (bottom).

All the tests resulted in $\sigma < 0.1\%$. The source of this error is a direct result from using different size triangles to approximate the treatment masks. In such an approximation a triangle is either completely visible or not visible at all. The

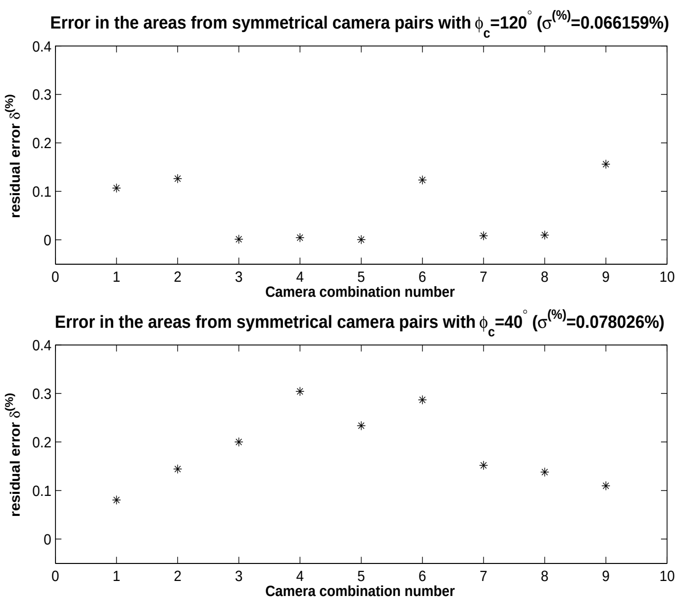


Figure 4.18: The area of a sphere visible from similar camera pairs with camera separation angle of 120° (top) and 160° (bottom).

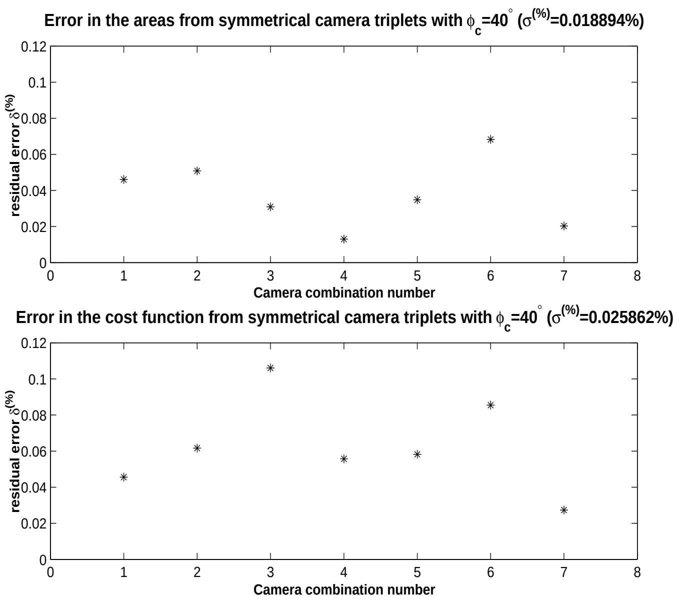


Figure 4.19: The area of a sphere that is visible from similar camera triplets and the cost function values for them. Both camera separation angles are 40°.

value of σ indicate the effect of ignoring the effect of partially occluded triangles. Such a small error in the divergence of values obtained for similar camera combinations is a good indication that the simulation routines involved in both the area calculations and the cost function calculations are correct.

4.6.2 Linearity test

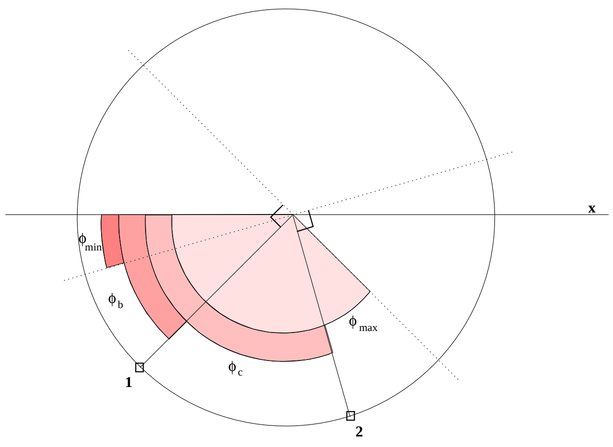


Figure 4.20: The integration area of sphere specified by ϕ_{min} and ϕ_{max}

The area of the sphere that is visible from a single camera pair was calculated by integrating over the relevant area. The relevant area is bounded by the polar angles ϕ_{min} and ϕ_{max}, as indicated in Figure 4.20, and the azimuth angles $\theta = 0°$ and $\theta = \frac{pi}{2}$. In this figure, the two dotted lines indicate which part of the sphere is visible to cameras 1 and 2 respectively. The area of the sphere that is visible to both cameras simultaneously depends on the value of ϕ_c. This last relation allow us to express ϕ_{min} and ϕ_{max} as a function of ϕ_c as follow:

$$\begin{aligned}\phi_{min} &= \phi_b + \phi_c - \frac{\pi}{2} \\ \phi_{max} &= \phi_b + \frac{\pi}{2} \\ \phi_{max} - \phi_{min} &= \pi - \phi_c \qquad (4.16)\end{aligned}$$

The area of the sphere visible from a camera pair was calculated by integrating over the area of the sphere defined by the boundaries in Equation 4.16. This integral was evaluated as follows:

$$\begin{aligned} dA &= r^2 sin(\theta) d\theta d\phi \\ A_{ij}(\phi_c) &= \int\int dA \\ &= r^2 \int_{\phi_{min}}^{\phi_{max}} d\phi \int_0^{\pi} \sin(\theta) d\theta \\ &= 2(\phi_{max} - \phi_{min}) \\ &= 2\pi - 2\phi_c \qquad (4.17) \end{aligned}$$

where $r = 1$ is the radius of the unit sphere used in the simulations and i and j are any two cameras separated by an angle ϕ_c. Equation 4.17 expresses the area of the sphere visible from a camera pair as a linear decreasing function of ϕ_c with an offset value of 2π and a gradient of -2. The properties of this linear function were compared to results from the simulation routines which are shown in Figure 4.21. The offset and gradient values from the simulations are 6.3 and -2.0 respectively. The good correspondence between the respective offset and gradient values is a good indication that the area calculations at different ϕ_c angles are correct.

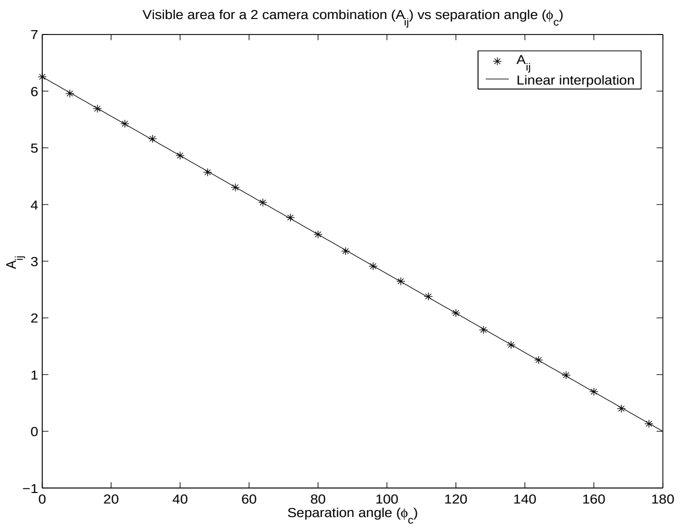


Figure 4.21: The area of a sphere visible by a camera pair at different ϕ_c angles.

4.6.3 Beam occlusion test

The view from the cameras that are positioned closest to the beam nozzles is occluded by these nozzles. This influence of the beam nozzles on the area of the sphere that is visible from a camera pair is shown in Figure 4.22. The graph in this figure shows that for $\phi_b < 20°$ the view of one camera is partially occluded by the beam nozzles whereas the other camera's view is completely occluded for $\phi_b < 25°$. The last of these cameras should therefore be the camera positioned closer to the beam nozzles. Also, the area of the sphere visible to both cameras simultaneously is always less than or equal to the area of the camera that views the smaller area. At $\phi_b > 60°$ neither the cameras' views are occluded by the beam nozzles and are therefore equal and constant. The results from the simulations for cameras positioned close to the beam nozzles are as expected. This result is also a good indication that the simulation calculations are correct for situations where the beam nozzles effect the view of the cameras.

The results from this test were also tested against the results from the second

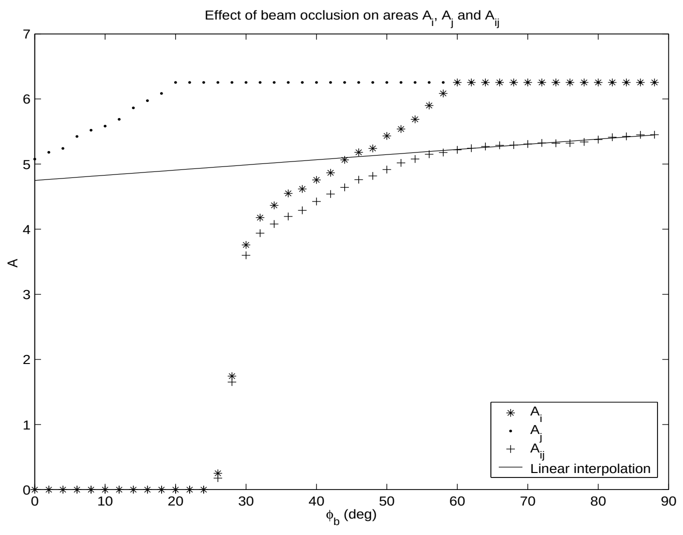


Figure 4.22: The effect of beam occlusion to the view of cameras i and j.

test. This test involved testing the area calculations of the simulation routines as a function of ϕ_b. A linear interpolation through the area values obtained from the third test at $\phi_b > 60°$ produced a linear function with offset and gradient values of 4.7 and 0.5 respectively. These values were compared against the offset and gradient values obtained from substituting Equation 4.13 in Equation 4.17. The resulting function is,

$$A_{ij}(\phi_b) = \frac{1}{2}\phi_b + \frac{3\pi}{2} \tag{4.18}$$

which shows the offset and gradient values as $\frac{3\pi}{2}$ and $\frac{1}{2}$ respectively. The good correspondence between these results is another good indication that the area calculations of the simulations are correct for different ϕ_b values also.

4.6.4 Optimal camera separation angle

The sphere is also used to determine the optimal camera separation angle between a camera pair. This optimal angle resulted from the cost function which includes both area calculations and weight function calculations (see Section 4.1.1). The weight function is at a maximum when $\phi_c = 90°$ whereas the area between a camera pair is at a maximum when $\phi_c = 0°$. The expected result from combining these two functions will be an optimal angle somewhere between these two extreme angles. This optimal angle is also expected to be slightly larger than 45° because the weight function is a cubic function, whereas the function that determines the area is a linear function (see Equation 4.17). The result from this test is shown in Figure 4.23. An optimal angle of 56° is a good indication that the calculations are correct.

The graph in Figure 4.23 also shows that the cost function is more sensitive (changes more rapidly) for angles smaller than 56°. For example, the change in the cost function values between $\phi_c = 20°$ and $\phi_c = 40°$ is much larger than the change in cost function values between $\phi_c = 80°$ and $\phi_c = 100°$. This result implies that angles larger than 56° are preferred to angles that are smaller than the optimal angle.

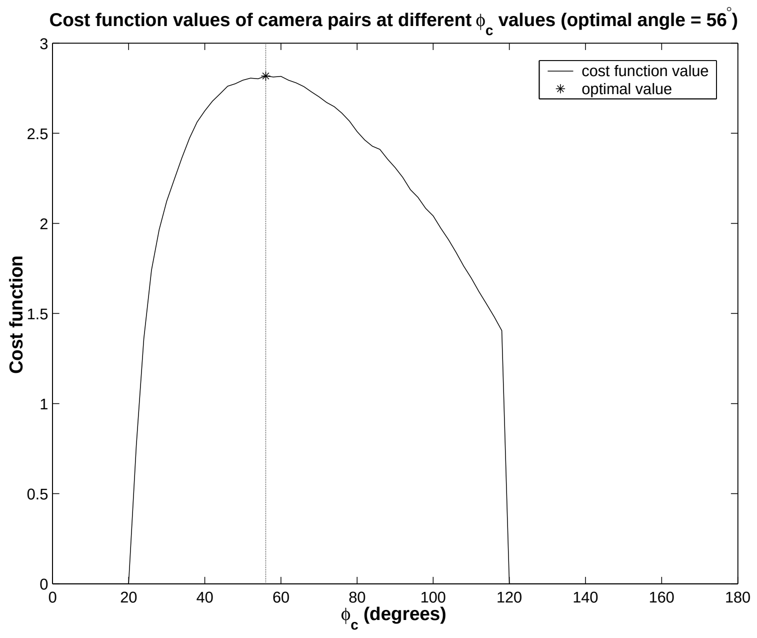


Figure 4.23: The optimal angle (ϕ_c) between a camera pair.

4.7 Simulation results

A number of different constraints on the placement of the cameras in the treatment vault were considered. These constraints were incorporated in different simulations to determine the optimal placement of the cameras subject to these constraints. Together, these simulations were used to perform an exhaustive search of the search space that defined the allowed camera positions in the treatment vault. The different constraints involved the following: the number of treatment beams considered (horizontal and/or skew treatment beam); the number of cameras considered (eight or nine); and whether cameras were kept at fixed distances from the isocenter or not. These different setup configurations are described next.

Simulation 1

Simulation 1 considered both treatment beams and a total of nine cameras mounted on two separate camera rigs. During this simulation, all nine cameras were kept at the same distance (1600 mm) from the isocenter. *Simulation 1a* and *Simulation 1b* represented two different variations of this simulation. *Simulation 1a* considered two independent camera rigs at variable heights and resulted in *Solution 1a*, the optimal solution. *Simulation 1b* required both camera rigs to be kept at the same height and resulted in *Solution 1b.* This solution represents the best solution from a manufacturing point of view. The setup in *Simulation 1b* is equivalent to having all nine cameras mounted on a single rig, with the height of the rig allowed to vary in the simulation.

Simulation 2

Simulation 2 considered both treatment beams and a total of nine cameras mounted on two separate camera rigs. This simulation required the radii of both camera rigs to be equal while the distance between each camera and the isocenter was allowed to vary between 1600 mm and 1950 mm. *Solution 2a* represents the optimal solution of this simulation.

Simulation 3

Simulation 3 considered only the horizontal treatment beam with the nine cameras mounted on two separate camera rigs. Again, all the cameras are kept at equal distances (1600 mm) from the isocenter. *Simulation 3a* and *Simulation 3b* represented the two different variations of this simulation. *Simulation 3a* considered two independent camera rigs at variable heights and resulted in *Solution 3a*, the optimal solution. *Simulation 3b* required both camera rigs to be kept at the same height and resulted in *Solution 3b*. This solution represents the best solution from a manufacturing point of view.

Simulation 4

Simulation 4 considered only the horizontal treatment beam with nine cameras mounted on two separate camera rigs. This simulation also required the radii of both camera rigs to be equal while the distance between each camera and the isocenter was allowed to vary between 1600 mm and 1950 mm. *Solution 4a* represented the optimal solution of this simulation.

Simulation 5

Simulation 5 considered only the horizontal treatment beam with eight cameras mounted on the two separate camera rigs. In this simulation, all eight cameras were kept at equal distances (1600 mm) from the isocenter. This simulation considered only the optimal solution, *Solution 5*.

The results from these simulation are presented in the *./SimResults* directory on the accompanied compact disc. These results are presented as output tables and 2D surface plots (height map). An output table shows the cost function values, the percentage area values or the minimum camera separation angles associated with every combination of variables θ_1 and θ_2. The height maps provide a graphical interpretation of the values in the output table. Each value of ϕ_b is associated with an output table and a height map. The highest cost function value indicates the values of ϕ_b, θ_1 and θ_2 that result in the optimal position of the cameras. The results from each of the above mentioned simulations are provided next.

4.7.1 Simulation 1

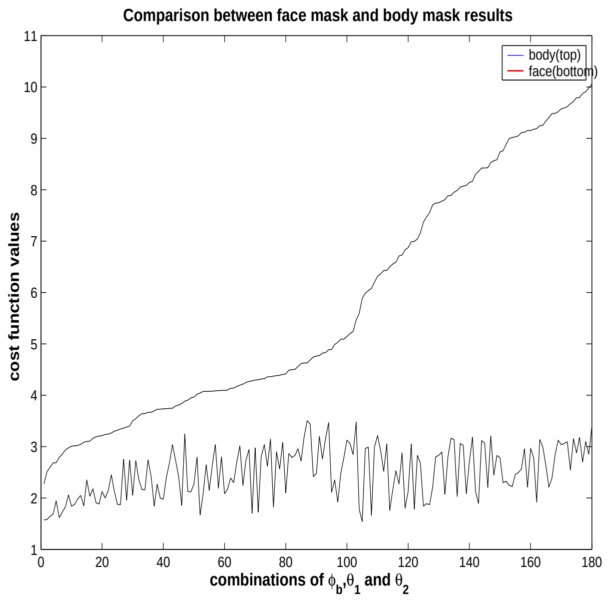


Figure 4.24: A comparison between the cost function values obtained from the face mask and the body mask at different ϕ_b angles.

This simulation used both the face mask and the body mask to determine the optimal position of the nine cameras subjected to the constraints specified by *Simulation 1* in the previous section. Each of these masks had a cost function value associated with every combination of the variables ϕ_b, θ_1 and θ_2. The individual results for both masks are shown in Figure 4.24. The line at the top of this graph is the sorted cost function values for the body mask. The bottom line indicates the corresponding cost function values for the face mask. The *CombinationAlgorithm()* from Section 4.5.2 combined these resulting cost function values by selecting the minimum cost function value for each combination of the variables ϕ_b, θ_1 and θ_2. It is evident from the graph in Figure 4.24 that the cost function values for the body mask are always greater than the corresponding cost function values for the face mask. Combining these results will therefore always produce the same results as the simulation with only the face mask. This result shows that the smaller and more detailed face mask is harder to observe and results in a worse worst case performance than the body mask. Since this will always be the case, all the other simulations considered only the face mask to determine the optimal camera positions.

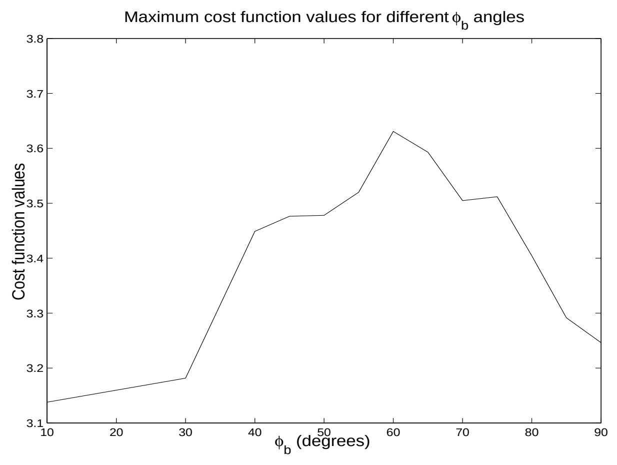


Figure 4.25: The maximum cost function values for different ϕ_b angles in *Simulation 1a* (max at $\phi_b = 60°$).

The simulations were based on maximizing the cost function. The maximum cost function value indicated the values of ϕ_b, θ_1 and θ_2 that resulted in the optimal camera positions. The maximum cost function value for each ϕ_b value in *simulation 1a* is provided in Figure 4.25. The global maximum of this graph indicates a maximum cost function value of 3.6308 at $\phi_b = 60°$. This value of ϕ_b, together with Equation 4.9, specified the optimal position of the cameras on the camera rigs. The maximum cost function value and the optimal value of ϕ_b were used as an index into the associated output table to determine the optimal values of both θ_1 and θ_2. The corresponding optimal values were $\theta_1 = 31.67°$ and $\theta_2 = 70.52°$. These values translate to a camera rig configuration where one camera rig is positioned at the highest allowed height of 2962.78 mm from the floor of the treatment vault, while the other rig is positioned at the lowest allowed height of 2133.6 mm from the floor of the treatment vault. Individually these distances resulted in camera rig radii of 838.2 mm and 1508.4 mm respectively. θ_2 indicates the camera rig for the five cameras which include cameras one and nine. The values associated with the optimal camera positions in *solution 1a* are listed in Table 4.3.

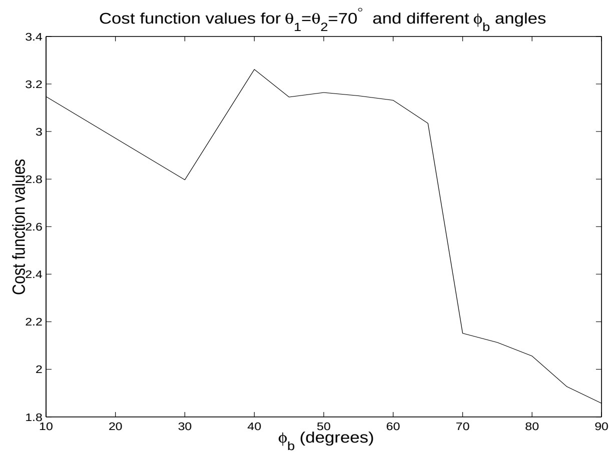


Figure 4.26: The maximum cost function values for different ϕ_b angles in *Simulation 1b* (max at $\phi_b = 40°$).

The values of ϕ_b, θ_1 and θ_2 that resulted in the maximum cost function value indicated the optimal camera positions. The maximum cost function value for each ϕ_b value in *simulation 1b* is shown in Figure 4.26. The global maximum of this graph indicates a maximum cost function value of 3.2611 at $\phi_b = 40°$. The camera rig configuration that corresponds to these two values were described by $\theta_1 = \theta_2 = 70.52°$. These values of θ_1 and θ_2 translate to a distance of 2133.6 mm from the floor of the treatment vault and a rig radius of 1508.4 mm for both camera rigs. A summary of the values associated with the optimal camera positions in *solution 1b* is also provided in Table 4.3.

Each cost function value is associated with a camera combination and consists of both an area component and an angle component. The area component indicates the common area of the mask visible by that camera combination while the angle component gives the minimum camera separation angle of the camera combination. The area component is expressed as both a percentage of the total mask area (152917 mm^2) and the physical area that is measured in cm^2. These values were obtained by substituting $\omega = 1$ in the cost function (Equation 4.3). The angle component is used to calculate the error that is achieved when reconstructing the 3D position of a marker from a camera pair separated by such an angle (see Section 4.1.1). The accuracy in the reconstruction was obtained from substituting the minimum camera separation angle in Equation 4.6. The cost function values, the areas value and the angles associated with *solution 1a* and *solution 1b* are summarized in Table 4.3.

	Solution 1a	**Solution 1b**
ϕ_b	60°	40°
θ_1	31.7°	70.5°
θ_2	70.5°	70.5°
Cost function	3.6308	3.2611
Area(%)	37.4 %	44.4 %
Area9cm^2)	572.5 cm^2	678.9 cm^2
Angle	44.6°	32.9°
Reconstruction error	2.658 mm	3.559 mm

Table 4.3: The camera positions associated with *solution 1a* and *solution 1b*.

4.7.2 Simulation 2

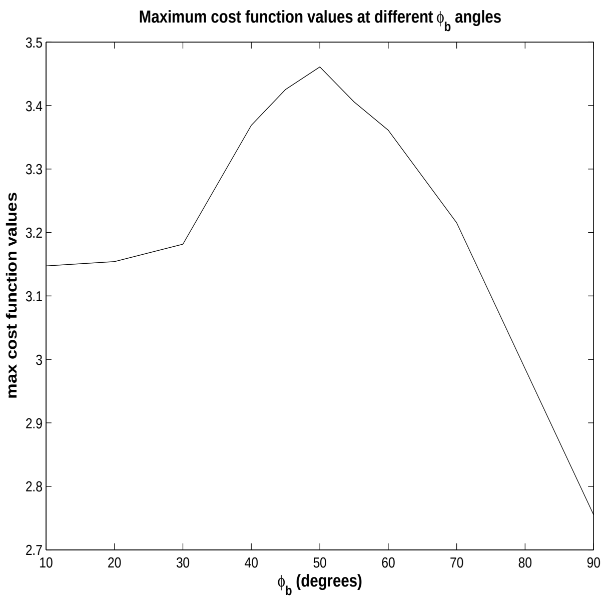


Figure 4.27: The maximum cost function values for different ϕ_b angles in *Simulation 2* (max at $\phi_b = 50°$).

Simulation 2 considered a setup in which the radii of both camera rigs were kept constant. This setup resulted in camera placement configurations where the distances between cameras and the isocenter varied between 1600 mm and 1950 mm. The maximum cost function value for each ϕ_b value in *solution 2a* is provided in Figure 4.27. The global maximum of this graph indicates a maximum cost function value of 3.4609 at $\phi_b = 50°$. The camera rig configuration that corresponds to these two values was described by $\theta_1 = 47.21°$ and $\theta_2 = 70.52°$. These values translate to a camera rig configuration where one camera rig is positioned at 2686.90 mm from the floor of the treatment vault while the other rig is positioned at the lowest allowed height of 2133.6 mm from the floor of the treatment vault. Both these camera rigs have radii of 1508.4 mm. The values associated with the optimal camera positions in *solution 2a* are listed in Table 4.4.

	Solution 2a
ϕ_b	$50°$
θ_1	$47.2°$
θ_2	$70.5°$
Cost function	3.4609
Area(%)	40.03 %
Area($\mathbf{cm^2}$)	612.15 cm^2
Angle	$35.9°$
Reconstruction error	3.2758 mm

Table 4.4: The camera positions associated with *solution 2a*.

4.7.3 Simulation 3

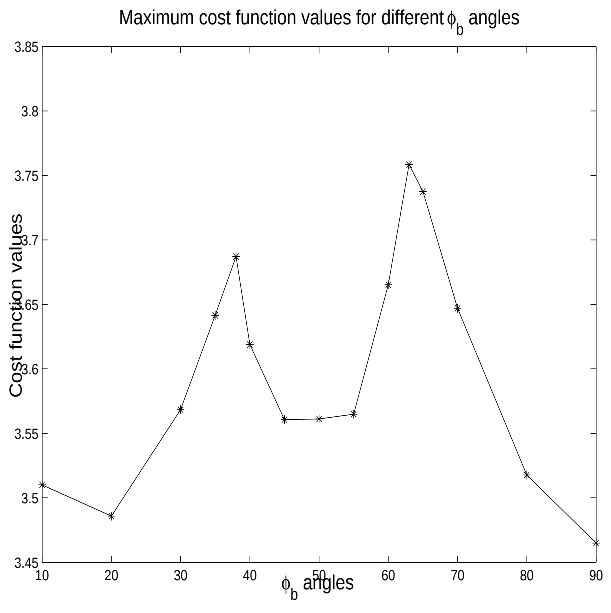


Figure 4.28: The maximum cost function values for different ϕ_b angles in *Solution 3a*.

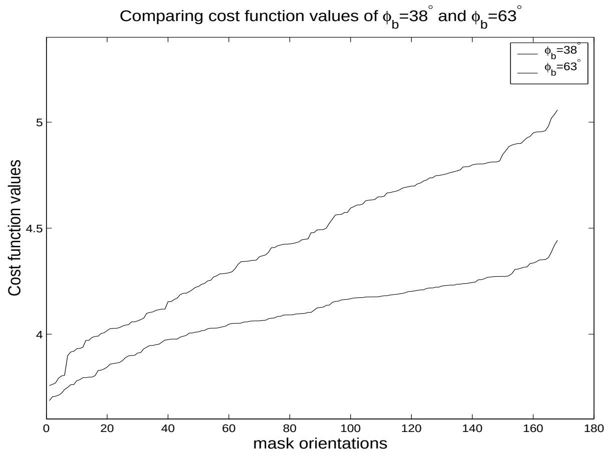


Figure 4.29: A comparison between the cost function values of $\phi_b = 38°$ (bottom) and $\phi_b = 63°$ (top) at every mask orientation.

The constraints in *simulation 3a* was the same as for *simulation 1a*, except that only the horizontal beam delivery system was considered. The maximum cost function value for each ϕ_b value in *solution 3a* is shown in Figure 4.28. However, the resulting graph indicates two local maxima, one at $\phi_b = 63°$ and one at $\phi_b = 38°$. One of these local maxima is also the global maxima. A comparison of the cost function values associated with both these ϕ_b angles as well as for every mask orientation is shown in Figure 4.29. This graph indicates that the cost function values associated with $\phi_b = 63°$ are always greater that the cost function values associated with $\phi_b = 38°$. This proved that the global maximum is indicated by $\phi_b = 63°$. The maximum cost function value was 3.7584 with $\theta_1 = 31.67°$ and $\theta_2 = 70.52°$ indicating the optimal camera rig configuration. These values translate to a camera rig configuration where one camera rig is positioned at the maximum allowed height of 2962.78 mm from the floor of the treatment vault while the other rig is positioned at the lowest allowed height of 2133.6 mm from the floor of the treatment vault. Individually, these distances resulted in camera rig radii of 838.2 mm and 1508.4 mm respectively. A summary of the values associated with the optimal camera positions in *solution 3a* is provided in Table 4.5.

The results from *Simulation 3a* are somewhat surprising, and so a closer exami-

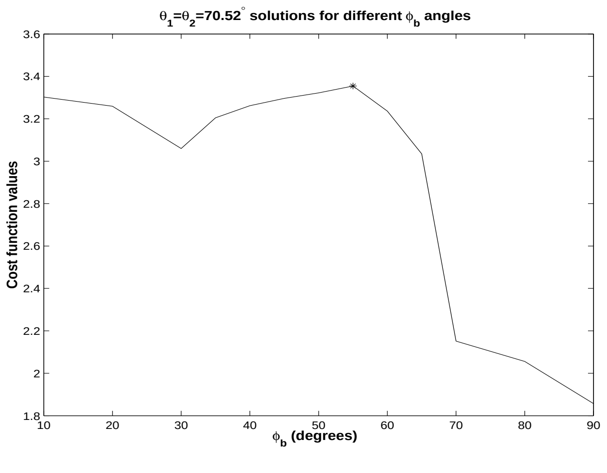


Figure 4.30: The maximum cost function values for different ϕ_b angles in *Solution 3b* (max at $\phi_b = 55°$).

nation is warranted. Intuitively, one would expect that the value of ϕ_b should be smaller for a simulation that considered only the horizontal beam delivery system. A smaller value of ϕ_b would increase the minimum angle between the cameras and therefore also the accuracy of reconstructing the position of the patient's mask. On the other hand, this will decrease the common area of the mask visible to the cameras. Although the angle component is normally the dominant factor in the cost function, its effect reduces with an increase in the minimum camera separation angles. In circumstances like these, it is possible for the area component to become the dominant factor in the cost function calculations. The nonlinearity between changes in these two components of the cost function explains the result from this simulation.

Simulation 3b required both camera rigs to be positioned at the same height. The maximum cost function value for each ϕ_b value in *simulation 3b* is provided in Figure 4.30. The global maximum of this graph indicates a maximum cost function value of 3.3543 at $\phi_b = 55°$. The camera rig configuration that corresponds to these two values was described by $\theta_1 = \theta_2 = 70.52°$. These values of θ_1 and θ_2 translate to a distance of 2133.6 mm from the floor of the treatment vault and a rig radius of 1508.4 mm for both camera rigs. The values associated with the optimal camera positions in *solution 3b* are listed in Table 4.5.

	Solution 3a	**Solution 3b**
ϕ_b	63°	55°
θ_1	31.67°	70.5°
θ_2	70.5°	70.5°
Cost function	3.7584	3.3543
Area(%)	38.53 %	32.06 %
Area($\mathbf{cm^2}$)	589.18 cm^2	490.20 cm^2
Angle	44.3°	29.4°
Reconstruction error	2.67098 mm	3.91244 mm

Table 4.5: The camera positions associated with *solution 3a* and *solution 3b*.

4.7.4 Simulation 4

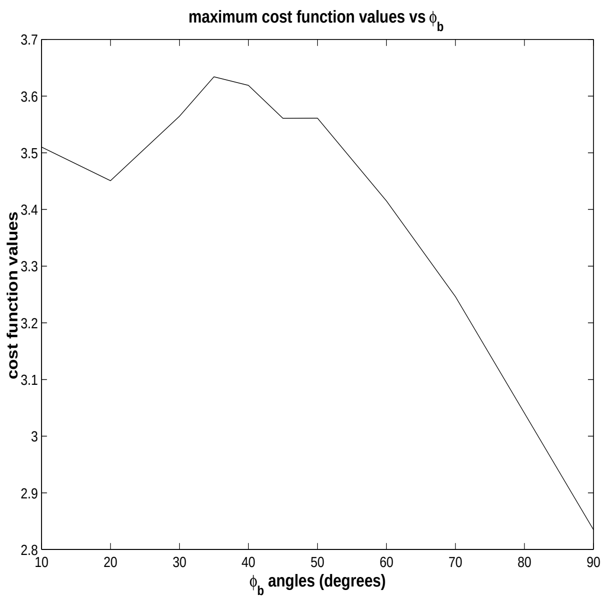


Figure 4.31: The maximum cost function values for different ϕ_b angles in *Simulation 4* (max at $\phi_b = 35°$).

Simulation 4 considered a setup in which the radii of both camera rigs were kept constant. This setup resulted in camera placement configurations where the distances between cameras and the isocenter varied between 1600 mm and 1950 mm. The maximum cost function value for each ϕ_b value in *solution 4a* is provided in Figure 4.31. The global maximum of this graph indicates a maximum cost function value of 3.6443 at $\phi_b = 35°$. The camera rig configuration that corresponds to these two values was described by $\theta_1 = 70.52°$ and $\theta_2 = 51.10°$. These values translate to a camera rig configuration where one camera rig is positioned at 2604.74 mm from the floor of the treatment vault while the other rig is positioned at the lowest allowed height of 2133.6 mm from the floor of the treatment vault. Both these camera rigs have radii of 1508.4 mm. The values associated with the optimal camera positions in *solution 4a* are listed in Table 4.6.

	Solution 4a
ϕ_b	35°
θ_1	70.52°
θ_2	51.10°
Cost function	3.6443
Area(%)	33.73 %
Area($\mathbf{cm^2}$)	515.84 cm^2
Angle	36.8°
Reconstruction error	3.195 mm

Table 4.6: The camera positions associated with *solution 4a*.

4.7.5 Simulation 5

Simulation 5 required only eight cameras to be mounted at fixed positions on the camera rigs. This camera placement configuration was described in Section 4.7. The maximum cost function value of this solution was 3.5152 which was associated with $\phi_b = 45°$, $\theta_1 = 70.5°$ and $\theta_2 = 47,2°$. The values associated with the optimal camera positions in *solution 5* are listed in Table 4.7.

	Solution 5
ϕ_b	45°
θ_1	70.52°
θ_2	47.21°
Cost function	3.5152
Area(%)	31.00 %
Area($\mathbf{cm^2}$)	474.07 cm^2
Angle	36.7°
Reconstruction error	3.201 mm

Table 4.7: The camera positions associated with *solution 5*.

4.7.6 Additional results

Since we are interested in solutions that satisfy both the single and the double beam line configuration, we substituted the solution of *simulation 2* into *simulation 4*'s setup to generate *solution 2b* and vice-versa to generate *solution 4b*. The setup of the cameras in *solution 2a* and *solution 4a* are very similar. In both these solutions one of the camera rigs is positioned at the minimum allowed hight of 70.52° while the other camera rig is positioned at either 47.2° or 51.10°. A relatively small change in the camera positions in each of these solutions will result in an identical camera configuration to the other solution. It is worth knowing the effect on the resulting cost function value as a result of such changes. An identical camera configuration for both solutions is highly favorable from a manufacturing point of view. *Solution 2b* and *solutions 4b* are provided in Table 4.8.

	Solution 2b	**Solution 4b**
ϕ_b	50°	35°
θ_1	70.52°	47.2°
θ_2	51.10°	70.52°
Cost function	3.1711	3.4817
Area(%)	39.64 %	32.21 %
Area($\mathbf{cm^2}$)	606.16 cm^2	492.53 cm^2
Angle	34.09°	38.33°
Reconstruction error	3.447 mm	3.060 mm

Table 4.8: The camera positions associated with *solution 2b* and *solution 4b*.

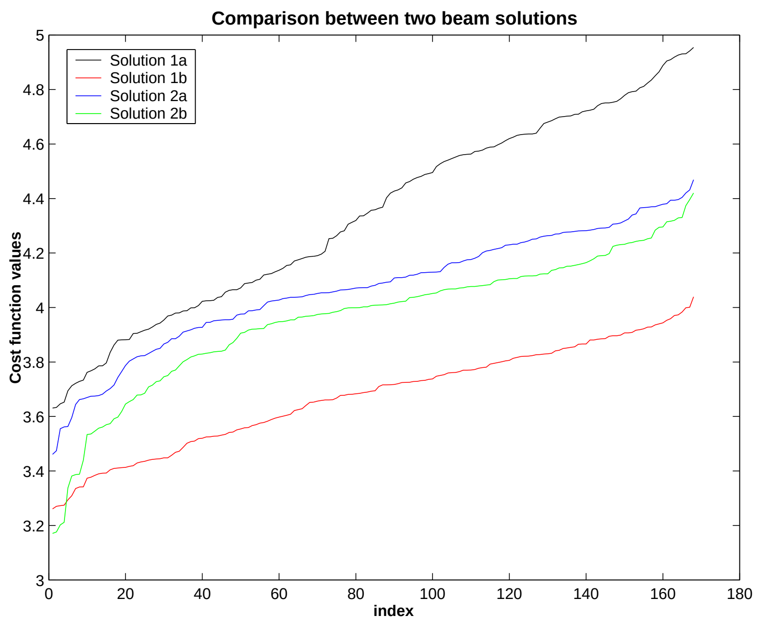
4.8 Summary

The camera positions, cost function value, area value and angle value for the different solutions in the simulation are listed in Table 4.9. Each of these cost function values are associated with a single camera triplet (or quadruplet) and a single mask orientation. The nature of the simulation procedure implies that this camera combination and mask orientation resulted in the lowest cost function value for a particular camera placement configuration (defined by ϕ_b, θ_1 and θ_2). A comparison between only the worst case cost function values might be misleading. Consider two solutions, A and B. Solution A might have higher cost function values associated with the majority of mask orientations than solution B, although the worst case cost function value of solution B is slightly higher than the worst case cost function value of solution A. In this case, solution A might be preferred to solution B, although this is not reflected in Table 4.9.

It is therefore necessary to also compare the different cost function values at all the mask orientations. These comparisons are shown in Figure 4.32. The graph at the top compares the solutions associated with both beam delivery systems while the graph at the bottom compares the solutions associated with only the horizontal beam delivery system. Each graph line is associated with a solution. They represent the cost function values at every mask orientation. The cost function values of each graph is sorted in ascending order to aid comparison of relationships between minimum, average and maximum values. As a result, no information regarding the cost function values at a particular mask orientation can be obtained from these graphs. The minimum cost function value in graph represents the worst case value for each solution. These values correspond to the values listed in Table 4.9. From these graphs we can extract both the average and the maximum cost function value for each solution, which are important indicators of how the solutions compare relative too each other. For example, consider the first graph in Figure 4.32. Here the minimum cost function value of *Solution 1b* is higher than the cost function value of *Solution 2b*. *Solution 1b* is therefore the better solution when comparing only these minimum cost function values like in Table 4.9. Comparing the rest of the cost function values, however, reveals that *Solution 2b* in fact performs better than *Solution 1b* on every mask orientation except four. Despite having a lower minimum cost function value, *Solution 2b* might be preferred to *Solution 1b*.

Two Beams	**Cost funct**	**Area** (%)	**Area** ($\mathbf{cm^2}$)	**Angle**	**error**	ϕ_b	θ_1	θ_2
Sol 1a	3.631	37.44 %	572.51 cm^2	44.56°	2.66 mm	60°	31.7°	70.5°
Sol 1b	3.261	44.39 %	678.92 cm^2	32.94°	3.56 mm	40°	70.5°	70.5°
Sol 2a	3.461	40.03 %	612.15 cm^2	35.90°	3.28 mm	50°	47.2°	70.5°
Sol 2b	3.171	39.64 %	606.16 cm^2	34.09°	3.45 mm	50°	70.5°	51.1°
One Beam								
Sol 3a	3.758	38.53 %	589.18 cm^2	44.30°	2.67 mm	63°	31.7°	70.5°
Sol 3b	3.354	32.06 %	490.20 cm^2	29.42°	3.91 mm	55°	70.5°	70.5°
Sol 4a	3.644	33.73 %	515.84 cm^2	36.77°	3.20 mm	35°	70.5°	51.1°
Sol 4b	3.482	32.21 %	492.53 cm^2	38.33°	3.06 mm	35°	47.2°	70.5°
Sol 5	3.515	31.00 %	474.07 cm^2	36.70°	3.20 mm	45°	70.5°	47.2°

Table 4.9: A summary of the components of the cost functions at the optimal solutions.



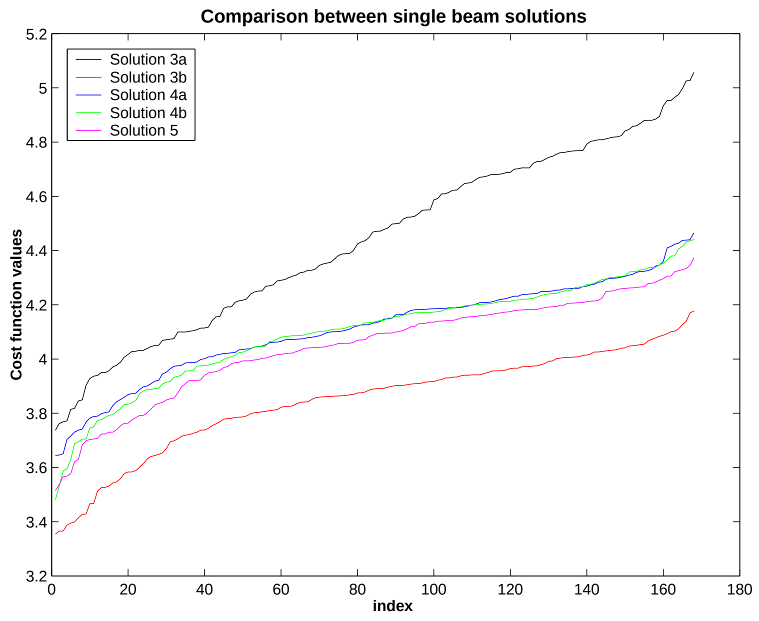


Figure 4.32: Cost function values vs mask orientations for the different solutions of the two (top) and single (bottom) treatment beams.

4.9 Criteria for selecting the best camera configurations

Three factors determined which camera setup should be implemented in the treatment vault. These factors are the optimization results from the simulations, the flexibility of the camera setups regarding the one and two beam line configurations and the complexity of manufacturing a support system for the selected camera setup. The camera setup that provides the best compromise amongst these different factors would be selected for implementation in the treatment vault. The next three subsections give a brief description of each of these factors.

4.9.1 Simulation results

A camera combination is associated with each cost function value. These cost function values provide a measure of both the visible area of a mask and the smallest camera separation angle associated with the camera combination. The smallest camera separation angle gives an indication of how accurately the position of a marker on the mask can be determined from two cameras separated by this camera separation angle. The highest cost function value from the solutions in Table 4.9 will therefore indicate the best camera position according to the simulations. The optimization results in Table 4.9 played an important role in selecting the final camera configurations.

4.9.2 Flexibility of camera configuration

In principle, separate optimal camera configurations can exist for both the one and the two beam line setups. The degree of similarity between these camera configuration also played an important role in selecting the final camera configuration. A single camera configuration that can be used for both the one and the two beam lines setup is highly favoured because it eliminates the need to construct a separate support system for each beam line configuration. Such a configuration is especially valuable in this project as initially only the horizontal beam delivery system will be implemented in the treatment vault and that the implementation of the second beam delivery system will occur only after the single beam-line system has been commissioned.

4.9.3 Manufacturing constraints

A structure is required to support the camera rigs from the roof of the treatment vault. The complexity of this structure, as well as its ability to cater for additional systems also influences the decision regarding the final camera position. Systems like the portal x-ray image acquisition system will be lowered down from the roof of the treatment vault to the height of the isocenter. Both the camera rigs and their support structure should allow for these movements of the portal x-ray image acquisition system. Although the simplest design for a support system is obviously preferred from a manufacturing point of view, more complex designs were also considered. We refer the reader back to Figure 4.8, which shows the relation between the position of the horizontal beam line, the isocenter and the portal x-ray image acquisition system.

4.10 Conclusion

A cost function was used to evaluate the merits of the different camera positions in each simulation. Each cost function value was associated with a particular camera combination and can be subdivided into an area component and an angle component. The area component indicated the size of the common area on the mask that was visible to that particular camera combination. The angle component indicated the minimum camera separation angle between the cameras in the camera combination. The accuracy that is achieved when reconstructing the position of the patient's mask from the camera combination is derived from the minimum camera separation angle.

Solution 1a and *Solution 3a* resulted in the highest cost function values for both the one and the two beam line setups, respectively. The camera positions associated with these solutions are: $\phi_b = 60°$, $\theta_1 = 31.7°$ and $\theta_2 = 70.5°$ for *Solution 1a* and $\phi_b = 63°$, $\theta_1 = 31.7°$ and $\theta_2 = 70.5°$ for *Solution 3a.* The values of θ_1 and θ_2 in these solutions indicate identical camera rig setups. In these setups, the one rig is positioned a the highest allowed height ($\theta_1 = 31.7°$ translates to 1362.78 mm above the isocenter) and the other one at the lowest allowed height ($\theta_1 = 70.5°$ translates to 533.6 mm above the isocenter). The small difference between the values of ϕ_b in each of these solutions also indicates a close correspondence between the camera positions on each camera rig. The values of ϕ_b, θ_1 and θ_2 are not the easiest to contend with from a manufacturing point of view, because they indicate a camera placement configuration where the two camera rigs are positions at the extreme high and extreme low positions. However, it is possible to manufacture a suitable support system for this setup (see Figure 4.33 [1]). The advantages that were gained from a single solution that is optimal for both beam line configurations overrules the complexity associated with manufacturing the support system. The combination of *solution 1a* and *solution 3a* were identified as the optimal solutions for the single and the double beam line configuration respectively. Only one solution can be implemented in the treatment vault, therefore, the camera positions associated with *solution 1a* were selected for implementation. The optimal camera positions are specified by $\phi_b = 60°$, $\theta_1 = 31.7°$ and $\theta_2 = 70.5°$ and will be implemented in the treatment vault.

[1]Conceptual design of the support system for the cameras was done by Evan de Kock.

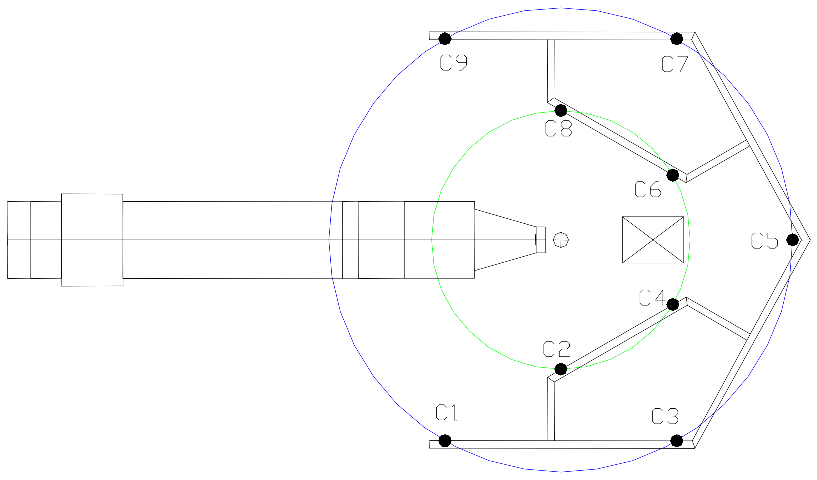


Figure 4.33: A top view of the proposed support system for the optimal camera positions in the treatment vault.

Chapter 5

Conclusion

The aim of this thesis was to aid in the development of the CT scanner SPG system and the Treatment SPG system currently being developed for the new patient positioning system at iThemba Labs. Two simulations were developed for this purpose. The first simulated the setup in the CT scanner room while the second simulated the setup in the treatment vault. These simulations allowed us to test different aspects of the project. This resulted in a better understanding of the setup in both SPG systems.
The simulation routine responsible for determining the guidelines for the marker positions on the patient's mask was described in Chapter 3. The aim was to clarify the effect of the fixed camera positions on the visibility of of the patient's mask during a CT scan. The visible areas determine where to place the markers on the patient's mask. The simulation routine produced a set of images of different marker positions on a oval model. These images were fed to a preliminary marker identification algorithm which resulted in either a *hit* or a *miss* depending on whether the markers were correctly identified or not. The results from these simulations indicated marker positions that satisfy the minimum requirement of the CT scanner SPG system. Furthermore, from the simulation results, we can conclude that enough of the mask is visible throughout the scan to obtain usable data from the

CT scanner SPG system.

The simulation routine responsible for determining the optimal position of the cameras in the treatment vault was described in Chapter 4. These are camera positions that will maximize the visibility of the patient's mask and minimize the error in calculating the position of the mask. Finding the optimal camera positions was an optimization problem in which a cost function was optimized over a high dimensional search space. The simulations resulted in several possible camera positions which were compared based on their respective cost function values, their flexibility toward future developments, such as the addition of a second beam delivery system, and the manufacturing constraints involved in building a support system for the cameras. An exhaustive search in the regions of these optimal solutions gave information about the sensitivity of the camera positions to small deviations from the optimal position. From the results of the treatment vault simulations, a camera placement configuration was chosen and will be implemented in the treatment vault.

From these results, we conclude that the goals stated in the introduction were achieved.

Appendix A

Matrix notation in Computer Graphics

$\mathbf{L}$	Local coordinate system
$\mathbf{W}$	World coordinate system
$\mathbf{V}$	View coordinate system
$\mathbf{D}$	Screen coordinate system
$\mathbf{G}$	Light coordinate system
$\mathbf{K}$	Texture coordinate system
$\mathbf{T}$	Translation matrix
$\mathbf{R}$	Rotation matrix
$\mathbf{S}$	Scaling matrix
$\mathbf{M_m}$	Modeling transformation matrix
$\mathbf{M_v}$	Viewing transformation matrix
$\mathbf{M_{mv}}$	Modelview transformation matrix

$\mathbf{M_p}$	Perspective projection matrix
$\mathbf{M_o}$	Ortographic projection matrix
$\mathbf{M_{vp}}$	Viewport tranformation matrix
$\mathbf{M_{tg}}$	Texgen matrix

Appendix B

Variables used in camera placement simulations

θ	The azimuth angle.
ϕ	The polar angle.
$\theta_{\mathbf{1}}$	The azimuth angle specifying the height of the first camera rig.
$\theta_{\mathbf{2}}$	The azimuth angle specifying the height of the second camera rig.
$\phi_{\mathbf{b}}$	The polar angle specifying the separation angle between the first and last cameras (nine camera setup).
$\phi_{\mathbf{c}}$	The polar angle specifying the separation angle between two cameras in a camera pair.
ψ	The angle specifying the separation angle between the optical axes of the cameras in a camera pair.

$\theta_{\mathbf{z}}^{\mathbf{F}}$	The rotation angle of the face mask around the z-axis.
$\theta_{\mathbf{x}}^{\mathbf{F}}$	The rotation angle of the face mask around the x-axis.
$\theta_{\mathbf{y}}^{\mathbf{F}}$	The rotation angle of the face mask around the y-axis.
$\theta_{\mathbf{z}}^{\mathbf{B}}$	The rotation angle of the body mask around the z-axis.
$\theta_{\mathbf{x}}^{\mathbf{B}}$	The rotation angle of the body mask around the x-axis.
$\theta_{\mathbf{y}}^{\mathbf{B}}$	The rotation angle of the body mask around the y-axis.

Bibliography

[1] P. Allard, I.A.F. Stokes & J.B. Blanchi, *Three-Dimensional Analysis of Human Movement*, Human Kinetics, July 1991.

[2] K.F. Bennett, H. Rüther, G. van der Vlugt & A.D.B. Yates, *Patient positioning for proton therapy at the National Accelerator Center.*, South African Journal of Science, Vol. 90, July 1994.

[3] D. Blythe, *Projective Textures*, **http://www.opengl.org/developers/code/sig99/advanged99/notes/**.

[4] N.A. Borghese & G. Ferrigno, *An algorithm for 3D automatic movement detection by means of standard TVcameras*, Proceedings of the IEEE International Conference on Computer Vision and Biomedical Engineering, Trans. Biomed. Eng., 1990, Vol. 32, pp. 1221-1225.

[5] R.L. Burden & J.D. Faires, *Numerical Analysis*, Sixth Edition, Brooks/Cole Publishing Company, 1977.

[6] C. Everitt, *Projective texture mapping*, **http://www.developer.nvidia.com/object/**.

[7] E. A. de Kock, *The conceptual design of a robot-based patient positioning system*, Medical Radiation Group, iThemba LABS, 8 March 2002.

[8] E.A. de Kock, *Logical program flow for the Treatment SPG System*, Medical Radiation Group, iThemba LABS, 1 April 2003.

[9] E.A. de Kock, *Ray-tracing calculations using CT data and prism representations for arbitrary 3-dimensional volumes*, Medical Radiation Group, iThemba LABS, 4 July 1997.

[10] U.R. Dhond & J.K. Aggarwal, *Structure from Stereo-A Review*, IEEE Transaction on systems, man, and cybernetics, 6 November 1989.

[11] N. Dodgsom, *Ray tracing primitives*, **http://www.cl.cam.ac.uk/Teaching/2000/AGraphHCI/SMEG/**.

[12] R. Hartley & A. Zisserman, *Multiple View Geometry in computer vision*, Cambridge University Press 2000.

[13] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald & W. Stuetzle, *Surface reconstruction from unorganized points*, Proceedings of SIGGRAPH '92, (Chicago, Illinois, July 26–31, 1992). In *Computer Graphics* 26(2) (July 1992), 71–78.

[14] H. Hoppe, T. Derose, T. Duchamp, J. McDonald & W. Stuetzle, *Mesh optimization*, Proceedings of SIGGRAPH '93 (Annaheim, California, August 1–6, 1993). In *Computer Graphics* Proceedings, Annual Conference Series, 1993,19–26.

[15] H. Hoppe, T. Derose, T. Duchamp, M. Halstead, H. Jin, J. McDonald & W. Stuetzle, *Piecewise smooth surface reconstruction*, Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994). In *Computer Graphics* Proceedings, Annual Conference Series, 1994.

[16] D.T.L. Jones, *Proton therapy: the promise of precision.*, Bulletin International Radiation Physics Society **15**, 3/4, 2001, pp. 4-13.

[17] S.B. Kang, J. Webb, C.L. Zitnich & T. Kanade, *An active multibaseline stereo system with real-time image acquisition*, School of Computer

Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 1994.

[18] G. McRae, *Point And Triangle*, **http://www.mcraefamily.com/MathHelp/Geometry/PointAndTriangle2.html**.

[19] N. Muller & R. van Rooyen, *CT-scanner Room Camera Positioning*, Medical Radiation Group, iThemba LABS, June 2002.

[20] S. Nakamura, *Applied numerical methods in C*, Prentice-Hall International Editions, Inc., 1993, pp. 25, pp.568.

[21] M. Okutomi & T. Kanade, *A multiple-baseline Stereo*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 4 April 1993.

[22] R.D. Rogus, R.L. Stern & H.D. Kubo, *Accuracy of a photogrammetry-based patient positioning and monitoring system for radiation therapy*, February 1999.

[23] M. Segal, C. Korobkin, R. van Widenfelt, J. Foram & P. Haeberli, *Fast Shadows and Lighting Effects Using Texture Mapping*, SIGGRAPH '92 Chicago, July 26-31, 1992.

[24] G. van der Vlugt, *A real-time photogrammetric system for patient positioning in proton therapy*, Department of Surveying and Geodetic Engineering, University of Cape Town, Commission V, 1989.

[25] R. van Rooyen, *Optical Tests for the CT-scanner SPG Setup*, Medical Radiation Group, iThemba LABS, 28 March 2003.

[26] R. van Rooyen, *Lens Specifications for CT-scanner room*, Medical Radiation Group, iThemba LABS, July 2002.

[27] D. Wagener, *Feature Tracking and Pattern Registration*, Master's thesis, The University of Stellenbosch and iThemba LABS, May 2003.

[28] A. Watt, *3D Computer Graphics*, Third Edition, Addison-Westley Publishing Company, 2000.

[29] L.M. Wong, C. Dumont $ A.M. Abidi, *Next best view system in a 3-D object Modeling task*, IRIS lab, Department of Electrical and Computer Engineering, March 2003.

[30] *OpenGL Programming Guide*, Addison-Westley Publishing Company, 1992.

[31] *NeHe Productions (OpenGL)*, **http://www.nehe.gamedev.net**.

[32] *Polygon Scan Conversion*, **http://xarch.tu-graz.ac.at/home/rurban/news/comp.graphics.algorithm**.

[33] *Ray tracing primitives*, **http://www.cl.cam.ac.uk/Teaching/2000/AGraphHCI/SMEG/**.

[34] *OpenGL*, **www.opengl.org**.